

# A COMPARATIVE STUDY OF ADAPTIVE ALGORITHMS WITH APPLICATIONS TO BEAMFORMING

**A THESIS**

*submitted in fulfilment of the  
requirements for the award of the degree*

*of*

**DOCTOR OF PHILOSOPHY**

*in*

**ELECTRONICS AND COMPUTER ENGINEERING**

**By**

**JAGADEESHA. S. N.**



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING  
UNIVERSITY OF ROORKEE  
ROORKEE-247 667 (INDIA)**

**OCTOBER, 1994**

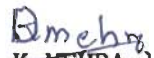
## CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **A Comparative Study of Adaptive Algorithms with Applications to Beamforming** in fulfilment of the requirement for the award of the **Degree of Doctor of Philosophy** and submitted in the Department of **Electronics and Computer Engineering** of the University is an authentic record of my own work carried out during a period from March 1991 to October 1994 under the supervision of Dr. D.K.Mehra and Dr.S.N.Sinha.

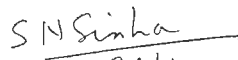
The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other university.

  
( S. N. JAGADEESHA )

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

  
( D. K. MEHRA )

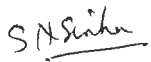

Professor  
Department of Electronics  
and Computer Engineering  
University of Roorkee,  
Roorkee - 247 667,  
India

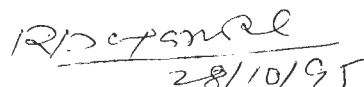
  
29/10  
( S. N. SINHA )


Reader  
Department of Electronics  
and Computer Engineering  
University of Roorkee,  
Roorkee - 247 667  
India

Date :-

The Ph.D. Viva-Voce examination of S.N. Jagadeesha  
Research Scholar, has been held on 28-10-95.

   
Signature of  
Supervisors

  
28/10/95  
Signature of  
H.O.D.

  
Signature of  
External  
Examiner

## ABSTRACT

Adaptive arrays are currently the subject of extensive investigations, as a means for reducing the vulnerability of the reception of desired signals to the presence of interference signals in radar, sonar, seismic and communication systems. The principal reason behind this widespread interest lies in their ability to sense automatically the presence of interference noise sources and to suppress them, while simultaneously enhancing the desired signal reception without the prior knowledge of the signal/interference environment. The interference signals may not only consist of deliberate electronic counter measures, nonhostile RF interferences, clutter scatter returns and natural noise sources but also coherent interferences. Coherent interferences can arise when multipath propagation is present or when "smart" jammers deliberately introduce coherent jamming by retrodirecting the signal energy to the receiver. Also, the signal environment may consist of either narrowband or broadband signal and interferences.

An adaptive array can be best described as a collection of sensors, feeding a weighting and summing network, with automatic signal dependent weight adjustment to reduce unwanted signals and/or emphasize the desired signal. In the case of broadband adaptive arrays, a tapped delay line is connected behind each sensor to compensate for the inter-element phase shift. The weight coefficients are adjusted recursively using suitable algorithms. In an adaptive array, the interference suppression is obtained by appropriately steering beam pattern nulls in the direction of interference sources,

while signal reception is maintained by preserving desirable main lobe features. Therefore, an adaptive array system relies heavily on spatial characteristics to improve the output signal-to-noise ratio (SNR).

A wide range of algorithms have been reported in the signal processing literature which can be used for adjusting the weights of an adaptive array. These include the conventional least-squares (LS) solution by direct matrix inversion or by Cholesky factorization, the classical least-mean-square (LMS) algorithm, the recursive least-square (RLS) algorithm, the fast RLS algorithms, QR decomposition algorithms based on Givens, Householders and Modified Gram-Schmidt techniques, and the rotation based fast RLS algorithms. Some of these algorithms are suitable for implementation using VLSI technology. Moreover, due to the recent advances in parallel computing architectures and VLSI technology, various computational, numerical and architectural concepts have merged. Consequently, it is becoming increasingly difficult to comprehend the interrelationships and tradeoffs among these concepts and approaches. A few of the above techniques, viz, the direct matrix inversion and the LMS algorithm, have been widely studied in the context of adaptive arrays. The difficulties in obtaining the inverse of the correlation matrix, when the matrix is ill conditioned, and the slow convergence and dependence of the time constant on the eigenvalues in LMS algorithm, make these techniques less attractive for application to adaptive arrays.

The QRD-LS algorithm based on Givens rotations has been recommended in the literature for narrowband adaptive array applications. This algorithm has fast convergence and is numerically

stable but, unfortunately, it is computationally expensive because of the square-root operations involved. Some of the other techniques, viz, the RLS algorithm in the case of narrowband beamformers and multichannel fast transversal filters (MFTF) and QRD-multichannel lattice algorithms for broadband arrays have been discussed only briefly in the literature and detailed investigations have not been carried out so far. The recursive modified Gram-Schmidt (RMGS) and the multichannel least-square Lattice (MLSL) algorithms have not been studied at all in the context of adaptive arrays.

The adaptive arrays based on above mentioned algorithms are effective in suppressing the interferences and enhancing the desired signal reception in a noncoherent signal environment. However, these techniques fail to suppress the coherent interferences. To overcome this problem, methods such as the structured correlation matrix method (redundancy averaging) and the spatial smoothing preprocessing scheme have been proposed in the literature. Of the two, the spatial smoothing scheme is more attractive and has received relatively wider attention. Several modifications of this scheme have also been proposed in the literature. Of these, the modified or forward/backward spatial smoothing scheme is important. However, the adaptive implementation in various algorithm based arrays has not received much attention so far.

This work encompasses the study of adaptive arrays covering the above aspects. A comparative study of the structured correlation matrix method and the spatial smoothing scheme using an optimum beamformer revealed that the structured correlation matrix method introduces a bias while placing nulls in the direction of

interferences. Also, the method is not suitable for broadband adaptive arrays as in this case the correlation matrix is nontoeplitz even in noncoherent situation. Moreover, the adaptive implementation of this method in various algorithm based processor is not possible, where as the spatial smoothing scheme is a practical method to suppress coherent interferences in an adaptive array.

We next consider the study of adaptive arrays based on recursive least-square algorithms having a computational complexity of the  $O(P^2)$ , where 'P' is the number of sensors in the array. It has been found that the conventional RLS algorithm based array suffers from numerical instability and fails to produce nulls in the direction of interferences arriving from endfire directions. Though the QRD-LS array based on Givens rotations has excellent numerical properties and superior nulling performance, it is computationally expensive because of the involvement of square-root operations. As an alternative, we have proposed the use of RMGS algorithm and its error feedback version for adaptive beamformers. These algorithms can also be implemented using systolic structures. The arrays based on these algorithms have numerical properties and nulling performance that are comparable with Givens rotation based QRD-LS array and at the same time, they are computationally less expensive. Therefore, the proposed RMGS algorithms based arrays represent a good compromise between the numerical stability and computational cost in the adaptive beamforming problems. For broadband arrays, however, these algorithms turn out to be computationally expensive with a complexity of the  $O(P^2M^2)$ , where 'M' is the number of taps in each delay line.

Next, we consider the arrays based on Fast-RLS algorithms for realizing broadband arrays. We have proposed the multichannel least-square Lattice [MLSL] algorithm for broadband adaptive array which has a computational complexity of  $O(P^3M)$ . Using MLSL algorithm as the basis, we formulate the Givens rotation based QRD-MLSL algorithm and apply it to the adaptive beamforming problem. We then derive the MFTF algorithm and study the adaptive arrays based on these algorithms. The algebraic approach has been used to derive these algorithms. Of the three broadband arrays realized, the MFTF algorithm has the least computational complexity.

Finally, we have considered, the spatial smoothing scheme and the forward/backward spatial smoothing scheme as an effective means to suppress coherent interferences. Our studies have revealed that, in optimum beamformers, both the methods are effective in placing nulls in the direction of coherent interferences. The adaptive implementation of spatial smoothing scheme on the QR decomposition algorithms, such as QRD-LS and RMGS algorithm based arrays, has received little attention so far. We have proposed a method of implementing spatial smoothing scheme on the arrays based on these algorithms. In this method, the elements of the upper triangular matrix are smoothed first and after a fixed number of snapshots, this smoothed upper triangular matrix is used to compute the optimum weights of the beamformer. The proposed method has been tested through computer simulations and produces deep nulls in the direction of coherent interferences. In the forward/backward spatial smoothing scheme, the signals of the respective forward and complex conjugated backward subarrays are first averaged. Then, the resultant signals are

used to smooth the weights or the elements of upper triangular matrix. Our numerical experiments show that the conventional spatial smoothing scheme has a much superior nulling performance as compared to the forward/backward spatial smoothing scheme.

The performance of various algorithms has been evaluated for the narrowband and broadband arrays in noncoherent as well as coherent interference environment, using computer simulations. Convergence characteristics of various beamformers have been tested by computing the residual power as a function of the number of adaptation samples. The comparative study has revealed that, QRD-LS array based on Givens rotations has the fastest convergence and the least residual power. The RMGS algorithm based array with error feedback has characteristics comparable with those of the QRD-LS array.

The nulling performance of various arrays has been studied with the help of voltage patterns. In the case of broadband arrays, the output waveforms has also been extracted to demonstrate the ability of the beamformers to track the desired signal.



## ACKNOWLEDGEMENTS

The author wishes to express his deep sense of gratitude to Dr. D.K.Mehra and Dr. S.N.Sinha for their keen interest, painstaking supervision and invaluable help throughout the course of this work. They have been generous in undertaking comprehensive discussion and meticulous reading and reviewing of the text, without which the work could not have come to its shape.

The facilities and working conditions provided by the Head, Department of Electronics and Computer Engineering are gratefully acknowledged.

The author is very much grateful to the principal, University B.D.T. College of Engineering, Davangere for sponsoring him under the quality improvement program and granting necessary leave in this regard.

Finally, Thanks are due to all friends who helped the author in one way or the other.

( S.N. JAGADEESHA )

# TABLES OF CONTENTS

	PAGE NO.
ABSTRACT	i
ACKNOWLEDGEMENTS	vii
CONTENTS	ix
LIST OF TABLES	xiii
LIST OF FIGURES	xv
CHAPTER - 1 INTRODUCTION	1
1.1 SURVEY OF THE EARLIER WORK	2
1.2 STATEMENT OF THE PROBLEM	11
1.3 ORGANIZATION OF THE DISSERTATION	13
CHAPTER - 2 ADAPTIVE BEAMFORMERS FOR NON-COHERENT AND COHERENT INTERFERENCE SUPPRESSION - A REVIEW	15
2.1 THE LEAST-MEAN-SQUARE ADAPTIVE ARRAY	15
2.1-1 The Least-Mean-Square [LMS] Algorithm	25
2.1-2 Sample Results	26
2.2 THE BROADBAND ADAPTIVE ARRAY	31
2.2-1 Adaptive Array with Constraints	34
2.2-2 Broadband Signal Simulation	39
2.2-3 Sample Results	40
2.3 BEAMFORMING IN THE PRESENCE OF COHERENT SIGNALS	41
2.3-1 Spatial Smoothing Preprocessing Scheme [SSPS]	45
2.3-2 Structured Correlation Matrix Method [SCMM]	55
2.3-3 Sample Results.	56

2.4 A COMPARISON OF STRUCTURED CORRELATION MATRIX METHOD [SCMM] AND THE SPATIAL SMOOTHING PREPROCESSING SCHEME (SSPS)	57
2.5 SUMMARY	63
<b>CHAPTER - 3 RECURSIVE LEAST SQUARES ALGORITHMS FOR ADAPTIVE BEAMFORMING</b>	<b>65</b>
3.1 THE EXACT LEAST-SQUARE ERROR CRITERION	67
3.2 THE RECURSIVE LEAST-SQUARE (RLS) ALGORITHM	69
3.3 THE QRD-LS ALGORITHM	70
3.3-1 Exact Least-Square Error Criterion in the Data Domain	71
3.3-2 QRD-LS Algorithm	72
3.3-3 Recursive Implementation	74
3.3-4 Givens Rotations	76
3.3-5 Direct Extraction of Residuals	81
3.3-6 Systolic Array Implementation	84
3.4 THE RECURSIVE MODIFIED GRAM-SCHMIDT [RMGS] ALGORITHM	89
3.4-1 The Modified Gram-Schmidt Procedure	90
3.4.2 The Complex Recursive Modified Gram-Schmidt [RMGS] Algorithm	94
3.4-3 Systolic Array Implementation	98
3.5 BROADBAND ADAPTIVE ARRAYS	100
3.6 SIMULATION RESULTS	102
3.6-1 Narrowband Arrays	103
3.6-2 Broadband Arrays	120
3.7 CONCLUSIONS	151
<b>CHAPTER - 4 FAST RECURSIVE LEAST-SQUARE ALGORITHMS FOR ADAPTIVE BEAMFORMERS</b>	<b>155</b>
4.1 THE MULTICHANNEL LEAST-SQUARE LATTICE [MLSL] ALGORITHM	157

4.2 THE QR-MLSL ALGORITHM	172
4.3 THE MULTICHANNEL FAST TRANSVERSAL FILTER [MFTF] ALGORITHM	179
4.4 SIMULATION RESULTS	185
4.5 CONCLUSIONS	208
<b>CHAPTER - 5 ADAPTIVE BEAMFORMERS FOR COHERENT INTERFERENCE SUPPRESSION</b>	<b>211</b>
5.1 THE FORWARD/BACKWARD SPATIAL SMOOTHING SCHEME	213
5.2 ADAPTIVE IMPLEMENTATION OF THE SPATIAL SMOOTHING SCHEME	216
5.2-1 Adaptive Processing	217
5.2-1.1 Implementation using weight oriented algorithms	218
5.2-1.2 Implementation using QR decomposition algorithms	219
5.3 ADAPTIVE IMPLEMENTATION OF THE FORWARD/ BACKWARD SPATIAL SMOOTHING SCHEME	226
5.3-1 Implementation Using Weight Oriented Algorithms	228
5.3-2 Implementation Using QR decomposition Algorithms	228
5.4 COMPUTER SIMULATIONS	233
5.5 CONCLUSIONS	245
<b>CHAPTER - 6 CONCLUSIONS</b>	<b>249</b>
6.1 SUGGESTIONS FOR FURTHER WORK	254
<b>APPENDIX - A COMPLEX GRADIENT OPERATOR</b>	<b>257</b>
<b>APPENDIX - B DERIVATION OF TIME UPDATE EQUATION FOR <math>r_{ij}(n)</math></b>	<b>263</b>

APPENDIX - C	ORDER UPDATE EQUATION FOR $\alpha_1(n)$	267
APPENDIX - D	RELATIONS BETWEEN MLSL AND QR-MLSL ALGORITHMS	269
APPENDIX - E	DERIVATION OF EQUATION FOR $C_{-P(M+1)}^{(n+1)}$	275
REFERENCES		277
RESEARCH PAPERS OUT OF THE PROPOSED WORK		285

## LIST OF TABLES

TABLE NO.	PAGE
2.1 Summary of the LMS algorithm.	27
2.2 Interference parameters for narrowband beamformers.	27
2.3 Interference parameters for the Frost Array.	41
2.4 Interference parameters.	56
2.5 Interference parameters.	59
2.6 Null depths produced by SCMM and SSPS beamformers.	60
3.1 The RLS algorithm.	70
3.2 The recursive QRD-LS algorithm.	77
3.3 The QRD-LS algorithm using Givens rotations.	83
3.4 Modified Gram-Schmidt [MGS] algorithm.	93
3.5 The complex recursive modified Gram-Schmidt algorithm.	96
3.6 The complex RMGS algorithm using error feedback.	97
3.7 Parameters of the interferences in example 3.6-1.1.	103
3.8 Weight coefficients of the four beamformers.	109
3.9 Null depths produced by the beamformers in dB.	109
3.10 Parameters of interferences in example 3.6-1.4.	120
3.11 Parameters of the broadband interferences in example 3.6-2.1.	125
3.12 Parameters of the broadband interferences in example 3.6-2.5.	146
3.13 Computational complexity of exact least-square beamformers.	152

4.1	The MLSL algorithm using a posteriori errors.	169
4.2	Initialization of the MLSL algorithm with a posteriori errors.	171
4.3	The QR-MLSL algorithm.	178
4.4	The multichannel fast transversal filter [MFTF] algorithm.	184
4.5	Computational complexity of the fast RLS algorithm based broadband beamformers.	208
5.1	RLS algorithm for adaptive implementation of spatial smoothing scheme.	221
5.2	RMGSEF algorithm for adaptive implementation of the spatial smoothing scheme.	223-224
5.3	The QRD-LS algorithm for implementation of spatial smoothing scheme.	225
5.4	RLS algorithm for adaptive implementation of forward/backward scheme.	230
5.5	RMGSEF algorithm for adaptive implementation of the forward/backward smoothing scheme.	232
5.6	Interference parameters for example 5.4-1.	234

## LIST OF FIGURES

FIG. NO.	PAGE NO.
2.1 Adaptive array with automatic weight adjustment.	16
2.2. Voltage patterns of narrowband beamformers with interferences arriving at $30^{\circ}$ , $-30^{\circ}$ , $60^{\circ}$ and $-60^{\circ}$ .	29
2.3 Convergence characteristics of LMS array	30
2.4 Tapped-Delay line multichannel processor for wideband signal processing.	32
2.5 Broadband antenna array equivalent processor.	37
2.6 Generation of a broadband signal.	40
2.7 Desired broadband signal.	42
2.8 Voltage pattern of Frost broadband array with interferences arriving at $50^{\circ}$ and $-50^{\circ}$ .	42
2.9 A uniform linear array of 'P + L' sensors divided into overlapping subarrays of size 'P' each.	47
2.10 Voltage patterns with desired signal at $0^{\circ}$ ; coherent interferences at $15^{\circ}$ and $55^{\circ}$ and noncoherent interferences at $-30^{\circ}$ and $-70^{\circ}$ .	58
2.11 Voltage patterns of the SSPS and SCMM beamformers with interferences at $-10^{\circ}$ and $10^{\circ}$ .	61
2.12 Voltage patterns of the SSPS and SCMM beamformers with interferences at $-50^{\circ}$ and $30^{\circ}$ .	61
2.13 Voltage patterns of the SSPS and SCMM beamformers with interferences at $-70^{\circ}$ and $70^{\circ}$ .	62
3.1 Systolic array implementation of the recursive QRD-LS algorithm using Givens rotations.	85
3.2 Cells for recursive QRD-LS algorithm : (a) boundary cell, (b) internal cell.	88



3.3	Cells for linear systolic array : (a) boundary cell, (b) internal cell	88
3.4	Systolic array implementation of the RMGS algorithm.	99
3.5	Convergence characteristics of narrowband beamformers with interferences arriving at $30^\circ$ , $-30^\circ$ , $60^\circ$ and $-60^\circ$ .	104-105
3.6	Voltage patterns of narrowband beamformers with interferences arriving at $30^\circ$ , $-30^\circ$ , $60^\circ$ and $-60^\circ$ .	107-108
3.7	Convergence characteristics of narrowband beamformers with interferences arriving at $5^\circ$ , $-5^\circ$ , $85^\circ$ and $-85^\circ$ .	111-112
3.8	Voltage patterns of narrowband beamformers with interferences arriving at $5^\circ$ , $-5^\circ$ , $85^\circ$ and $-85^\circ$ .	113-114
3.9	Convergence characteristics of narrowband beamformers with interferences arriving at $60^\circ$ , $-60^\circ$ , $80^\circ$ and $-80^\circ$ .	116-117
3.10	Voltage patterns of narrowband beamformers with interferences arriving at $60^\circ$ , $-60^\circ$ , $80^\circ$ and $-80^\circ$ .	118-119
3.11	Convergence characteristics of narrowband beamformers with interferences arriving at $15^\circ$ , $-15^\circ$ , $30^\circ$ , $-30^\circ$ , $45^\circ$ , $-45^\circ$ , $60^\circ$ , $-60^\circ$ , $75^\circ$ and $-75^\circ$ .	121-122
3.12	Voltage patterns of narrowband beamformers with interferences arriving at $15^\circ$ , $-15^\circ$ , $30^\circ$ , $-30^\circ$ , $45^\circ$ , $-45^\circ$ , $60^\circ$ , $-60^\circ$ , $75^\circ$ and $-75^\circ$ .	123-124
3.13	Convergence characteristics of broadband beamformers with interferences arriving at $60^\circ$ and $-60^\circ$ .	126-127
3.14	Voltage patterns of broadband beamformers with interferences arriving at $60^\circ$ and $-60^\circ$ .	129-130
3.15	Output waveforms of broadband beamformers.	131-132
3.16	Convergence characteristics of broadband beamformers with interferences arriving at $50^\circ$ and $60^\circ$ .	133-134
3.17	Voltage patterns of narrowband beamformers with interferences arriving at $50^\circ$ and $60^\circ$ .	135-136

3.18	Convergence characteristics of broadband beamformers with interferences arriving at $5^{\circ}$ and $-5^{\circ}$ .	137-138
3.19	Voltage patterns of narrowband beamformers with interferences arriving at $5^{\circ}$ and $-5^{\circ}$ .	140-141
3.20	Convergence characteristics of broadband beamformers with interferences arriving at $80^{\circ}$ and $-80^{\circ}$ .	142-143
3.21	Voltage patterns of broadband beamformers with interferences arriving at $80^{\circ}$ and $-80^{\circ}$ .	144-145
3.22	Convergence characteristics of broadband beamformers with interferences arriving at $30^{\circ}$ , $-30^{\circ}$ , $60^{\circ}$ and $-60^{\circ}$ .	147-148
3.23	Voltage patterns of broadband beamformers with interferences arriving at $30^{\circ}$ , $-30^{\circ}$ , $60^{\circ}$ and $-60^{\circ}$ .	149-150
4.1	Convergence characteristics of fast RLS broadband beamformers with interferences arriving at $60^{\circ}$ and $-60^{\circ}$ .	187-188
4.2	Output signal waveforms of the three fast RLS broadband beamformers with interferences arriving at $60^{\circ}$ and $-60^{\circ}$ .	189-190
4.3	Convergence characteristics of fast RLS broadband beamformers with interferences arriving at $50^{\circ}$ and $60^{\circ}$ .	192-193
4.4	Output signal waveforms of the three fast RLS broadband beamformers with interferences arriving at $50^{\circ}$ and $60^{\circ}$ .	194-195
4.5	Convergence characteristics of fast RLS broadband beamformers with interferences arriving at $5^{\circ}$ and $-5^{\circ}$ .	197-198
4.6	Output signal waveforms of the three fast RLS broadband beamformers with interferences arriving at $5^{\circ}$ and $-5^{\circ}$ .	199-200
4.7	Convergence characteristics of fast RLS broadband beamformers with interferences arriving at $80^{\circ}$ and $-80^{\circ}$ .	201-202

4.8	Output signal waveforms of the three fast RLS broadband beamformers with interferences arriving at $80^{\circ}$ and $-80^{\circ}$ .	203-204
4.9	Voltage pattern of MFTF beamformer with interferences arriving at $60^{\circ}$ and $-60^{\circ}$ .	206
4.10	Voltage pattern of MFTF beamformer with interferences arriving at $50^{\circ}$ and $60^{\circ}$ .	206
4.11	Voltage pattern of MFTF beamformer with interferences arriving at $5^{\circ}$ and $-5^{\circ}$ .	207
4.12	Voltage pattern of MFTF beamformer with interferences arriving at $80^{\circ}$ and $-80^{\circ}$ .	207
5.1	The forward/backward spatial smoothing scheme.	214
5.2	Flow-chart for adaptive implementation of spatial smoothing scheme using the RLS adaptive processor.	220
5.3	Flow-chart for adaptive implementation of spatial smoothing scheme using RMGSEF adaptive processor.	222
5.4	Flow-chart for adaptive implementation of forward/backward smoothing scheme using RLS adaptive processor.	229
5.5	Flow-chart for adaptive implementation of forward/backward smoothing scheme using RMGSEF adaptive processor.	231
5.6	Voltage patterns of SSPS and FBSS adaptive beamformers with interferences arriving at $30^{\circ}$ , $-30^{\circ}$ , $60^{\circ}$ and $-60^{\circ}$ .	235
5.7	Voltage patterns of SSPS and FBSS adaptive beamformers with interferences arriving at $30^{\circ}$ , $-30^{\circ}$ , $65^{\circ}$ and $-65^{\circ}$ .	237
5.8	Voltage patterns of SSPS and FBSS adaptive beamformers with interferences arriving at $30^{\circ}$ , $-45^{\circ}$ , $60^{\circ}$ and $-75^{\circ}$ .	239
5.9	Voltage patterns of SSPS and FBSS adaptive beamformers with interferences arriving at $5^{\circ}$ , $-5^{\circ}$ , $75^{\circ}$ and $-75^{\circ}$ .	240

5.10 Voltage patterns of SSPS and FBSS RMGSEF adaptive beamformers with interferences arriving at $-30^{\circ}$ , $-35^{\circ}$ , $60^{\circ}$ and $65^{\circ}$ .	242
5.11 Voltage patterns of SSPS and FBSS RMGSEF adaptive beamformers with interferences arriving at $30^{\circ}$ , $40^{\circ}$ , $50^{\circ}$ and $60^{\circ}$ .	244
5.12 Voltage patterns of SSPS and FBSS RMGSEF adaptive beamformers with interferences arriving at $35^{\circ}$ , $40^{\circ}$ , $45^{\circ}$ and $50^{\circ}$ .	246

SNR	Signal - to - noise ratio	1
SSPS	Spatial smoothing preprocessing scheme	9
SSPS - RMGSEF	Spatial smoothing preprocessing scheme - Recursive modified Gram - Schmidt with error feedback	234
SSPS - QRD - LS	Spatial smoothing preprocessing scheme - QR decomposition - Least - square	234

## ABBREVIATIONS

<u>Abbreviation</u>	<u>Expansion</u>	<u>First Appearance</u>
		Page No.
DOA	Direction - of - arrival	10
Fast - RLS	Fast recursive Least - square	4
FBSS	Forward/Backward spatial smoothing scheme	10
FBSS - RMGSEF	Forward/Backward spatial smoothing scheme - recursive modified Gram - Schmidt algorithm with error feedback	234
FBSS - QRD - LS	Forward/Backward smoothing scheme - QR decomposition - Least - square	234
FTF	Fast transversal filter	5
LMS	Least - mean - square	3
LS	Least - square	4
LSL	Least - square - Lattice	7
MFTF	Multichannel fast transversal filter	11
MGS	Modified Gram - Schmidt	5
QRD - LS	QR decomposition - Least - square	5
QR - LSL	QR Least - square Lattice	7
QR - MLSL	QR - multichannel Least - square Lattice	9
RLS	Recursive Least - square	4
RMGS	Recursive modified Gram - Schmidt	6
RMGSEF	Recursive modified Gram - Schmidt with error feedback	6
SCMM	Structured correlation matrix method	10

SNR	Signal - to - noise ratio	1
SSPS	Spatial smoothing preprocessing scheme	9
SSPS - RMGSEF	Spatial smoothing preprocessing scheme - Recursive modified Gram - Schmidt with error feedback	234
SSPS - QRD - LS	Spatial smoothing preprocessing scheme - QR decomposition - Least - square	234

## CHAPTER - 1

### INTRODUCTION

Adaptive arrays are currently the subject of extensive investigation as a means to reduce the vulnerability of reception of desired signal to the presence of interference signals in radar, sonar and communication systems. Interference signals may consist of deliberate electronic countermeasures, nonhostile RF interference, clutter scatter returns and natural noise sources. Adaptive arrays have the ability to sense automatically the presence of interference sources and to suppress the signal from these sources, while simultaneously enhancing the desired signal reception without any prior knowledge of signal / interference environment.

An adaptive array consists of an array of sensor elements and a real time adaptive signal processor. The adaptive processor automatically adjusts the array beam sensitivity pattern so that a measure of the quality of the array performance is improved. Interference signal suppression is obtained by appropriately steering the beam pattern nulls and reducing sidelobe levels in the direction of interference sources. At the same time, the desired signal reception is maintained by preserving desirable main lobe features. An adaptive array system, therefore, relies heavily on spatial characteristics to improve the output signal-to-noise ratio (SNR).

In the following sections, we present a brief summary of the earlier work carried out in the field of adaptive arrays, followed by the statement of the problem and organization of the material in this dissertation.



## 1.1 SURVEY OF THE EARLIER WORK

The term 'Adaptive Antenna' was first used by Van Atta [60] to describe a self phasing antenna array that automatically reradiated a signal in the direction from which it was received, thereby acting as a retrodirective system.

The development of phase locked loops was a major step that made self-steering or self-phasing type of adaptive array possible [1]. A phase-lock-loop array operates by aligning the phase of the signal from each element with that of a reference signal, before summing the signals to produce the array output. In the 1960's, this type of arrays were extensively studied. Some typical phase-lock-loop arrays are described in a special issue [23] on active and adaptive antennas. However, these arrays are vulnerable to interferences because the phase-lock-loops can track only one signal at a time. If an interference signal arrives that is stronger than the desired signal, it can easily capture the antenna beam.

In 1957, Howells invented a sidelobe cancellor capable of automatically nulling out the effect of one jammer [20]. In 1966, Applebaum derived the control law governing the operation of the adaptive array antenna. The algorithm derived by Applebaum is based on maximizing the signal-to-noise ratio at the antenna array output and included the sidelobe cancellor as a special case. His 1966 report was reprinted in the 1976 issue of IEEE Transactions on Antennas and propagation.

In 1966, Shor [54] introduced an adaptive array based on a maximum signal-to-noise ratio (SNR) concept like the Applebaum array, but differing from the latter in that, the Shor feedback loops are

based on a steepest ascent optimization of signal-to-noise ratio. However, this array has not received much attention in the literature because of its complexity.

Widrow and his coworkers at Stanford University proposed the LMS algorithm for weight adjustment in adaptive arrays in 1967 [65]. The LMS algorithm is based on the method of steepest descent [22] and minimizes the mean-square-error between the actual array output and the ideal array output. There is little difference between the Applebaum array and the LMS array from a mathematical viewpoint [8]. Rather, the difference between the two is more of application. The Applebaum array is useful when the desired signal arrival angle is known in advance, whereas, the LMS array is useful when a reference signal correlated with the desired signal is available.

The adaptive arrays based on the LMS and maximum SNR algorithms are simple from the point of view of implementation and computational complexity. However, their slow rate of convergence limits their applications to adaptive control problems presented by small communications and data collection arrays [41]. For more complex problems, such as command and control of remote vehicles or rapid angular tracking in radar communication systems, algorithms with much faster rate of convergence are often required. We briefly summarize here the work carried out in the field of adaptive signal processing for the above purpose.

The area of adaptive signal processing has grown at a rapid rate during the last decade. The demand for high performance systems, combined with the availability of ever increasing computational power, has motivated the search for more sophisticated signal processing

algorithms capable of operating in uncertain, time varying environments. The basic aim is to reduce the computational complexity to a level comparable to that of LMS algorithm and, at the same time, achieve a much faster convergence. This has led to several new algorithms. These algorithms are based on the method of least-squares in which the index of performance minimized is the sum of weighted error squares. Depending on the structure used for implementing the adaptive filter, four different classes of algorithms are identified that originate from the method of least-squares.

*(i) Recursive Least-squares Algorithm*

The RLS algorithm bears a close relationship with the Kalman filter algorithm [17] and has been derived independently by several investigators. However, the original reference on the RLS algorithm appears to be that of Plackett [47].

This algorithm assumes the use of a tapped delay line structure (transversal filter) as the basis of the adaptive filter. The transversal filter is a continuous time device whose input is formed as a linear combination of voltages taken from uniformly spaced taps in a non-dispersive delay line [27]. The derivation of the algorithm is based on a result in linear algebra known as matrix inversion lemma [22]. The RLS algorithm provides a much faster rate of convergence than the LMS algorithm, at the expense of increased computational complexity.

*(ii) Fast Recursive Least-squares (Fast-RLS) Algorithm*

By exploiting certain properties that arise in the case of

serialized data, various schemes have been developed to overcome the computational complexity of RLS algorithm. There are two families of fast RLS algorithms viz, the fast transversal filter (FTF) algorithm [6] and the fast Lattice algorithms [12]. The FTF algorithm uses a parallel combination of four transversal filters. On the other hand, the Lattice algorithm uses multistage lattice predictor as the structural basis of the adaptive filter to resolve the issue of computational complexity. This predictor consists of a cascade of stages, each in the form of lattice. An important property of multistage lattice predictor is that its individual stages are decoupled from each other in a time averaged sense. This property is exploited in the derivation of recursive least-square lattice algorithm [12]. The multistage lattice filter has a highly pipelined modular structure and is well suited for implementation using very large scale integration (VLSI) technology.

These algorithms effectively retain the advantages of the conventional RLS algorithm and yet, their computational complexity is reduced to a level comparable to that of simple LMS algorithm.

(iii) *QR Decomposition Least-square Algorithms*

These algorithms are based on orthogonal transformations. There are three well established orthogonalization techniques, viz, Givens, Householders and modified Gram-Schmidt procedures [29], which are termed as QR techniques in mathematics

literature. This term comes from the fact, that these algorithms perform decomposition of the data matrix as a product of an orthogonal matrix  $Q$  and an upper triangular matrix  $R$ .

The QR decomposition algorithm based on Givens rotations is popularly known as QRD-LS algorithm and was introduced in 1981 by Gentleman and Kung [15]. The algorithm configuration consists of two stages. The first stage involves an orthogonal triangularization process that is achieved by applying the QR decomposition method, based on Givens procedure, directly to the input data matrix in a recursive fashion. As new input data enters the computation, the recursive procedure conducts a linear transformation of the input data matrix into upper triangular form. At the end of entire recursion, this special structure of the data matrix is exploited to compute the least-square weight vector. The algorithm is stable, robust, rapidly convergent but computationally expensive.

Ling and Proakis [36] have derived another equally efficient, but computationally less expensive, QR decomposition recursive least-square algorithm using modified Gram-Schmidt procedure [61]. This algorithm is known as recursive modified Gram-Schmidt (RMGS) algorithm. They have also derived an error feedback form of this algorithm which has better numerical properties compared to the basic RMGS algorithm. These algorithms also act directly on the input data matrix in a recursive fashion and convert it into an upper triangular form.

All the above mentioned algorithms may be implemented using

a systolic array which represents a highly efficient and dedicated structure. The systolic structure offers the desirable features of modularity, local interconnections, and highly pipelined and synchronized parallel processing. All these properties make systolic arrays particularly well suited for VLSI implementation.

(iv) *The QR-LSL Algorithm*

The fast-RLS algorithms, viz, the FTF and least-square lattice algorithms have a computational complexity that is comparable to the simple LMS algorithm. However, they are not very stable in the numerical sense. On the other hand, QR decomposition algorithms are highly stable but computationally expensive. Due to this reason, there has been an increasing interest in rotation-based fast-RLS algorithms which are numerically more robust. These algorithms are also known as fast QR algorithms. Cioffi [7] has developed a fixed order fast adaptive ROTOR'S (FAR) algorithm which can be interpreted as a QR-based fast Kalman algorithm. This algorithm can be implemented with a pipelined array of processors called "ROTORS" and "CISORS".

Ling [37] has described an order-recursive-rotation- based QR-LSL algorithm using the relationship between Givens rotation based algorithms and modified Gram-Schmidt method. Proudler et al, [49] have suggested another approach of deriving a QR-LSL algorithm from the Givens rotation based

QRD-LS algorithm. Regalia and Bellanger [51] introduced a family of QR-lattice fast least-square algorithms by exposing the duality between the Gram-Schmidt orthogonalization and the lattice algorithm. Yang and Bohme [68] have pointed out a fourth possibility. It differs from previous ones in that it starts directly with lattice recursions. In this approach, the LSL algorithm is reformulated to a rotation-based one by suitable transformation of the filter quantities. This is achieved in two steps: first by the use of Cholesky decomposition to transform from covariance domain to information domain and, subsequently, using the time recursive QR-update technique to incorporate new data into square-root factor produced by Cholesky decomposition.

These algorithms have the advantages of both the approaches, that is, the numerical robustness of the QR algorithms and the computational simplicity of lattice algorithms. Further, they can be implemented using highly pipelined structures.

Of the above mentioned algorithms, the RLS algorithms have been suggested by Brennan et al, [4] and Baird [3] for adaptive beamforming applications, but extensive results on the performance of RLS beamformer are not available in the literature. Ward et al [62] have proposed the QRD-LS algorithm based on Givens rotations to the adaptive beamforming problem. This beamformer has been reported to have excellent numerical properties and nulling abilities, but is computationally expensive. Application of fast RLS algorithms, viz, FTF and LSL, and the hybrid QR-LSL algorithms to adaptive beamforming

requires their multichannel formulations. Slock and Kailath [58] have presented a form of multichannel FTF algorithm for broadband beamforming. Similarly, QR-multichannel LSL algorithm has been presented by Mcwhirter and Proudler [40] for broadband beamforming. However, they have not provided results on the performance of these algorithms in adaptive beamformers. The fast lattice (LSL), RMGS and the QR-LSL algorithm based on Cholesky factorization have not been studied so far, in the context of adaptive beamformers.

The interfering signals in the signal environment of an adaptive array may be either noncoherent or coherent with respect to the desired signal. The coherent interference problem arises due to multipath propagation or due to 'smart' jammers. Smart jammers are used in a hostile environment to induce such phenomenon deliberately.

The adaptive arrays, viz., the Howell-Applebaum array and the LMS array are designed under the assumption that the interfering signals are noncoherent with the desired signal. These conventional forms of adaptive beamformers breakdown in the presence of coherent interfering signals. This problem was initially identified by White [63], Gabriel [14] and Widrow et al, [64]. Shan and Kailath [57] described an adaptive beamformer that incorporates 'Spatial Smoothing' as a means to overcome the coherent interference problem. Spatial smoothing is a preprocessing scheme that partitions the total array of sensors into subarrays and generates the average of the subarray covariance matrices. They have also shown that when this averaged covariance matrix is used in conjunction with conventional arrays, a minimum of  $2K$  sensor elements are required to null  $(K-1)$  coherent interferences. Thus, the method suffers from reduced effective



aperture area. To overcome the problem of reduced effective aperture area, a modification to the spatial smoothing scheme, known as forward/backward spatial smoothing scheme, has been suggested by Evans [11]. This method has been further investigated by Williams et al, [66] and pillai et al, [50] in the context of direction-of-arrival estimation, using signal subspace algorithms. The method has been reported to provide a larger effective area as compared to the spatial smoothing scheme, without much increase in computational burden.

An alternative solution, known as structured correlation matrix method, to the problem of coherent interferences in adaptive beamforming has been suggested by Godara [18]. In this method, the output covariance matrix of the array is averaged along the diagonals, and each element on the diagonal is replaced by its average. This method preserves the array aperture and, thus, does not suffer from the problem of reduced effective aperture.

Of the above mentioned spatial averaging schemes, the spatial smoothing scheme has received wider attention and has been studied by various researchers [52,69] using an optimum beamformer. However, in real time applications, it is necessary to implement the scheme adaptively which can be done by using time recursive algorithms, such as, LMS, RLS and QR decomposition algorithms. The adaptive implementation of spatial smoothing scheme, using various time recursive algorithms has not received much attention. Only Shan and Kailath [57] have briefly discussed this aspect. using LMS algorithm. Similarly, implementation of the forward/backward spatial smoothing scheme for adaptive beamformers has also not received much attention. Only Park and Un [45] have discussed the adaptive

implementation of this scheme. In their scheme, the forward/backward spatial smoothing scheme is incorporated by rearranging the spatial subarray data in a parallel manner using a set of subbeamformers based upon QRD-LS algorithm. This constitutes a parallel implementation of forward/backward spatial smoothing scheme.

The signal environment of an adaptive array may consist of either narrowband or broadband signal and interferences. If the signals are broadband, a tapped delay line is connected behind each element of the array. This type of arrays have been studied extensively using LMS algorithm [65,16]. The multichannel FTF and the hybrid QR-multichannel lattice algorithms have been proposed by Slock et al, [58] and, Mcwhirter and Proudler [40], respectively for broadband beamforming. However, they have not provided any results on the performance of beamformers based on these algorithms. Moreover, it is not clearly known, whether the multichannel forms of the fast RLS algorithm retain the advantages they possess in their single channel form. The orthogonalization based QR decomposition algorithms, viz, the rotation based QRD-LS and RMGS algorithms have also not been studied, so far, in the context of broadband beamforming.

## 1.2 STATEMENT OF THE PROBLEM

The present work encompasses a study of various adaptive algorithms with reference to their application to adaptive beamforming problem. Detailed investigations have been carried out on the numerical properties, convergence characteristics and nulling abilities of the adaptive beamformers based on different algorithms.

The algorithms derived from the method of least-squares with exponentially windowed complex signals are considered for the above purpose. Application of spatial smoothing scheme and forward / backward spatial smoothing scheme for combating coherent interference problem in adaptive beamformers is also studied. Schemes to implement these two averaging schemes in adaptive beamformers have been presented.

The problem, as treated in this study, may be divided into three main parts.

- (i) A comparative study of adaptive beamformers based on RLS, RMGS, RMGSEF and rotation based QRD-LS algorithms for narrowband signal environments.
- (ii) Performance evaluation of broadband adaptive beamformers based on RLS, RMGS, RMGSEF, rotation based QRD-LS algorithm and fast RLS algorithms.
- (iii) A study of spatial averaging schemes using different adaptive algorithms to combat coherent interference in adaptive beamformers.

In all the numerical examples presented in this dissertation, 6-element uniform linear array of isotropic elements has been considered, unless otherwise specified. The desired signal of strength 0.1 has been assumed to arrive from a direction broadside to the array. Normalized centre frequency ( $\omega_0$ ) of the desired signal has been assumed to be 1. In case the desired signal is broadband, the bandwidth ( $\Delta$ ) has been assumed to be 0.5 and the normalized centre frequency is 1.

### 1.3 ORGANIZATION OF THE DISSERTATION

The work embodied in this dissertation has been arranged in six chapters. In chapter 2, we review some of the basic concepts in the area of adaptive beamforming. These include the least-mean-square(LMS) criterion, the LMS algorithm and the Frost array. Next, adaptive beamforming in coherent-interference environment is briefly reviewed and the two techniques, namely, the spatial smoothing scheme and the structured correlation matrix method, have been compared on the basis of computer simulation results.

The exact least-square algorithms have been discussed in chapter 3 in the context of adaptive beamforming. We first introduce the least-square filtering problem and present an overview of the recursive least square algorithm. Next, the exact least-square error criterion is redefined in data domain and the QRD-LS algorithm based on Givens rotations is discussed. Then, we propose the RMGS algorithm as a suitable alternative to the more computationally complex QRD-LS algorithm for the adaptive beamforming problem. Finally, the results of an extensive numerical study of these beamformers are presented followed by a discussion of their relative performances.

In chapter 4, we discuss adaptive beamformers based on least-square lattice, the hybrid multichannel QR-lattice (QR-MLSL) and the multichannel FTF algorithms. These algorithms have been derived using the algebraic approach. Finally, we discuss their suitability for beamforming applications on the basis of computer simulation results.

The forward/backward spatial smoothing scheme has been introduced in chapter 5 and schemes for adaptive implementation of

spatial smoothing and forward/backward spatial smoothing techniques, using various adaptive algorithms, have been discussed. Finally, we discuss their utility in adaptive beamformers to overcome the coherent interferences on the basis of computer simulation results.

Chapter 6 concludes the dissertation with a comparison of adaptive beamformers based on various algorithms in noncoherent signal environments followed by a comparison of spatial averaging schemes to overcome coherent interference problem.

Also included are five appendices which respectively contain the complex gradient operator, derivations of time recursive equations of the complex RMGS algorithm, proofs for the relations between lattice and hybrid QR-lattice algorithms and derivation of the equation for extended Kalman gain vector.

## CHAPTER - 2

### ADAPTIVE BEAMFORMERS FOR NON-COHERENT AND COHERENT INTERFERENCE

#### SUPPRESSION - A REVIEW

In this chapter, some of the basic concepts in the area of adaptive beamforming have been briefly reviewed. An optimum beamformer, based upon the Least-Mean-Square(LMS) criterion, is described first for the suppression of narrowband interferences which are noncoherent with the desired signal. This leads to the Weiner-Hopf equation in matrix form. The adaptive implementation is briefly explained using the well known LMS algorithm. Next, the imposition of signal protection constraint, which ensures a constant gain, in a prescribed look-direction, is discussed in the context of a broad-band adaptive array. Finally, adaptive beamforming for coherent interference suppression are briefly reviewed and the two main techniques, namely, the spatial smoothing preprocessing scheme and the structured correlation matrix method, have been compared on the basis of computer simulation results.

#### 2.1 THE LEAST-MEAN-SQUARE ADAPTIVE ARRAY

Fig.2.1 shows a typical adaptive beamformer. It consists of a uniform linear array of 'P' identical sensors receiving 'K' narrowband signals, that arrive at the array from directions  $\theta_0, \theta_1, \dots, \theta_{K-1}$ , and an adaptive processor for automatic weight adjustment. The signals are assumed to be complex analytic. The desired signal is assumed to be at an angle  $\theta_0$  and is of the form

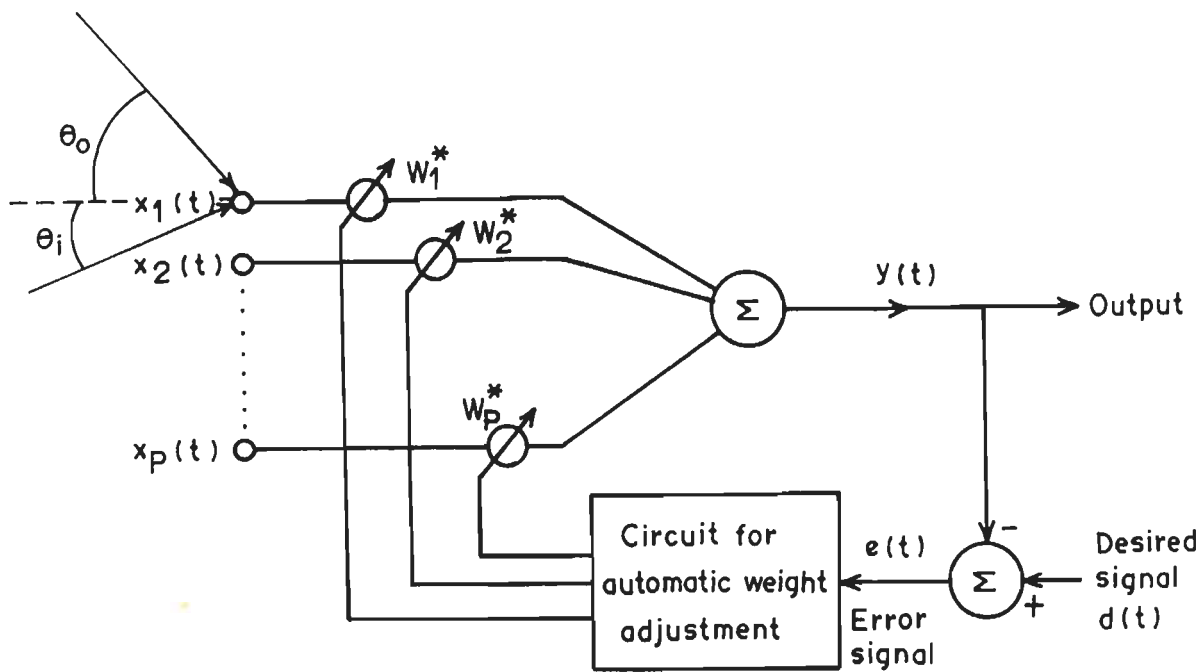


Fig.21.Adaptive Array with Automatic Weight Adjustment.

$$S(t) = S_0 e^{j(\omega_0 t + \phi_0)} \quad \dots(2.1)$$

where  $S_0$ ,  $\omega_0$  and  $\phi_0$ , respectively, denote its amplitude, frequency and phase. The phase  $\phi_0$  is taken to be a uniformly distributed random variable with the probability density function

$$p(\phi_0) = \begin{cases} \frac{1}{2\pi} & 0 \leq \phi_0 \leq 2\pi \\ 0 & \text{elsewhere} \end{cases} \quad \dots(2.2)$$

The remaining (K-1) signals are interfering or jamming signals of the form

$$J_i(t) = S_i e^{j(\omega_i t + \phi_i)}, \quad i = 1, 2, \dots, K-1 \quad \dots(2.3)$$

where  $S_i$ ,  $\omega_i$  and  $\phi_i$  are respectively, the amplitude, frequency and phase of the  $i^{\text{th}}$  interfering signal.

We define a column vector  $\underline{S}$  as

$$\underline{S} = [S(t), \underline{J}(t)]^T \quad \dots(2.4)$$

where,

$$\underline{J}(t) = [j_1(t), j_2(t), \dots, j_{K-1}(t)] \quad \dots(2.5)$$

and the superscript 'T' denotes the transpose operation.

The direction or the steering vector of the  $i^{\text{th}}$  source,  $\underline{a}(\tau_i)$  is given by

$$\underline{a}(\tau_i) = \left[ 1, e^{-j\omega_i \tau_i}, \dots, e^{-j(P-1)\omega_i \tau_i} \right]^T, \quad i = 0, 1, \dots, K-1 \quad \dots(2.6)$$

where  $\tau_i = (d/c)\sin\theta_i$ , 'c' being the propagation velocity of planewaves and 'd' being the interelement spacing. Here, 'd' is



assumed to be less than or equal to half wavelength to avoid spatial aliasing problems.

If we define a matrix  $A$  of steering vectors

$$A = [\underline{a}(\tau_0), \underline{a}(\tau_1), \dots, \underline{a}(\tau_{K-1})] \quad \dots(2.7)$$

and assume white Gaussian noise  $n_i(t)$ , at each element, the received signals at the array can be written as

$$\begin{aligned} \underline{X}(t) &= [x_1(t), x_2(t), \dots, x_p(t)]^T \\ &= A \underline{S} + \underline{n}(t) \\ &= \underline{a}(\tau_0) S(t) + \bar{A} \underline{J}(t) + \underline{n}(t) \end{aligned} \quad \dots(2.8)$$

where  $A$  is a  $P$ -by- $K$  matrix and  $\bar{A}$  is a  $P$ -by- $(K-1)$  matrix. It is assumed here that the signal, interferences and noise are all stationary, zero-mean random processes uncorrelated with each other.

The output of the array,  $Y(t)$ , is the weighted sum

$$Y(t) = \underline{W}^H \underline{X}(t) \quad \dots(2.9)$$

where

$$\underline{W} = [w_1, w_2, \dots, w_p]^T \quad \dots(2.10)$$

is a weight vector determined according to the Least-Mean-Square error criterion and the superscript 'H' denotes the Hermitian transpose.

For discrete systems, the input signals of the array are in discrete time sampled data form and the output of the array is written as

$$Y(n) = \underline{W}^H \underline{X}(n) \quad \dots(2.11)$$

where 'n' denotes the sampling instant.

In order that the adaptation takes place, a "desired signal"  $d(t)$ , when in time continuous form, or  $d(n)$ , when sampled, must be applied to the beamformer. The difference between the desired response and the array output forms the error signal,  $e(n)$ , given by

$$e(n) = d(n) - \underline{W}^H \underline{X}(n) \quad \dots(2.12)$$

This signal is used as a control signal for the weight adjustment circuit. The purpose of adaptation is to find a set of weights that permit the output of the adaptive array, at each instant of time, to be as close as possible to the desired signal. There are 'N' such equations corresponding to 'N' instants of time and 'P' unknown weight values which form components of  $\underline{W}$ .

When 'N' is very large compared to 'P', the minimization of the sum of squares of errors gives the required solution. That is, a set of weights  $\underline{W}$  is determined so as to minimize  $\sum_{n=1}^N e(n)e^*(n)$ . Since the signals are assumed to be stationary, the quantity of interest is the expected value of the mean squared error

$$E \left[ e(n) e^*(n) \right] = \bar{\epsilon}^2 \quad \dots(2.13)$$

where the asterisk denotes the complex conjugation.

The mean-square-error can be calculated by substituting eqn.(2.12) in eqn.(2.13) which yields

$$\bar{\epsilon}^2 = E \left[ d(n) d^*(n) \right] + \underline{W}^H \Phi \underline{W} - \underline{W}^H \underline{\theta} - \underline{\theta}^H \underline{W} \quad \dots(2.14)$$

where  $\Phi$  and  $\underline{\theta}$  are, respectively, the correlation matrix and the cross correlation vector which are given by

$$\Phi = E \begin{bmatrix} x_1(n)x_1^*(n) & x_1(n)x_2^*(n) & \dots & x_1(n)x_P^*(n) \\ x_2(n)x_1^*(n) & x_2(n)x_2^*(n) & \dots & x_2(n)x_P^*(n) \\ \vdots & \vdots & \ddots & \vdots \\ x_P(n)x_1^*(n) & x_P(n)x_2^*(n) & \dots & x_P(n)x_P^*(n) \end{bmatrix} \quad \dots (2.15)$$

$$\underline{\theta} = E \begin{bmatrix} x_1(n)d^*(n) \\ x_2(n)d^*(n) \\ \vdots \\ x_P(n)d^*(n) \end{bmatrix} \quad \dots (2.16)$$

It may be seen from eqn.(2.14), that  $\varepsilon^{-2}$  is a quadratic function of the weight vector  $\underline{W}$ . The weight vector yielding minimum of  $\varepsilon^{-2}$ , which we denote by  $\underline{W}_{opt}$ , may be found by setting

$$\nabla_{\underline{W}} \left\{ \varepsilon^{-2} \right\} = 0 \quad \dots (2.17)$$

where  $\nabla_{\underline{W}}$  denotes the gradient with respect to  $\underline{W}$  (See Appendix-A).

Since,

$$\nabla_{\underline{W}} \left\{ \varepsilon^{-2} \right\} = -2 \underline{\theta} + 2 \Phi \underline{W} \quad \dots (2.18)$$

we find

$$\Phi \underline{W}_{opt} = \underline{\theta} \quad \dots (2.19)$$

or

$$\underline{w}_{\text{opt}} = \Phi^{-1} \underline{\theta} \quad \dots (2.20)$$

where  $\Phi$  is assumed to be nonsingular, so that its inverse exists. If this  $\underline{w}_{\text{opt}}$  is used in eqn.(2.11) to compute the output of the array, the beamformer is called an 'optimum beamformer'. Eqn.(2.20) is the Wiener-Hopf equation in matrix form and is, consequently, referred to as the optimum Wiener solution.

If we use  $d(n) = S(n)$ , it then follows from equations (2.1) and (2.8) that

$$\underline{\theta} = E[\underline{X}(n) d^*(n)] = S_o^2 \underline{a}(\tau_o) \quad \dots (2.21)$$

Therefore,

$$\underline{w}_{\text{opt}} = S_o^2 \Phi^{-1} \underline{a}(\tau_o) \quad \dots (2.22)$$

Since the desired signal, interferences and noise are all assumed to be uncorrelated with each other, the correlation matrix,  $\Phi$  in the above equation can be expressed in factored form as [41]

$$\Phi = S_o^2 \underline{a}(\tau_o) \underline{a}^H(\tau_o) + \Phi_{jj} + \sigma^2 I \quad \dots (2.23)$$

where  $I$  is the identity matrix.

In eqn.(2.23),  $S_o^2 \underline{a}(\tau_o) \underline{a}^H(\tau_o)$  represents the correlation matrix of the desired signal,  $\Phi_{jj}$  is the correlation matrix for the interferences and  $\sigma^2 I$  is the noise component.

We can write

$$\Phi_{jj} + \sigma^2 I = \Phi_{nn} \quad \dots (2.24)$$

so that

$$\Phi = S_o^2 \underline{a}(\tau_o) \underline{a}^H(\tau_o) + \Phi_{nn} \quad \dots(2.25)$$

Using matrix inversion lemma[22], eqn.(2.22) can be simplified as

$$\underline{W}_{opt} = \left[ \frac{S_o^2}{1 + S_o^2 \underline{a}^H(\tau_o) \Phi_{nn}^{-1} \underline{a}(\tau_o)} \right] \Phi_{nn}^{-1} \underline{a}(\tau_o). \quad \dots(2.26)$$

In the above equation, the term inside the square brackets is a scalar and, therefore,  $\underline{W}_{opt}$  can be written as

$$\underline{W}_{opt} = \beta \Phi_{nn}^{-1} \underline{a}(\tau_o) \quad \dots(2.27)$$

where  $\beta$  is a scalar constant.

Besides the Least-Mean-Square error criterion discussed above, other criteria can also be used for optimizing the adaptive array. All these criteria, however, lead to the same expression for the optimum weight vector (eqn.(2.27)) ; only the value of the scalar constant  $\beta$  changes. For example, Howell-Applebaum array is also an optimum array that maximizes the signal to interference plus noise ratio (SINR). In the case of Frost array, which will be discussed subsequently, the optimal weight vector is obtained by minimizing the array output power, subject to unity gain constraint in the look direction.

The ability of the array to place nulls in the direction of interferences can be explained as follows. If we express the correlation matrix  $\Phi_{jj}$  in modal representation [57], then

$$\Phi_{JJ} = \bar{A} E \left[ \underline{J}(n) \underline{J}^H(n) \right] \bar{A}^H = \sum_{i=1}^{K-1} \lambda_i \underline{e}_i \underline{e}_i^H \quad \dots (2.28)$$

Where  $\lambda_i$  and  $\underline{e}_i$ , respectively, denote the nonzero eigenvalues and corresponding eigenvectors of the P-by-P matrix  $\Phi_{JJ}$ . The matrix will have a rank (K-1) because  $\bar{A}$  has full rank. Also  $E[\underline{J}(n)\underline{J}^H(n)]$  has rank (K-1), it being the Covariance matrix of (K-1) noncoherent interferences. If we assume that the noise intensity is very small compared to the interference signals  $j_i(t)$ , so that

$$\lambda_i \gg \sigma^2 \quad \text{and} \quad (1/\sigma^2) \gg \frac{1}{\lambda_i + \sigma^2} \quad \text{for } i = 1, 2, \dots, K-1.$$

Then we can write

$$\begin{aligned} \underline{w}_{\text{opt}} &= \beta \Phi_{nn}^{-1} \underline{a}(\tau_0) \\ &= \beta \left\{ \sum_{i=1}^{K-1} \frac{1}{\lambda_i + \sigma^2} \underline{e}_i \underline{e}_i^H + \sum_{i=K}^P (1/\sigma^2) \underline{e}_i \underline{e}_i^H \right\} \underline{a}(\tau_0) \\ &\approx (\beta/\sigma^2) \sum_{i=K}^P \rho_i \underline{e}_i \end{aligned} \quad \dots (2.29)$$

Where  $\rho_i = \underline{e}_i^H \underline{a}(\tau_0)$ . By construction, the direction vectors  $\{\underline{a}(\tau_1), \dots, \underline{a}(\tau_{K-1})\}$  of the interference signals, which are columns of matrix  $\bar{A}$ , lie in the span of first (K-1) eigenvectors  $\{\underline{e}_1, \dots, \underline{e}_{K-1}\}$  and are, therefore, orthogonal to the remaining eigenvectors  $\{\underline{e}_K, \dots, \underline{e}_P\}$ .

Thus we have

$$\underline{w}_{\text{opt}}^H \underline{a}(\tau_l) \approx \frac{\beta}{\sigma^2} \sum_{i=K}^P \rho_i^* \underline{e}_i^H \underline{a}(\tau_l) = 0, \quad l = 1, \dots, K \quad \dots (2.30)$$

Therefore, the beam pattern will have deep nulls in the direction of interferences.

It can be seen from eqn.(2.20), that the computation of the optimum weight vector,  $\underline{W}_{opt}$  requires the knowledge of two quantities : (i) the correlation matrix  $\Phi$  and (ii) the cross correlation vector  $\underline{\theta}$ . The correlation matrix  $\Phi$  plays a key role in the statistical analysis and design of optimum beamformers. Hence, it is necessary to know its properties. The correlation matrix  $\Phi$  is Hermitian and Toeplitz in structure and is always nonnegative definite. Also, all the eigenvalues of the matrix  $\Phi$  are real and nonnegative and the eigenvectors corresponding to distinct eigenvalues are linearly independent with each other [22].

Although the solution of eqn.(2.20) is straight forward, it presents serious computational difficulties when the number of elements 'P' is large and the data rate is high. Not only it is necessary to invert a P-by-P matrix, but  $P(P+1)/2$  autocorrelation and cross correlation measurements are required to obtain the elements of  $\Phi$ . Also, no perfect solution of eqn.(2.20) is possible in practice, because larger number of samples would be required to estimate the elements of the correlation matrix accurately [65].

An alternative method for minimizing the mean-square-error is the Least-Mean-Square[LMS] algorithm. This method does not require explicit measurements of correlation functions or matrix inversion and accomplishes the minimization by gradient search technique.

### 2.1-1 The Least-Mean-Square[LMS] Algorithm

The LMS algorithm is based on the method of steepest descent [42]. According to this method, the value of the weight vector at time (n+1) is computed using the simple recursive relation

$$\underline{W}(n+1) = \underline{W}(n) + \frac{1}{2} \mu \begin{bmatrix} \hat{\nabla}(n) \\ -\hat{\nabla}(n) \end{bmatrix} \quad \dots (2.31)$$

where  $\underline{W}(n)$  and  $\underline{W}(n+1)$ , respectively, represent the weight vector before and after adaptation,  $\mu$  is a scalar constant controlling the rate of convergence and  $\hat{\nabla}(n)$  is the estimate of the gradient vector.

In the LMS algorithm, the estimate of the gradient vector,  $\hat{\nabla}(n)$ , where the symbol ' $\hat{\phantom{x}}$ ' denotes the estimate, is developed by substituting the instantaneous estimates of the correlation matrix,  $\Phi$  and the cross correlation vector  $\underline{\theta}$ , in eqn.(2.18). These instantaneous estimates are based on sample values of the input signal vector and the desired response which are defined as

$$\Phi(n) = \underline{X}(n) \underline{X}^H(n) \quad \dots (2.32)$$

$$\hat{\underline{\theta}}(n) = \underline{X}(n) d^*(n) \quad \dots (2.33)$$

Correspondingly, the instantaneous estimate of the gradient vector is given by

$$\hat{\nabla}(n) = -2\underline{X}(n) d^*(n) + 2\underline{X}(n) \underline{X}^H(n) \underline{W}(n). \quad \dots (2.34)$$

Substituting  $\hat{\nabla}(n)$  in eqn.(2.31), we get

$$\underline{W}(n+1) = \underline{W}(n) + \mu \underline{X}(n) \left[ d^*(n) - \underline{X}^H(n) \underline{W}(n) \right] \quad \dots (2.35)$$

Equivalently, this result may be written in the form of a



pair of equations given by

$$e(n) = d(n) - \underline{W}^H(n) \underline{X}(n) \quad \dots (2.36)$$

$$\underline{W}(n+1) = \underline{W}(n) + \mu \underline{X}(n) e^*(n) \quad \dots (2.37)$$

Eqn.(2.36) defines the estimation error  $e(n)$ , the computation of which is based on the current estimate of the weight vector  $\underline{W}(n)$ . The iterative procedure is started with an initial guess  $\underline{W}(0)$  for which a convenient choice is the null vector. The scalar constant  $\mu$ , controls the rate of convergence and the stability of algorithm. For convergence to occur,  $\mu$  should be so chosen that,  $0 < \mu < 2/\lambda_{\max}$  where  $\lambda_{\max}$  is the largest eigenvalue of the correlation matrix  $\Phi$ .

The main virtue of LMS algorithm is its simplicity, since it requires only 3P arithmetic operations per time sample. However, it needs a large number of data samples to guarantee convergence. Also, it cannot handle a scenario of multiple jammers with different powers [70], and is sensitive to signal statistics. The algorithm is summarized in Table 2.1.

### 2.1-2 Sample Results

The following numerical example compares the performance of the LMS beamformer with that of the optimum beamformer.

In this example, a uniform linear array of six isotropic elements has been assumed. The signal environment consists of a desired signal and four noncoherent interferences. The arrival angles and signal strengths of the interferences are given in Table-2.2.

**Table 2.1**  
**Summary of the LMS Algorithm**

Input definitions

$$w_i(0) = 0, \quad i = 1, 2, \dots, P$$

$$0 < \mu < 2/\lambda_{\max}$$

Algorithm	Complexity
For n = 1, 2, ..., N do	
For i = 1 to P do	
$e(n) = d(n) - w_i^*(n) x_i(n)$	P (T 2.11)
$w_i(n+1) = w_i(n) + \mu x_i(n) e^*(n)$	2P (T 2.12)
Total	3P

**Table 2.2**  
**Interference Parameters for Narrowband Beamformers**

parameter	Interference 1	Interference 2	Interference 3	Interference 4
$S_i$	10.0	10.0	10.0	10.0
$\theta_i$	$30^\circ$	$-30^\circ$	$60^\circ$	$-60^\circ$
$\omega_i$	1.1	0.9	1.2	0.8

In the case of the optimum beamformer, the array correlation matrix,  $\Phi$ , was computed using 1000 snapshots of data and then a noise power  $\sigma^2 = 0.001$  was added to its diagonal elements. Gaussian elimination technique was used to obtain its inverse. The eigenvalues of the correlation matrix are 783.76, 452.96, 596.79, 564.01, 0.05, 0.001 and as expected, these are all real and nonnegative.

In the case of LMS array, the weight vector was initialized

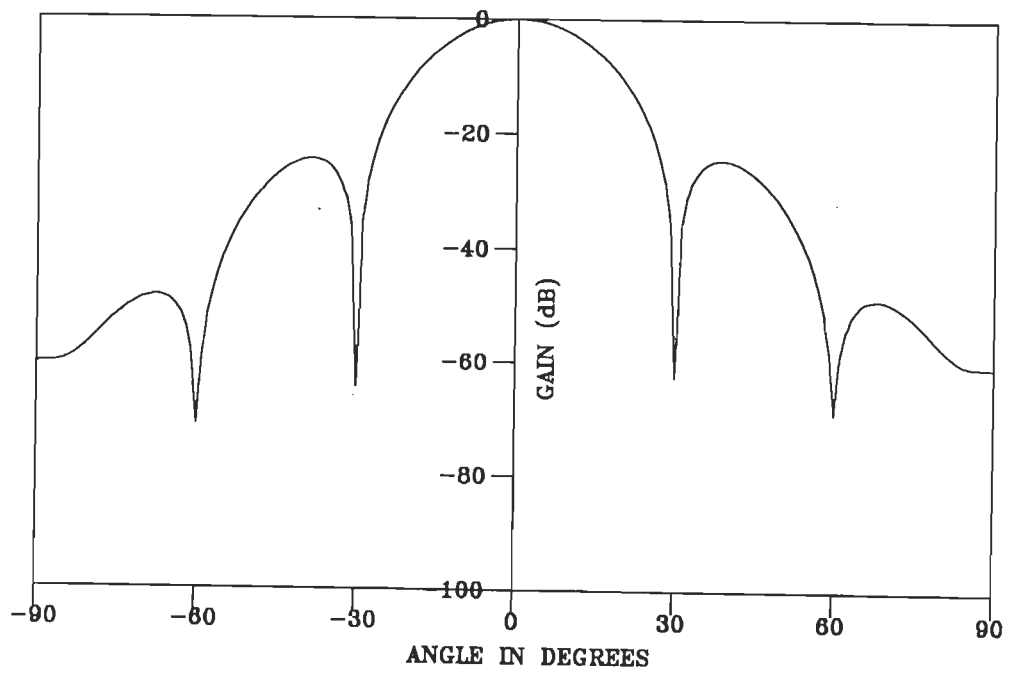
with all its components set to zero. Corresponding to the largest eigenvalue, the permissible upper bound for  $\mu$  was found to be 0.0255; a value of 0.0001 was selected for the present problem. The internal antenna noise was simulated by generating independent, normally distributed, random numbers using Box-Muller's transformation on uniformly distributed numbers in the range (0,1) [55]. Since, the LMS algorithm takes a large number of iterations to converge, 1000 iterations were used.

The performance of the array was evaluated by computing the array pattern. A unit amplitude test signal was assumed to be incident upon the array from an angle  $\theta$ . The voltage pattern was then obtained by computing the array output  $Y$  (eqn.2.11) as a function of  $\theta$ .

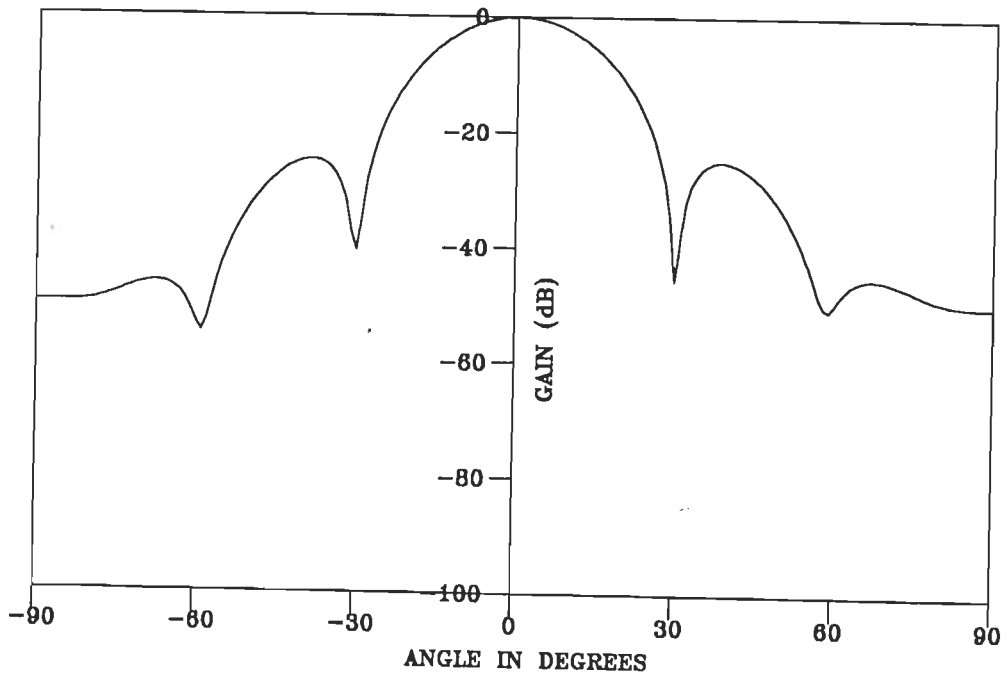
Fig.2.2 shows the computed patterns for the optimum beamformer and the LMS array. As can be seen from the figure, both techniques succeed in placing nulls in the direction of interferences. However, the nulls placed by the LMS array are relatively shallow as compared with those produced by the optimum beamformer. This is because the LMS algorithm is based on a statistically defined error criterion and uses instantaneous estimates for the correlation matrix  $\Phi$  and the cross correlation vector  $\underline{\theta}$ . Moreover, the LMS algorithm is based on the steepest descent technique which is an approximate technique.

To illustrate the numerical properties and the convergence characteristics of the LMS array, the output residual power as a function of the number of adaptation samples is shown in Fig.2.3.

Here, the output residual power at the time instant 'n' has been defined as



(a) OPTIMUM BEAMFORMER



(b) LMS BEAMFORMER

FIG.2.2 VOLTAGE PATTERNS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ$ ,  $-30^\circ$ ,  $60^\circ$  &  $-60^\circ$ .

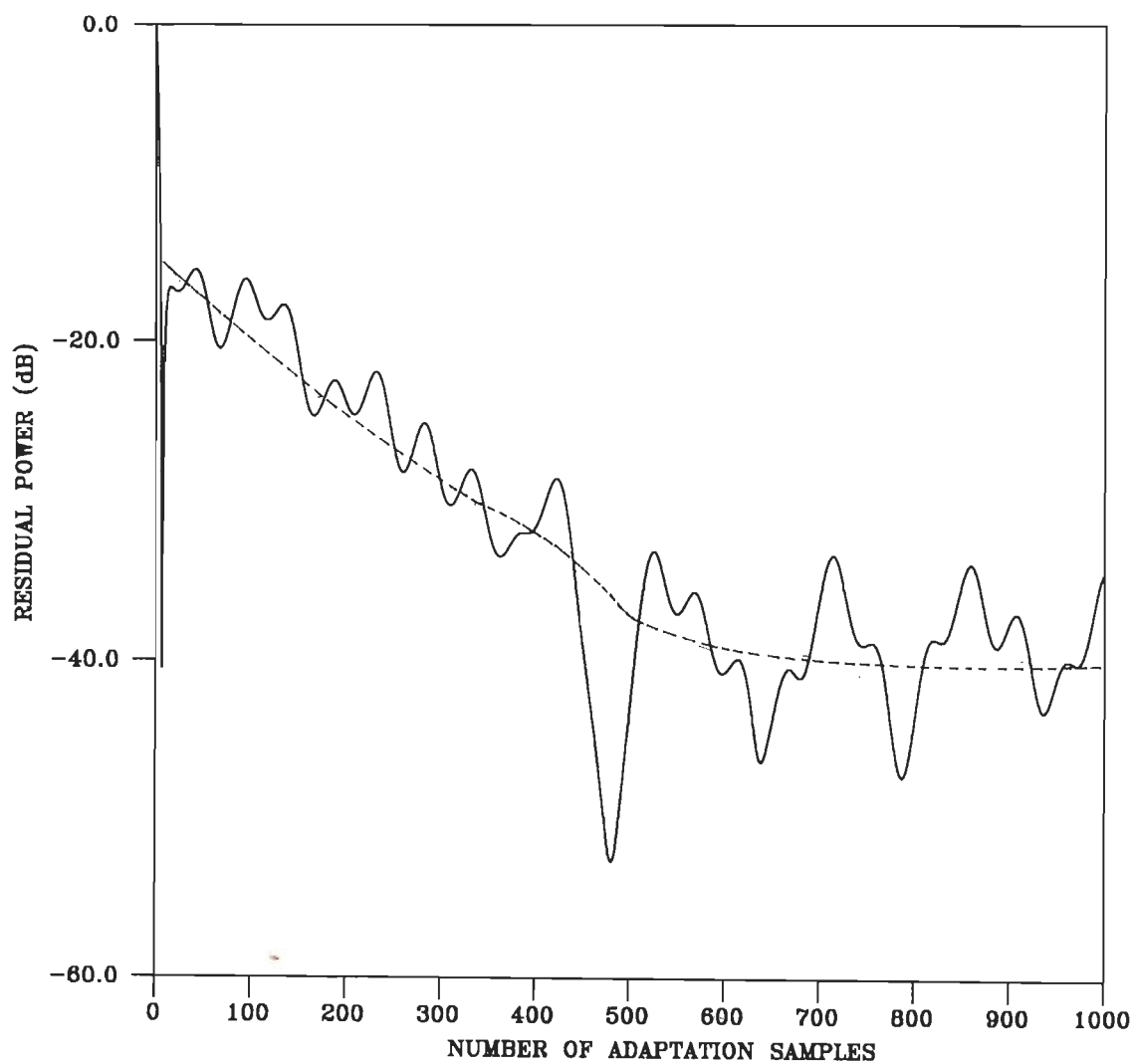


Fig. 2.3 CONVERGENCE CHARACTERISTICS OF LMS ARRAY

$$P_r(n) = |e(n)|^2 = |Y(n) - d(n)|^2 \quad \dots(2.38)$$

As can be seen from the figure, the LMS array exhibits poor convergence. The residual power gradually reduces from about - 17dB to a steady state value of about -40dB after 500 iterations.

## 2.2 THE BROADBAND ADAPTIVE ARRAY

In sec.2.1, it has been assumed that the signals are narrowband. The pattern nulls are placed in the direction of interferences by a set of weights denoted by the weight vector  $\underline{W}$ . In the event the desired signal and the interference signals are "broadband", that is, signals have a frequency content encompassing a significant bandwidth, the performance of the array will be degraded. This is because the interelement phase delay depends on frequency which makes the selection of appropriate weight vector for one frequency  $\omega_1$ , not appropriate for a different frequency  $\omega_2$  and the array pattern null shifts as the value of wavelength of the desired signal,  $\lambda_0$ , changes. This fact leads to the conclusion that different weight vectors are required at different frequencies, if an array null is to be maintained in some direction for the frequencies over a band of interest [41,8].

A simple and effective way of obtaining different weight vectors at a number of frequencies over a band of interest is to use a tapped delay line network coupled with weights, as shown in Fig.2.4. The tapped delay line network allows the array to insert a variable amount of phase delay behind each element. This delay can be used by the processor to compensate for the interelement propagation delay. If

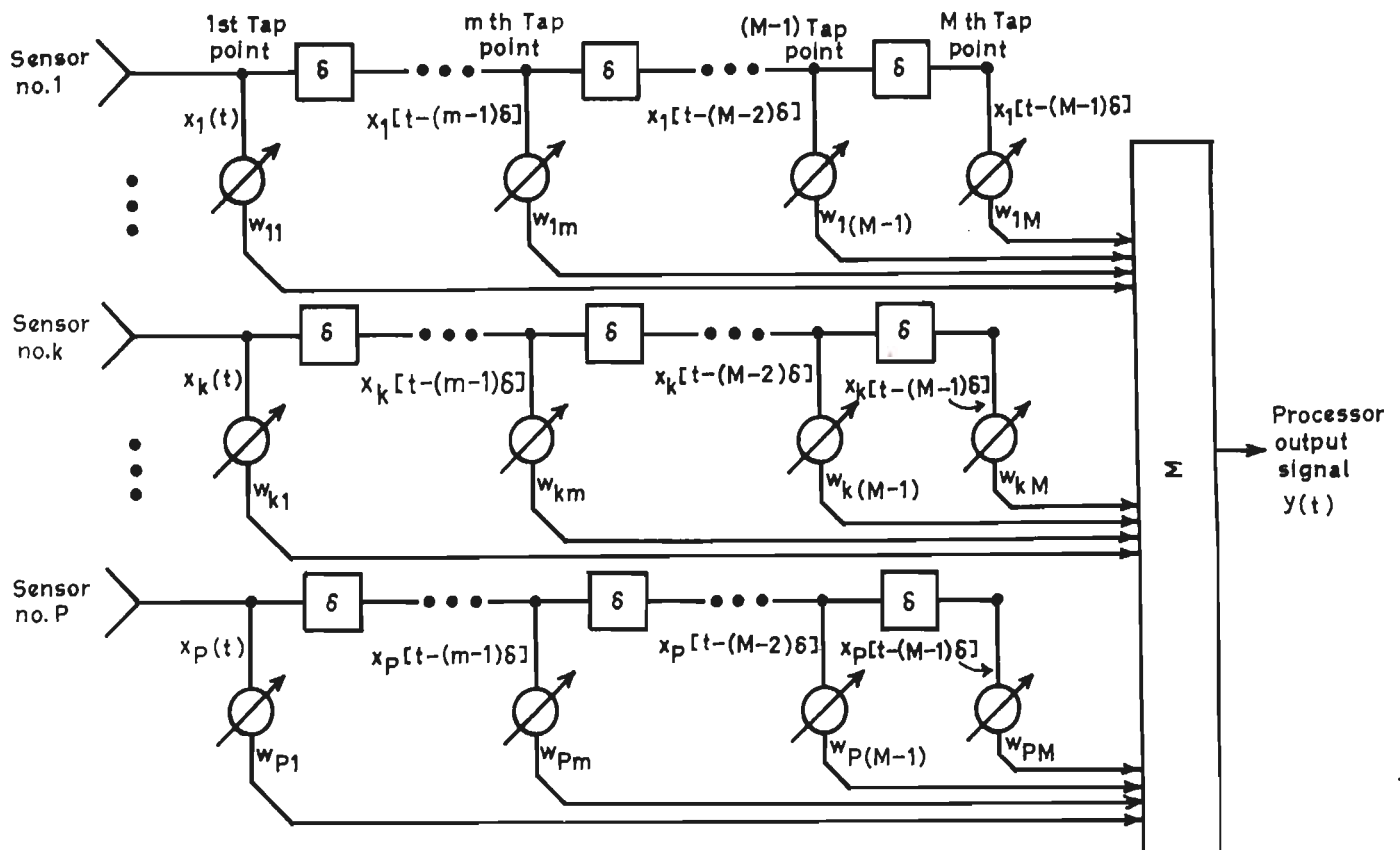


Fig.2.4 .Tapped-delay line multichannel processor for wideband signal processing .

the tap spacing is sufficiently close, the network approximates an ideal filter which allows complete control of gain and phase of each frequency in the pass band.

The basic configuration of a broadband beamformer is shown in Fig.2.4. Behind each of the 'P' sensors, a tapped delay line is connected which consists of 'M' tap points, (M-1) time delays of  $\delta$  seconds each and 'M' complex weights. If  $x_1(t), \dots, x_p(t)$  denote the signals at the array input, we may define a complex vector  $\underline{X}_1(t)$ , such that

$$\underline{X}_1^T(t) = [x_1(t), x_2(t), \dots, x_p(t)] \quad \dots(2.39)$$

In all the delay lines, signals appearing at the second tap point are merely a time delayed version of the signals appearing at the first tap point. So a complex vector  $\underline{X}_2(t)$  may be defined as

$$\underline{X}_2^T(t) = [x_1(t-\delta), x_2(t-\delta), \dots, x_p(t-\delta)] \quad \dots(2.40)$$

Continuing in the same manner for all M tap points, the complete signal vector for the entire array becomes

$$\underline{X}^T(t) = [\underline{X}_1^T(t), \underline{X}_2^T(t), \dots, \underline{X}_M^T(t)] \quad \dots(2.41)$$

The reason for expressing the signal vector in this form is that this construction leads to a block Toeplitz form for the correlation matrix,  $\Phi$ , of the input signals.

Similarly, the weight vector for the entire array can be written as



$$\underline{W} = \left[ \underline{w}_1^T, \underline{w}_2^T, \dots, \underline{w}_M^T \right] \quad \dots (2.42)$$

where

$$\underline{w}_1^T = \left[ w_{11}, w_{21}, w_{31}, \dots, w_{P1} \right] \text{ .etc,} \quad \dots (2.43)$$

As a consequence of signal and weight vector definitions introduced above, the array output can be written as

$$\begin{aligned} Y(t) &= \sum_{m=1}^M \left[ \underline{w}_m \right]^H \underline{X}_m(t) \\ &= \underline{W}^H \underline{X}(t). \end{aligned} \quad \dots (2.44)$$

which is exactly of the same form as in eqn.(2.11).

### 2.2-1 Adaptive Array with Constraints

The addition of constraints to the LMS adaptive array was first described by Frost [13]. This algorithm called the "constrained LMS" algorithm, is a simple stochastic gradient algorithm in which the direction of the arrival of the desired signal and a frequency band of interest are defined a priori. A major advantage of this algorithm is its self-correcting feature. This feature permits it to operate for arbitrarily long periods of time in a digital computer implementation, without deviating from its constraints, because of cumulative round off and truncation errors [13].

Assuming that, the signals are in discrete time sampled form, eqn.(2.44) can be written as

$$Y(n) = \underline{W}^H \underline{X}(n) \quad \dots (2.45)$$

The expected output power of the array is

$$\begin{aligned} E[Y(n) Y^*(n)] &= E\left[\underline{W}^H \underline{X}(n) \underline{X}^H(n) \underline{W}(n)\right] \\ &= \underline{W}^H \Phi \underline{W} \end{aligned} \quad \dots (2.46)$$

The constraint that the weights on the  $m^{\text{th}}$  vertical column of tap points sum to a chosen number  $l_m$  is expressed by the requirement

$$\underline{C}_m^T \underline{W} = l_m, \quad m = 1, 2, \dots, M \quad \dots (2.47)$$

where  $(M \times 1)$  vector  $\underline{C}_m$  has the form

$$\underline{C}_m^T = \left[ 0, \dots, 0, \underbrace{0, \dots, 0, 1, \dots, 1, 0, \dots, 0}_{m^{\text{th}} \text{ group of } P \text{ elements}} \right] \quad \dots (2.48)$$

Constraining the weight vector to satisfy the  $M$  equations in (2.47),  $\underline{W}$  is restricted to a  $(PM-M)$  dimensional plane.

Now, we define the constraint matrix  $C$  as

$$C = \left[ \underline{C}_1, \dots, \underline{C}_m, \dots, \underline{C}_M \right] \quad \dots (2.49)$$

which is of dimension  $(PM \times M)$ .

We also define  $\underline{W}$  as the  $M$ -dimensional vector of weights of

the look-direction equivalent tapped delay line (Fig.2.5), as

$$\underline{\$} = \left[ 1_1, \dots, 1_m, \dots, 1_M \right] \quad \dots(2.50)$$

The constraints in eqn.(2.47) can now be written as

$$C^T \underline{W} = \underline{\$} \quad \dots(2.51)$$

As the look-direction frequency response is fixed by M constraints, minimization of the nonlook-direction noise power is the same as the minimization of total output power. Thus, the problem of finding the optimum weight vector  $\underline{W}_{opt}$  can be summarized as

$$\begin{aligned} & \underset{\underline{W}}{\text{minimize}} \quad \left[ \underline{W}^H \Phi \underline{W} \right] \\ & \text{Subject to } C^T \underline{W} = \underline{\$} \end{aligned} \quad \dots(2.52)$$

This is called the constrained LMS problem.

The optimum weight vector  $\underline{W}_{opt}$  is found by the method of Lagrange multipliers and is given by [13]

$$\underline{W}_{opt} = \Phi^{-1} \underline{C} \left[ \underline{C}^H \Phi^{-1} \underline{C} \right]^{-1} \underline{\$} \quad \dots(2.53)$$

The above equation is not easy to interpret because of the complicated way the constraint matrix C appears in the equation. However, a simple example will make it clear. Suppose it is desired to minimize the interference and noise, while maintaining the array response in the desired signal direction equal to unity. The

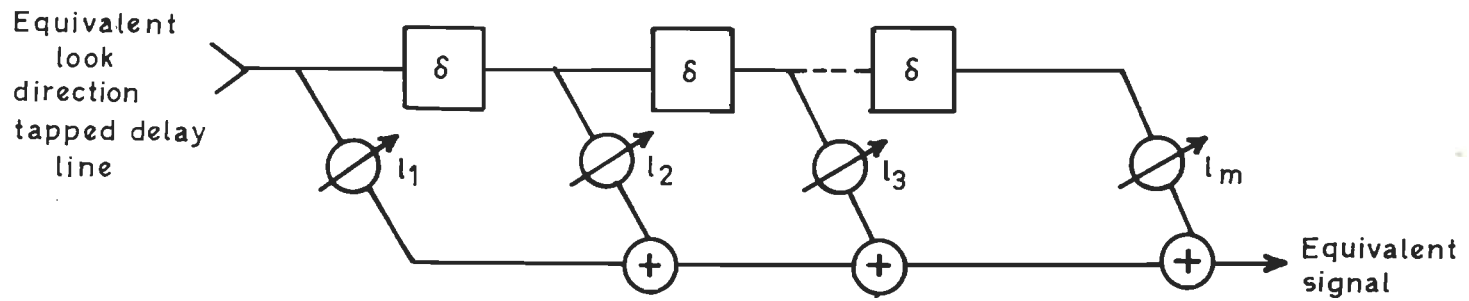


Fig.2.5. Broadband Antenna Array Equivalent Processor.

constraint is then

$$\underline{a}^H(\tau_o) \underline{W} = 1. \quad \dots(2.54)$$

Therefore,

$$\underline{C} = \underline{a}(\tau_o) \text{ and } \underline{\$} = 1 \quad \dots(2.55)$$

Substituting for  $\underline{C}$  and  $\underline{\$}$  in eqn.(2.53), we get

$$\underline{W}_{opt} = \Phi^{-1} \underline{a}(\tau_o) \left[ \underline{a}^H(\tau_o) \Phi^{-1} \underline{a}(\tau_o) \right]^{-1} \quad \dots(2.56)$$

Comparing the above equation with eqn.(2.27), We have

$$\beta = \left[ \underline{a}^H(\tau_o) \Phi^{-1} \underline{a}(\tau_o) \right]^{-1} \quad \dots(2.57)$$

Since  $\beta$  is a scalar, premultiplying the eqn.(2.56) by  $\underline{a}^H(\tau_o)$ , we have

$$\underline{a}^H(\tau_o) \underline{W}_{opt} = \frac{\underline{a}^H(\tau_o) \Phi^{-1} \underline{a}(\tau_o)}{\underline{a}^H(\tau_o) \Phi^{-1} \underline{a}(\tau_o)} = 1 \quad \dots(2.58)$$

So the desired constraint relation is automatically satisfied.

There are also other ways of using the constraints. For example, a constraint could also be used to produce a fixed null in the array pattern in a certain direction, if desired. Such a constrained null might be useful if a source of interference exists at some known angle [8].

## 2.2-2 Broadband Signal Simulation

When statistically independent white noise samples are filtered, the shape of the resulting power spectrum, or "colour", of the resulting sequence is determined by the transfer function of the filter [55]. The digital filtering algorithm, in general, produces samples that are correlated, resulting in a nonwhite spectrum.

For the simulation of a random function with a predetermined power spectrum, the input power density,  $\bar{S}_{xx}$ , is made equal to unity at frequencies below one-half of the sampling rate. Therefore, the variance of the input samples is

$$\sigma_x^2 = \frac{1}{2\pi} \int_{-\pi/2}^{\pi/2} \bar{S}_{xx}(j\omega) d\omega = \frac{1}{T} \quad \dots (2.59)$$

where  $T$  is the sampling interval. Thus, if  $r_n$  is the  $n^{\text{th}}$  independent uniform random sample between 0 and 1 generated by a computer library routine, then

$$d_n = \sqrt{\frac{12}{T} \left[ r_n - \frac{1}{2} \right]} \quad \dots (2.60)$$

would be the appropriate white noise sample.

In a general purpose computer program, the process begins with the generation of independent random samples  $r_n$  in the interval (0,1). Each sample  $r_n$  is then converted to  $d_n$ , a sample of the white uniform sequence using eqn.(2.60). It is then fed through a bandpass filter to produce the desired sample set,  $g_m$ .

As an example, we consider the generation of the random

sequence of a broadband signal with unit power density, having all its power concentrated between 0.75Hz and 1.25Hz. We use a sampling interval of  $T = 0.05$  sec, which is well below one half the sampling rate of the centre frequency 1Hz of the desired broadband signal.

The situation calls for a bandpass filter with a pass band 0.75Hz-1.25Hz. We have used the Butterworth bandpass filter routine SPFIL2 [55], with five sections in cascade, to simulate the filter on the computer. The details of the sequence generation process are illustrated in Fig.2.6. The program causes a white uniform sequence to be generated and then filtered to produce  $g_m$ . The routine is called with frequencies 0.0325 and 0.0625. The filter has been assumed to have 10 two-pole sections.

The entire sequence  $g_m$ , a broadband signal of bandwidth 0.75-1.25Hz, is shown in Fig.2.7.

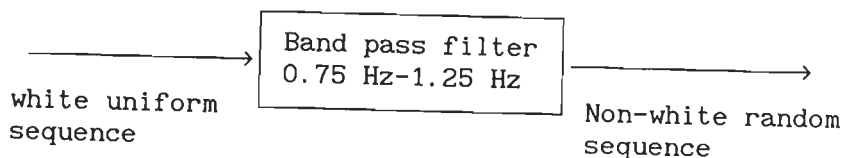


Fig. 2.6 Generation of a broadband signal

### 2.2-3 Sample Results

The following example simulated on the computer illustrates the ability of the Frost array to null broadband interferences.

We consider a uniform linear array of six isotropic elements. The output of each sensor is processed using a tapped delay line containing four multiplying weights and three ideal time delays of  $(1/4\omega_0)$  seconds each. The signal environment consists of one

desired signal and two interferences that are broadband in nature. The arrival angles and signal strengths are given in Table 2.3.

Table 2.3  
Interference Parameters for the Frost Array

parameter	Interference 1	Interference 2
$S_i$	0.01	0.01
$\theta_i$	$50^\circ$	$-50^\circ$
$\omega_i$	$1.0 \omega_o$	$1.2 \omega_o$
$\Delta_i$	$\pm 0.4$	$\pm 0.5$

A constraint of unity gain in the look-direction has been used. The desired signal strength has been assumed to be equal to 0.001. Optimum weights have been computed by forming the correlation matrix,  $\Phi$ , from 1000 snapshots of input data. A white noise component of variance 0.0001 has been added to the diagonal elements of  $\Phi$ .

The array pattern, shown in Fig.2.8, illustrates the ability of the Frost array to place nulls in the direction of the broadband interferences. It may be noted here that a 6-element array with a half wavelength spacing has its natural nulls located at  $\pm 20^\circ$  and  $\pm 40^\circ$ . As is seen in Fig.2.8, the adaptation process leaves the nulls at  $\pm 20^\circ$  undisturbed while the other two nulls are steered to  $\pm 50^\circ$ , that is, in the direction of interferences.

### 2.3 BEAMFORMING IN THE PRESENCE OF COHERENT SIGNALS

A key assumption in the discussion, so far, is that the desired signal, interference and noise are all zero-mean processes



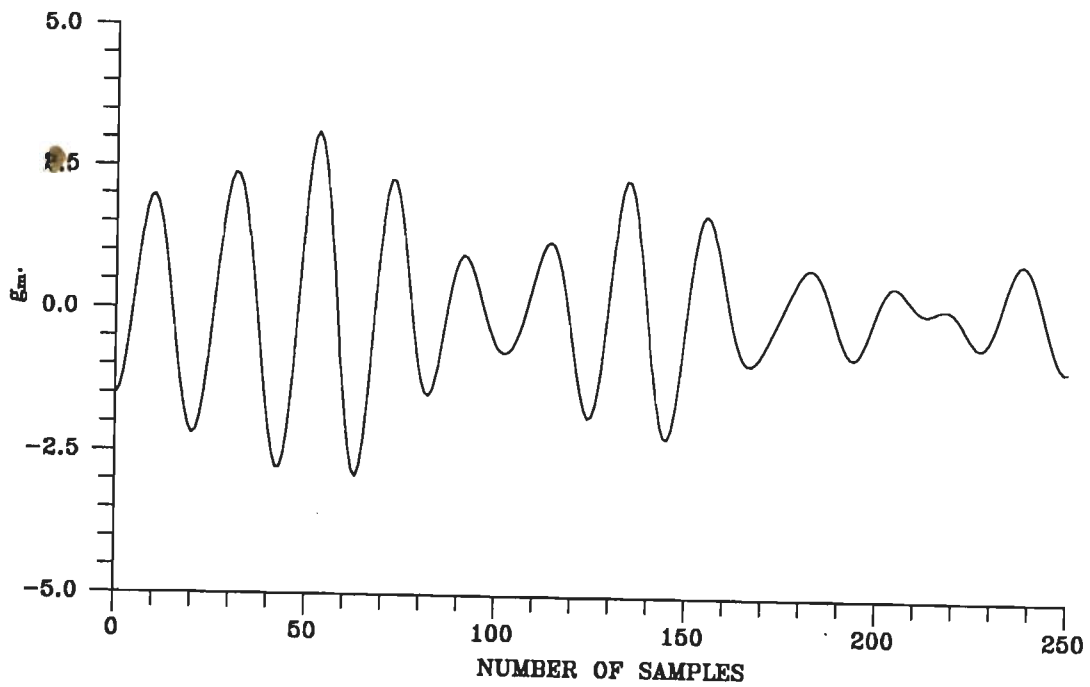


FIG.2.7 DESIRED BROADBAND SIGNAL

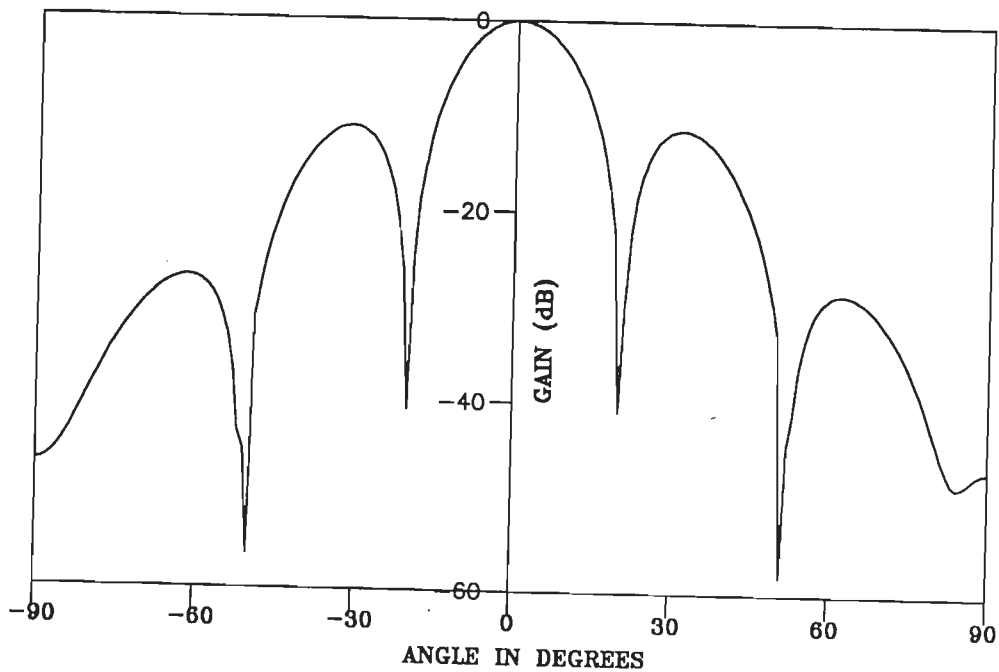


FIG.2.8 VOLTAGE PATTERN OF FROST BROADBAND ARRAY WITH INTERFERENCES ARRIVING AT  $50^\circ$  AND  $-50^\circ$

uncorrelated with each other. The assumption is not valid in practical situations where due to the presence of multipath propagation or due to smart jammers, even fully coherent interferences can exist.

If the signals impinging on the array are coherent, then the vector  $\underline{A} \underline{S}$  in eqn.(2.8) can be written as [57]

$$\begin{aligned} \underline{A} \underline{S} &= \underline{a}(\tau_0) s(t) + \sum_{i=1}^{K-1} \underline{a}(\tau_i) j_i(t) \\ &= \left[ \underline{a}(\tau_0) + \gamma_1 \underline{a}(\tau_1) + \dots + \gamma_{K-1} \underline{a}(\tau_{K-1}) \right] s(t) \\ &= \underline{b} s(t) \end{aligned} \quad \dots (2.61)$$

where

$$\underline{b} = \left[ \underline{a}(\tau_0) + \gamma_1 \underline{a}(\tau_1) + \dots + \gamma_{K-1} \underline{a}(\tau_{K-1}) \right] \quad \dots (2.62)$$

$\gamma_i$  being the fixed constants given by

$$\gamma_i = \left[ \frac{S_i}{S_0} \right] e^{j(\phi_i - \phi_0)}, \quad i = 1, \dots, K-1 \quad \dots (2.63)$$

It may be noted that since  $\underline{b}$  is a linear combination of all the steering vectors, it yields another steering vector.

Substituting for  $\underline{A} \underline{S}$  in eqn.(2.8), we get the received signals at the array as

$$\underline{X}(t) = \underline{b} s(t) + \underline{n}(t) \quad \dots (2.64)$$

In this case, the covariance matrix  $\underline{A} E[\underline{S}\underline{S}^H] \underline{A}^H$  will have rank one. Therefore, the correlation matrix  $\Phi$  will have one non-zero eigenvalue  $\lambda_1 + \sigma^2$  and  $(P-1)$  eigenvalues equal to  $\sigma^2$  [57].

In this situation, the weight vector in eqn.(2.27) reduces to

$$\begin{aligned}
 \underline{w}_{\text{opt}} &= S_o^2 \Phi^{-1} \underline{a}(\tau_o) \\
 &= S_o^2 \left[ \frac{1}{\lambda_1 + \sigma^2} \underline{e}_1 \underline{e}_1^H + \sum_{i=2}^P \frac{1}{\sigma^2} \underline{e}_i \underline{e}_i^H \right] \underline{a}(\tau_o) \\
 &= S_o^2 \left[ \sum_{i=2}^P \frac{1}{\sigma^2} \left( \underline{e}_i^H \underline{a}(\tau_o) \right) \underline{e}_i \right] \quad \dots (2.65)
 \end{aligned}$$

The array output will now be given by

$$\begin{aligned}
 Y(n) &= \underline{w}_{\text{opt}}^H \underline{X}(n) \\
 &\triangleq \beta \left[ \sum_{i=2}^P \frac{1}{\sigma^2} \underline{e}_i^H \underline{a}^H(\tau_o) \underline{e}_i \right] \left[ \underline{b} s(t) + \underline{n}(t) \right] \quad \dots (2.66)
 \end{aligned}$$

Since the vector  $\underline{b}$  lies along  $\underline{e}_1$  and is orthogonal to  $\{\underline{e}_2, \underline{e}_3, \dots, \underline{e}_P\}$ , there will be no desired signal output from the conventional array when the desired signal is coherent with the interfering signals [57]. The output  $Y(n)$ , in this case, will simply be a weighted combination of the noise vector  $\underline{n}(t)$ .

The signal cancellation phenomenon can also be explained with the help of a Frost array, subject to unity gain constraint in the look-direction. In this case, the optimal weight vector is obtained by minimizing the array output power. Therefore, if an interfering signal, say from the direction  $\theta_i$ , is coherent with the desired signal, the minimum will be achieved if and only if, the array gain in the direction  $\theta_i$  is such that the interfering signal exactly cancels the desired signal.

Moreover, steering vector  $\underline{a}(\tau_i)$  has a Vandermonde structure.

The orthogonality of  $\underline{b}$  and  $\underline{W}_{opt}$  will imply nonorthogonality of  $\underline{W}_{opt}$  and any of the  $\{\underline{a}(\tau_1), \dots, \underline{a}(\tau_{K-1})\}$ . Therefore, there can not be nulls in the direction of the interfering signals [57].

To overcome the above mentioned problem, several methods have been proposed in the literature for the cancellation of coherent interferences with different degrees of success [57,11,18]. A practical approach to remove coherence between the sources is "the spatial smoothing scheme" suggested by Evans et al [11]. An analysis of this scheme and its application to adaptive beamforming has been given by Shan and Kailath [57]. This scheme essentially decorrelates the signals and, thus, eliminates the problem encountered with coherent signals.

### 2.3-1 Spatial Smoothing Preprocessing Scheme [SSPS]

Under noncoherent signal conditions, the correlation matrix,  $\Phi$ , is Toeplitz in the case of narrowband beamformers, and block Toeplitz, in the case of broadband beamformers. When coherent signals are present, this Toeplitz structure is destroyed and also, the rank of the matrix  $\Phi$  reduces to unity. A suitable algorithm is, therefore, required to restore the rank of the covariance matrix to  $K$ , where  $K$  is the number of sources.

The spatial smoothing scheme restores the rank of the matrix  $\Phi$ , through progressive diagonalization, thereby decorrelating the sources effectively. In this scheme, the uniform linear array of 'P' sensors is extended by augmenting it with 'L' additional sensors. The extended array is then divided into overlapping subarrays of size 'P'

with first subarray formed from sensors  $\{1, \dots, P\}$ , the second from  $\{2, \dots, P+1\}$ , and so on, as shown in Fig.2.9. First the correlation matrix of each subarray is formed. Next, a spatially smoothed correlation matrix,  $\bar{\Phi}$ , is defined by taking the mean of the subarray correlation matrices, That is

$$\bar{\Phi} = \frac{1}{L+1} \sum_{i=1}^{L+1} \Phi^{(i)} \quad \dots (2.67)$$

If the number of subarrays, so formed, is equal to or greater than the number of sources 'K' and the number of elements in each subarray, 'P' is greater than the number of sources, then the correlation matrix,  $\bar{\Phi}$ , has full rank and the source correlation matrix approaches the same form as the source correlation matrix for noncoherent situation, i.e., a Toeplitz structure. This spatially smoothed correlation matrix can then be used in an optimum beamformer.

The phenomenon of restoration of rank of the correlation matrix can be explained as follows [57].

Let the signal vector for the first subarray be given by

$$\underline{Z}_1(t) = [x_1(t), x_2(t), \dots, x_p(t)]^T \quad \dots (2.68)$$

By shifting down the subarray one element at a time, we form  $\underline{Z}_2(t), \underline{Z}_3(t), \dots, \underline{Z}_{L+1}(t)$ . If we assume that the signal from all the 'K' sources are coherent and are of the form  $e^{j(\omega_o t + \phi_i)}$ ,  $i = 0, 1, \dots, K-1$ , then we can write for the  $k^{th}$  subarray

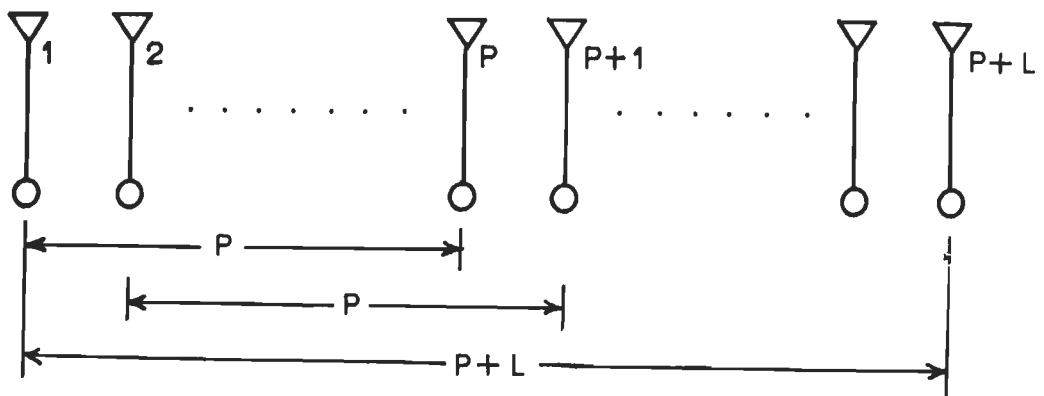


Fig.2.9. A uniform linear array of ' $P+L$ ' sensors divided into overlapping subarrays of size ' $P$ ' each.

$$\underline{z}_k(t) = A D^{k-1} \underline{s} + \underline{n}_k(t) \quad \dots(2.69)$$

where  $D^{k-1}$  is the  $(k-1)^{\text{th}}$  power of the diagonal matrix  $D$  with entries  $\{e^{-j\omega_0 \tau_i}, i=0, 1, \dots, K-1\}$ .

The covariance matrix of the  $k^{\text{th}}$  subarray is given by

$$\begin{aligned} \Phi_{zz}^{(k)} &= E \left[ \underline{z}_k(t) \underline{z}_k(t)^H \right] \\ &= A D^{k-1} \Phi_{ss} \left[ D^{k-1} \right]^H A^H + \sigma^2 I \end{aligned} \quad \dots(2.70)$$

where  $\Phi_{ss}$  is the source covariance matrix defined by

$$\Phi_{ss} = E \left[ \underline{s} \underline{s}^H \right] \quad \dots(2.71)$$

The spatially smoothed correlation matrix is now given by

$$\begin{aligned} \bar{\Phi} &= \frac{1}{L+1} \sum_{l=1}^{L+1} \Phi_{zz}^{(l)} \\ &= \frac{1}{L+1} A \Delta \Sigma \Delta^H A^H + \sigma^2 I \end{aligned} \quad \dots(2.72)$$

where  $\Delta = [D^0, D^1, \dots, D^L]$  and  $\Sigma = I_{L+1} \otimes \Phi_{ss}$  is a block diagonal matrix with  $\Phi_{ss}$  on the diagonal.

By defining

$$\bar{S} = \Delta \Sigma \Delta^H \quad \dots(2.73)$$

the eqn.(2.73) can be written as

$$\bar{\Phi} = \frac{1}{L+1} A \bar{S} A^H + \sigma^2 I \quad \dots (2.74)$$

Since, all the K input signals are coherent, the signal covariance matrix  $\Phi_{SS}$  is a nonnegative definite matrix of rank one. Hence, it can be written as

$$\Phi_{SS} = \Phi_{SS}^{1/2} \left[ \Phi_{SS}^H \right]^{1/2} = \underline{r} \underline{r}^H \quad \dots (2.75)$$

where  $\underline{r}$  is a vector of dimension K and is given by

$$\underline{r} = \left[ r_0, r_1, \dots, r_{K-1} \right]^T \quad \dots (2.76)$$

If  $r_i$  is zero for some i, it means that the  $i^{\text{th}}$  column and  $i^{\text{th}}$  row of  $\Phi_{SS}$  would be zero, which is contrary to the assumption that all inputs are coherent with nonzero power. Therefore, a reasonable assumption is that

$$r_i = 0, \quad i = 0, 1, \dots, K-1 \quad \dots (2.77)$$

using  $\rho\{A\}$  to denote the rank of A, we can say that

$$\rho\left\{ \bar{S} \right\} = \rho\left\{ A \Sigma \Delta^H \right\} = \rho\left\{ \Sigma^{H/2} \Delta^H \right\} \quad \dots (2.78)$$

After some algebraic operations it can be shown that [57]

$$\rho\left\{ \Sigma^{H/2} \Delta^H \right\} = \rho\left\{ F \text{diag}(\underline{r}) \right\} \quad \dots (2.79)$$



where  $F$  is a vandermonde matrix with  $(i+1)^{th}$  column as

$$\left[ 1, e^{-j\omega_0 \tau_i}, e^{-j2\omega_0 \tau_i}, \dots, e^{-jL\omega_0 \tau_i} \right]$$

and  $\text{diag}\{\underline{r}\}$  is a diagonal matrix with elements  $\{r_i, i=0, 1, \dots, K-1\}$ .

Matrix  $F$  has full rank because  $\tau_i$  are assumed distinct.  $\text{Diag}\{r\}$  has also full rank as  $\underline{r}$  has nonzero entries

Therefore,

$$\rho\{\bar{S}\} = \rho\{F\} = \min\{L+1, K\} \quad \dots (2.80)$$

It is thus evident that the rank of the matrix  $\bar{S}$  will be restored to  $K$ , if and only if,  $L + 1 \geq K$ .

Once  $\bar{S}$  has rank  $K$ , the signal subspace will not collapse. Then the noise eigenvectors will be orthogonal to columns of  $A$  and by the analysis given in sec.2.1, will give nulls in the interference directions. From the above analysis, it is seen that the minimum number of subarrays required to restore the rank of the correlation matrix is  $K$ . At the same time, the number of elements in each subarray should be atleast  $K+1$ . Therefore, we must have atleast twice as many sensors as signal sources. In other words, the spatial smoothing scheme suffers from reduced effective aperture.

It may be recalled that, the spatial smoothing scheme restores the rank of the matrix  $\Phi$ , through progressive diagonalization, thereby effectively decorrelating the sources. The rate at which diagonalization takes place depends on the number of subarrays formed. This is referred to as the degree of smoothing [52].

247214



This decorrelation results in reduced signal cancellation and increased rejection of the coherent interference as a function of degree of spatial smoothing. However, the spatial smoothing scheme still suffers from signal cancellation effect and the interference rejection is also not total.

The spatial smoothing scheme has been investigated by several authors. Reddy et al [52] have shown that signal cancellation and interference rejection are strongly influenced by the correlation between the desired signal and interference, with high correlation leading to significant signal loss. This coupling between the correlation and beamformer performance is somewhat weakened as additive sensor noise is increased. These authors have also demonstrated that the rate at which spatial smoothing scheme progressively decorrelates the incident wave fronts, depends upon the spacing and direction of arrival of sources. The number of subarrays required for decorrelating the sources increases as the angular separation  $|\theta_0 - \theta_1|$  becomes smaller and smaller. In particular, the number of elements required goes up when the sources approach the end-fire directions. Yeh et al [69] have derived an expression for the output signal-to-noise ratio(SNR) for a spatially smoothed adaptive array. Their results show that the array performance depends upon the number of subarrays, the angular separation, relative power levels and initial phase difference between the desired signal and coherent interference. In order to have good interference suppression,  $|\theta_0 - \theta_1|$  should be greater than the beamwidth of the array. There is a compromise between increasing the number of subarrays and decreasing the number of elements in each subarray for better performance.

Lineberger and Johnson [33] have analysed the structure of the correlation matrix in a coherent signal environment and have shown that coherence among signals induces a modulation along the diagonals of the correlation matrix,  $\Phi$ , because of which the matrix loses its Toeplitz structure. They also show that, the spatial smoothing is essentially restricted to equally spaced arrays and even after spatial smoothing the sources may remain coherent. Further, inspite of the rank being restored, the correlation matrix may be badly conditioned. In essence, the spatial smoothing scheme suffers from the reduced effective aperture, signal cancellation phenomenon and the interference suppression is also not total.

Several modifications and improvements over the basic spatial smoothing scheme and some alternative spatial averaging schemes have been suggested in the literature to overcome these limitations, viz, the reduced effective aperture, signal cancellation phenomenon and the incomplete suppression of coherent interferences.

An important modification is the forward/backward spatial smoothing scheme [11] which achieves a larger effective aperture as compared to the spatial smoothing scheme without much increase in computational burden. Though this scheme has received wide attention in Direction-of-Arrival(DoA) estimation [66,50], not much attention has been paid to its application in adaptive beamforming.

The salient features of the method can be explained as follows. First, the subarrays are formed as in the spatial smoothing scheme. These are referred to as forward subarrays. In addition, an equal number of identical subarrays are formed in the backward direction with the last element in the array being treated as the

first element of the first backward subarray. The input signal to the backward subarrays are complex conjugated. Next, the subarray correlation matrices for each of the forward and backward subarrays are formed. Forward and backward subarray correlation matrices are averaged separately to obtain the forward and backward smoothed correlation matrices. Finally, these two correlation matrices are averaged and forward/backward smoothed correlation matrix is formed. This method has been reported to effect a saving of  $1/4K$  elements in the required  $2K$  elements of the spatial smoothing scheme.

Su et al [56] have presented an alternative approach called the "spatial processing algorithm" to combat the signal cancellation effect. This method employs a number of subbeamformers having the same structure as the conventional beamformers which are arranged in a parallel manner. For the first time instant, the first subbeamformer is employed to update the weights and then these weights are copied into the remaining subbeamformers. For the second time instant, the weights are updated using the second subbeamformer and then the weights are copied into the remaining subbeamformers. So the adaptation process sequentially propagates, one by one, along the subbeamformers. After the adaptation reaches the last subbeamformer, the process restarts from the first one. The array output is then computed by averaging the delayed outputs of all these subbeamformers. In this method, this weight propagation from one subbeamformer to another incorporates spatial averaging.

Widrow et al [64] have proposed an adaptive beamforming scheme using two beamformers(master and slave beamformers) to separate the desired signal and the interferences during adaptation. This

method does not involve any spatial averaging and can be explained as follows. The Frost adaptive beamformer is employed to generate a suitable set of weights to satisfy the look-direction gain constraint and to minimize the output power. The weights are then deployed in the slaved beamformer to provide jammer rejection without signal cancellation. However, only one coherent jammer can be suppressed by this beamformer. Recently, Pei et al [46] have replaced the conventional Frost beamformer in the structure of reference [64] by an optimum beamformer and have incorporated the spatial smoothing scheme of Shan et al [57] into the master beamformer to overcome signal cancellation.

Lee and Wu [30] have proposed an algorithm which can create an adaptive beamformer to reject coherent jammers while providing the desired signal reception in the look-direction. The desired signal is removed from the received signal in an adaptive process, which is additional to the adaptive procedure of the spatial smoothing scheme. After the spatial smoothing process, an appropriate set of weights and an estimate of the desired signal is obtained. A slave beamformer, based on this set of weights, is then utilized to achieve the goal of coherent jammer rejection while simultaneously preserving the desired signal.

As mentioned earlier, complete suppression of interferences is not possible in spatial smoothing technique. The reason for this is that the resultant spatially smoothed matrix can not be made close to Toeplitz by simple averaging over the finite array aperture. Takao and Kikuma [59] describe a technique called "Adaptive spatial averaging". In this technique, the input correlation matrices of the subarrays are

adaptively averaged so as to produce a Toeplitz matrix which would be obtained if the interference were not correlated with the desired signal. The averaged matrix is free from correlation terms between the desired signal and interference and, therefore, may be used to derive optimum weight for the array element just as in the noncoherent interference environment.

The spatial smoothing scheme does not consider the cross correlations of the subarray outputs. Du and Kirlin [10] have proposed an improved spatial smoothing scheme which fully utilizes the correlations of the array outputs and produces a more stable estimate of the covariance matrix.

### 2.3-2 Structured Correlation Matrix Method [SCMM]

An alternative solution to the problem of coherent interference suppression has been suggested by Godara [18]. This method, called the structured correlation matrix method [SCMM], exploits the structure of the array correlation matrix (ACM). In noncoherent signal environment, the ACM has a Toeplitz structure which is lost when correlated sources are present. In the structured correlation matrix method (SCMM), this constraint, ie, Toeplitz structure is implemented by averaging the unconstrained correlation matrix along the diagonals. The entries along the  $i^{\text{th}}$  diagonal of this structure correlation matrix (SCM),  $\bar{\Phi}_i$ , are given by

$$\bar{\Phi}_i = \frac{1}{P-i} \sum_{l=1}^{P-i} \Phi_{l, l+i}, \quad i = 0, 1, \dots, P-1 \quad \dots (2.81)$$

This structured correlation matrix is now used to compute the weights of the beamformer.

In this method, the size of the correlation matrix does not change, unlike the case of spatial smoothing preprocessing scheme. The array aperture is, therefore, preserved in this case. At the same time, however, there is a possibility of an error creeping in because of the difference between the elements of true correlation matrix and the Toeplitzed matrix. Under certain circumstances, the Toeplitzed structure may not resemble the actual correlation matrix at all.

### 2.3-3 Sample Results

The effectiveness of these two techniques viz, the spatial smoothing preprocessing scheme (SSPS) and the structured correlation matrix method (SCMM), in nulling the coherent interferences is demonstrated by the following simulation example.

A 6-element uniform linear array of isotropic elements has been considered. The signal environment is assumed to consist of a desired signal and four interferences, two of which are fully coherent with the desired signal. The various parameters of the interferences are listed in Table-2.4.

**Table - 2.4**  
**Interference Parameters**

Parameter	Interference 1	Interference 2	Interference 3	Interference 4
$S_i$	10.0	10.0	10.0	10.0
$\theta_i$	$15^\circ$	$55^\circ$	$-30^\circ$	$-70^\circ$
$\omega_i$	1.0	1.0	1.1	0.9

The above signal environment was first used in the optimum beamformer of sec.2.1. 1000 snapshots of data were utilized to compute the array correlation matrix. A noise variance of 0.01 was then added

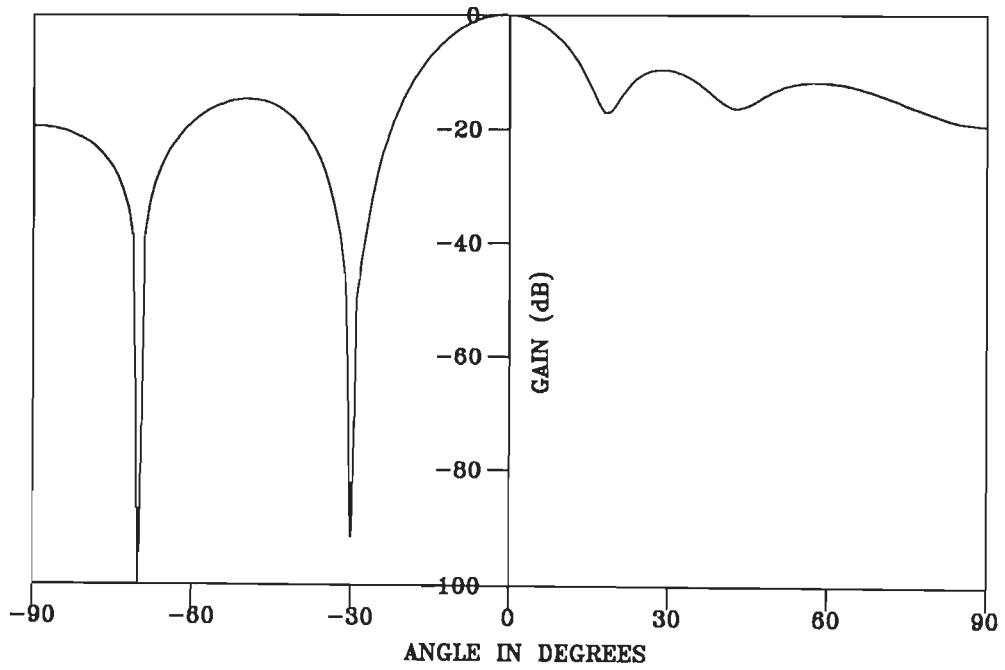
to the diagonal elements of  $\Phi$ . The resulting pattern is shown in Fig.2.10. It is found that while deep nulls occur in the direction of noncoherent interferences  $(-30^\circ, -70^\circ)$ , the beamformer fails to put sharp nulls in the directions of coherent interferences arriving at  $15^\circ$  and  $55^\circ$ .

Next, the structured correlation matrix method and the spatial smoothing preprocessing scheme were used in the optimum beamformer. In the latter case, the array was augmented by four elements and five overlapping subarrays of six elements each were formed. The resulting patterns are shown in Fig.2.10, where it is observed that both the techniques successfully place sharp nulls in the direction of all the four interferences.

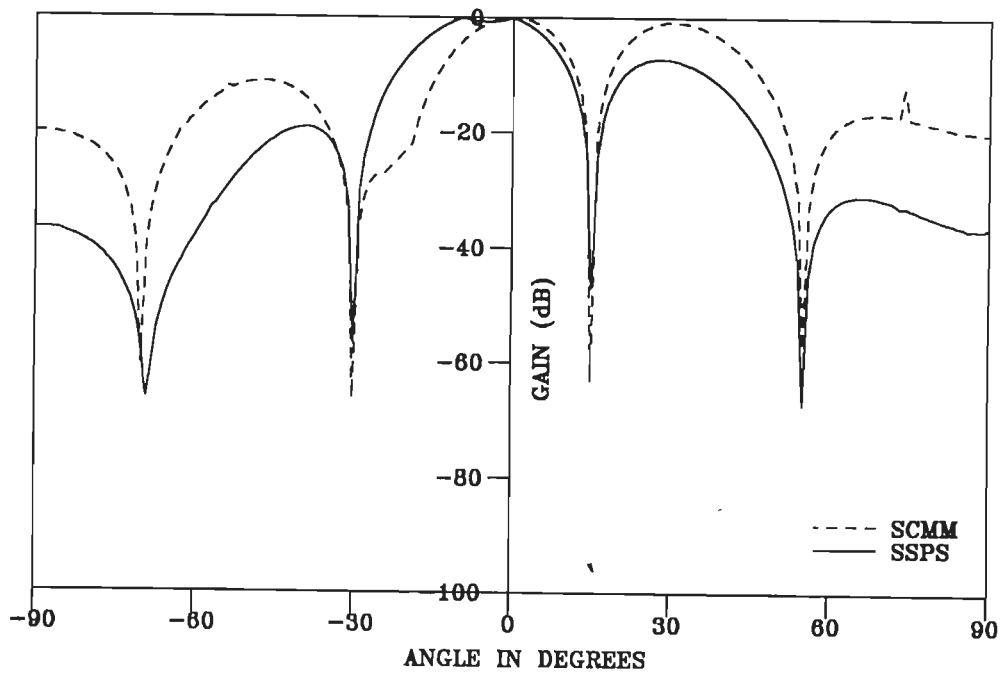
#### **2.4 A COMPARISON OF STRUCTURED CORRELATION MATRIX METHOD (SCMM) AND THE SPATIAL SMOOTHING PREPROCESSING SCHEME (SSPS)**

Of the two methods discussed in the previous section, the spatial smoothing scheme has received considerable attention and several modifications have been proposed in the literature [41,46,56] to overcome its two main draw backs, viz, signal cancellation phenomenon and the reduction in effective aperture area. The structured correlation matrix method, on the other hand, is relatively recent and has not been studied in detail, so far. Only Indukumar and Reddy [24] have addressed this technique in the context of Direction-of-Arrival estimation using signal subspace algorithms [44]. They have shown that the resulting covariance matrix, in this case, is not guaranteed to be nonnegative definite and induces a bias in DOA estimates. We, therefore, present here a comparative study of the two





(a) OPTIMUM BEAMFORMER



(b) SSPS AND SCMM INCORPORATED OPTIMUM BEAMFORMERS

FIG.2.10 VOLTAGE PATTERNS WITH DESIRED SIGNAL AT  $0^\circ$ , COHERENT INTERFERENCES AT  $15^\circ$  &  $55^\circ$  AND NONCOHERENT INTERFERENCES AT  $-30^\circ$  &  $-70^\circ$ .

techniques which is based on computer simulations [25].

In all the numerical examples presented here, a six element array has been assumed in the case of SCMM. In the implementation of spatial smoothing scheme, this array has been augmented by four elements and five overlapping subarrays of 6 elements each have been formed. The overall array size in SSPS, is therefore 10 elements. The signal environment consists of a desired signal incident from broadside and two fully correlated interferences.

As a first example, we consider two interferences which are closely spaced to the desired signal (Table 2.5).

**Table - 2.5**

**Interference parameters**

Parameter	Interference 1	Interference 2
$S_i$	10.0	10.0
$\theta_i$	$10^\circ$	$-10^\circ$
$\omega_i$	1.0	1.0

The computed patterns are shown in Fig.2.11. As can be seen from the figure, both the methods succeed in placing nulls in the direction of interferences. However, the nulls placed by SSPS are much deeper than those produced by SCMM (Table 2.6). Further, in both the methods, the maxima of pattern gain occur in directions other than the direction of arrival of the desired signal.

Table - 2.6

Null Depths Produced by SCMM and SSPS Beamformers

$\theta_i$	Null depths	
	SCMM	SSPS
$10^\circ$	-29dB	-43dB
$10^\circ$	-29dB	-43dB

Next, we consider two widely separated interferences which are unsymmetrically located on either side of the desired signal at  $30^\circ$  and  $-50^\circ$ . Fig.2.12 compares the output voltage patterns for the two techniques. It can be seen that while the spatial smoothing scheme places sharp nulls exactly at the location of interferences, a bias is introduced by SCMM. The null is produced at  $-46^\circ$  instead of at  $-50^\circ$ . Also, there is a spurious null at  $35^\circ$ . The reason for this offset and the spurious null is that the resulting matrix produces a signal subspace which is inconsistent with that of the underlying signal model.

Fig.2.13 shows the voltage patterns when interferences are modelled to arrive at  $+70^\circ$  and  $-70^\circ$ , that is from near end-fire directions. It can be seen that the SSPS produces sharp nulls of depth -60dB in the direction of interferences. The SCMM, on the other hand, fails to produce nulls at  $\pm 70^\circ$ .

From the above examples, it is evident that, in the case of structured correlation matrix method, a bias is introduced in the placing of nulls, which increases as the interferences are moved away from the broadside (desired signal arrival angle). This increasing bias, ultimately, leads to the array's failure in placing nulls in the

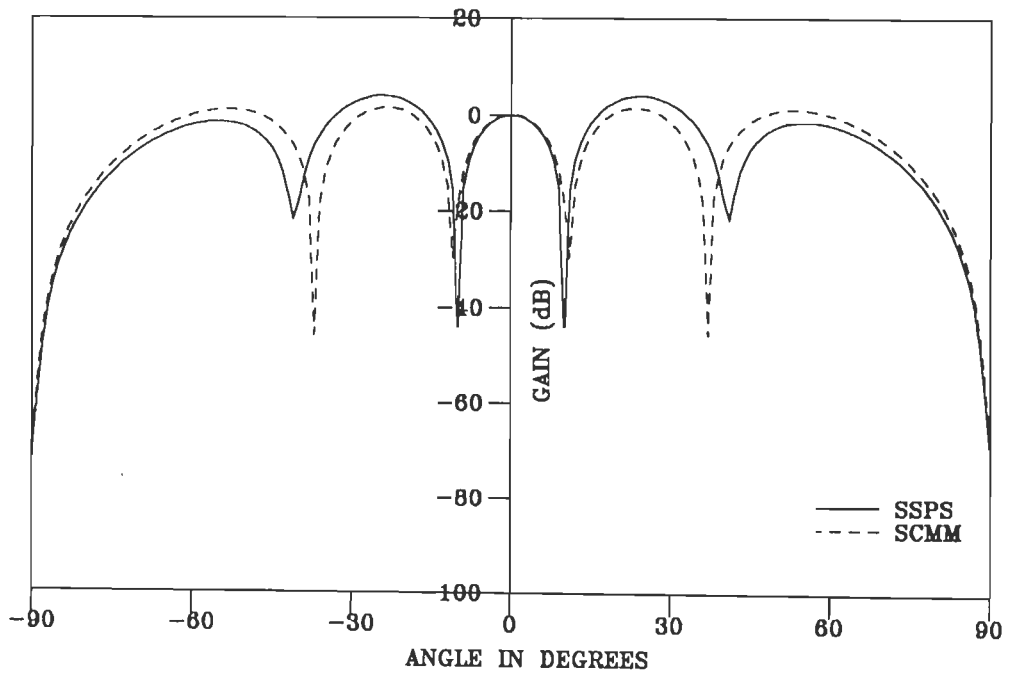


FIG.2.11 VOLTAGE PATTERNS OF THE SSPS AND SCMM BEAMFORMERS WITH INTERFERENCES AT  $-10^\circ$  &  $10^\circ$ .

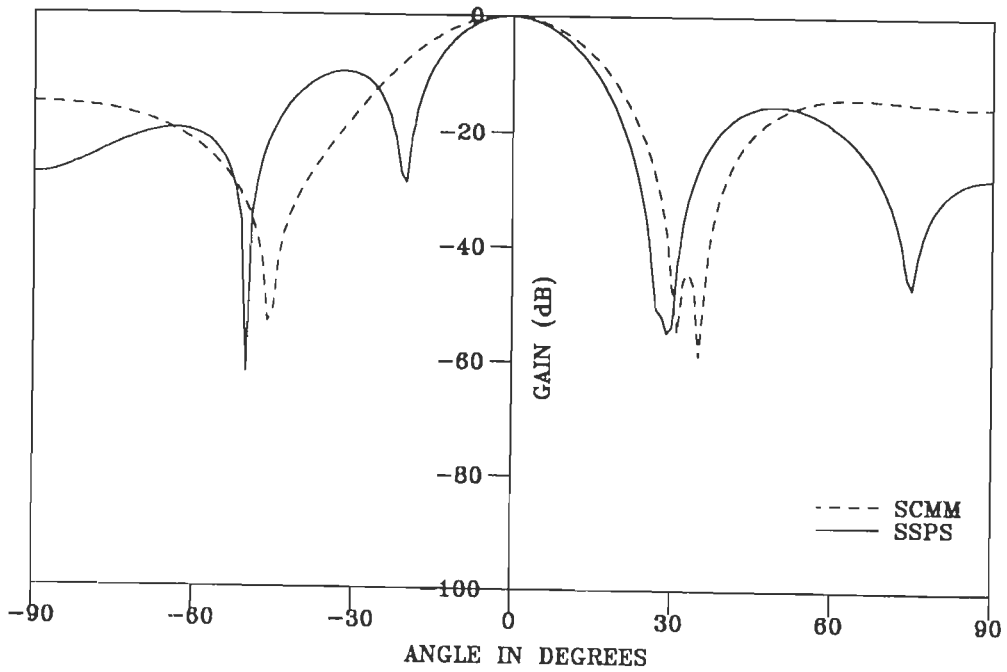


FIG.2.12 VOLTAGE PATTERNS OF THE SSPS AND SCMM BEAMFORMERS WITH INTERFERENCES AT  $-50^\circ$  &  $30^\circ$ .

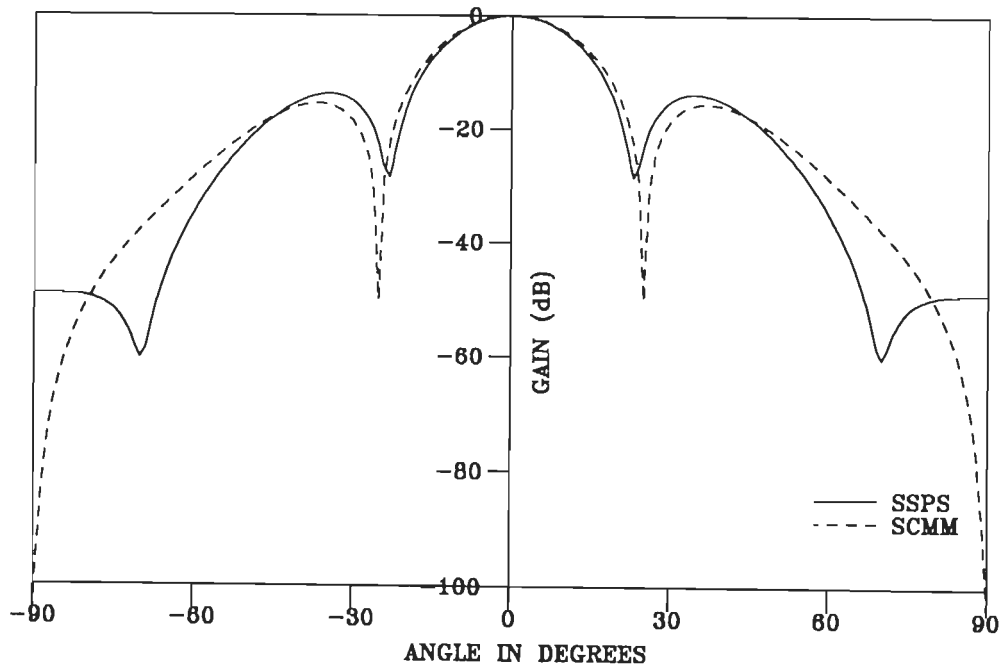


FIG.2.13 VOLTAGE PATTERNS OF THE SSPS AND SCMM BEAMFORMERS WITH INTERFERENCES AT  $-70^\circ$  &  $70^\circ$ .

direction of near end-fire interferences.

The simulation results show that the spatial smoothing scheme is more promising for the suppression of interferences. Its implementation does not change the signal processing operations significantly; only the dimensions of the correlation matrix are altered. The structured correlation matrix method, on the other hand, is much simpler in implementation compared to the spatial smoothing scheme and also, it preserves the array aperture. Extensive computer simulation studies have revealed that only for certain combinations of  $S_o, S_i, \theta_d, \theta_i$  and  $P$ , the SCMM nulls the interferences satisfactorily ; for other combinations, the method fails. Moreover, SCMM is not applicable to broadband arrays for which the array correlation matrix will not be Toeplitz, even in noncoherent situations [25]. Finally, the adaptive implementation of SCMM does not seem to be possible, whereas, SSPS lends itself to adaptive implementation to real time applications.

## 2.5 SUMMARY

In this chapter, a general formulation of the adaptive array utilizing the Least-Mean-Square criterion has been discussed. The broadband signal simulation which has not been discussed in the adaptive array literature, so far, has been discussed by devoting a separate subsection. The reasons for the failure of conventional beamforming arrays in coherent signal environment has been discussed and, as a remedy, various spatial averaging schemes have been described. A comparison of the spatial smoothing scheme and the structured correlation matrix method using computer simulations

revealed that the former is superior method for combating the signal cancellation in coherent signal environment.

## CHAPTER - 3

### RECURSIVE LEAST-SQUARES ALGORITHMS FOR ADAPTIVE BEAMFORMING

The computational problems associated with the calculations of weight coefficients in the direct matrix inversion approach can be avoided by using the LMS algorithm. However, the drawbacks of the LMS algorithm, namely, the slow rate of convergence and the dependence of time constant on the eigenvalue spread, has motivated the search for adaptive filtering algorithms which provide faster convergence and are not sensitive to signal statistics. In this, least-squares [LS] estimation has played a prominent role. The most popular time recursive LS estimation scheme is the RLS algorithm [3] whose time recursive nature makes it attractive for adaptive beamformers. However, it is sensitive to roundoff errors, when finite precision arithmetic is used for its implementation [36]. To remedy this problem, algorithms based on matrix factorization and orthogonal transformations have been derived and investigated. The orthogonal transformations, such as, Givens, Householder and modified Gram-Schmidt, are known to be less sensitive to roundoff errors. Time recursive version of the Givens transformation and modified Gram-Schmidt procedure have also been developed and discussed in the context of systolic array implementation [15,36].

The QRD-LS algorithm using Givens rotations has been applied to adaptive beamformers by Ward et al [62]. He also discusses its implementation using a triangular systolic array where the error residual is extracted directly as the output of the array. If the weight coefficients are also to be computed, the back substitution



method may be used which can also be implemented using a linear systolic array.

Lewis [34] has used the above algorithm for beamforming in a situation where the information regarding the desired signal is not known, but a priori information in the form of signal-to-data cross correlation vector is available. Heiligman and Purdy [19] have described a method of weight computation using the triangular processor itself. They have also discussed its property of graceful degradation; the loss of one or more component processors does not cause the computed weights to degrade catastrophically.

Little attention has been paid in the literature so far, towards the application of time recursive form of modified Gram-Schmidt procedure to adaptive beamforming problem. Ling and Proakis [36] have presented a time recursive form of Gram-Schmidt algorithm [RMGS] for solving a general least-square minimization problem. This algorithm is reported to be robust to roundoff errors and can be implemented using systolic structure. As a consequence, the RMGS algorithm is well suited for adaptive beamforming problem. Ling et al [36] have presented an improved error feedback version of the RMGS algorithm [RMGSEF] which is more robust as compared to RMGS algorithm. In this chapter, we present the results of extensive computer simulations which show that the RMGS class of beamformers offer a good compromise between the RLS and QRD-LS beamformers with reference to stability and computational complexity.

The organization of this chapter is as follows. The exact least-square error criterion is defined in section 3.1, with reference to the adaptive beamforming problem. Sec. 3.2 gives an overview of

the RLS algorithm. In sec.3.3, the exact least-square error criterion is redefined in data domain and the QRD-LS algorithm, in general and the QRD-LS algorithm using Givens rotations, in particular, are discussed. The proposed RMGS algorithm and its application to the adaptive beamformers is presented in sec.3.4. The application of these techniques in broadband signal environment is discussed in sec.3.5. Results of an extensive numerical study of these beamformers, based on computer simulations, is reported in sec.3.6. Finally, a discussion of the relative performance of these adaptive beamformers is presented in sec.3.7.

### 3.1 THE EXACT LEAST-SQUARE ERROR CRITERION

Consider the P-element linear narrowband array model of Fig.2.1. In the RLS algorithm, the computation is started with known initial conditions and the information contained in the new input data samples to the array is used to update the old estimates. Therefore, the length of the observable data is variable. Accordingly, the index of performance to be minimized is expressed as  $\xi(n)$ , where 'n' is the variable length of the observable data. Moreover, a weighting factor is introduced into the definition of the performance index to ensure that the data in the distant past are forgotten. Thus,

$$\xi(n) = \sum_{i=1}^n \lambda^{n-i} |e(i)|^2 \quad \dots (3.1)$$

where  $\lambda^{n-i}$  is the exponential weighting factor and  $e(i)$  is the difference between the desired signal and the array output. That is

$$e(i) = d(i) - \underline{W}^H(n) \underline{X}(i), \quad i = 1, \dots, n \quad \dots (3.2)$$

where  $\underline{X}(i)$  is the array input data vector at time  $i$ , defined by

$$\underline{X}^T(i) = [x_1(i), x_2(i), \dots, x_p(i)] \quad \dots (3.3)$$

and  $\underline{W}(n)$  is the tap weight vector at time 'n' defined by

$$\underline{W}^T(n) = [w_1(n), w_2(n), \dots, w_p(n)]. \quad \dots (3.4)$$

The optimum value of the weight vector,  $\underline{W}_{opt}(n)$ , for which the performance index  $\xi(n)$  of eqn.(3.1) attains the minimum value is given by

$$\underline{W}_{opt}(n) = \Phi^{-1}(n) \underline{\theta}(n) \quad \dots (3.5)$$

where,

$$\Phi(n) = \sum_{i=1}^n \lambda^{n-i} \underline{X}(i) \underline{X}^H(i) \quad \dots (3.6)$$

and

$$\underline{\theta}(n) = \sum_{i=1}^n \lambda^{n-i} \underline{X}(i) d^*(i) \quad \dots (3.7)$$

The recursive relations for  $\Phi(n)$  and  $\underline{\theta}(n)$  can be obtained directly from eqns.(3.6) and (3.7) and are given by

$$\Phi(n) = \lambda \Phi(n-1) + \underline{X}(n) \underline{X}^H(n) \quad \dots (3.8)$$

$$\underline{\theta}(n) = \lambda \underline{\theta}(n-1) + \underline{X}(n) d^*(n) \quad \dots (3.9)$$

The traditional approach for computing eqns.(3.5), (3.8) and (3.9) is to use the matrix inversion lemma [22], which enables the recursive computation of  $\Phi^{-1}(n)$  instead of  $\Phi(n)$  and leads to the familiar RLS algorithm.

### 3.2 THE RECURSIVE LEAST-SQUARES [RLS] ALGORITHM

The recursive equation for updating the weight vector in accordance with the least-square error criterion can be shown to be [22]

$$\underline{W}(n) = \underline{W}(n-1) + \underline{C}(n) \eta^*(n) \quad \dots (3.10)$$

where  $\underline{C}(n)$  and  $\eta(n)$  are, respectively, referred to as the gain vector and a priori estimation error and are given by

$$\underline{C}(n) = \frac{\lambda^{-1} \Phi^{-1}(n-1) \underline{X}(n)}{1 + \lambda^{-1} \underline{X}^H(n) \Phi^{-1}(n-1) \underline{X}(n)} \quad \dots (3.11)$$

$$\eta(n) = d(n) - \underline{W}^H(n-1) \underline{X}(n) \quad \dots (3.12)$$

Using the matrix inversion lemma, the recursive equation for the inverse of the correlation matrix is obtained as

$$\Phi^{-1}(n) = \lambda^{-1} \Phi^{-1}(n-1) - \lambda^{-1} \underline{C}(n) \underline{X}^H(n) \Phi^{-1}(n-1) \quad \dots (3.13)$$

Equations(3.10) to (3.13) constitute the RLS algorithm which is summarized in Table-3.1 [22].

The convergence rate of RLS algorithm is much superior to that of LMS algorithm when convergence time is measured in terms of the number of samples of the input data [8]. However, this improvement in performance is achieved at the expense of a large increase in the computational complexity. Specifically, to compute the gain vector  $\underline{C}(n)$ , a P-by-P matrix  $\Phi^{-1}(n)$  must be adapted and stored once per iteration. Hence, on the order of  $P^2$  arithmetic operations must be performed per iteration of the RLS algorithm. This is in direct

contrast to the LMS algorithm, in which an order of  $P$  arithmetic operations are required. It may be mentioned here that the RLS algorithm is useful only when the number of elements  $P$  in the array is small and the eigenvalue spread is high.

**Table 3.1**  
**The RLS Algorithm**

Initialize the algorithm by setting

$$\Phi^{-1}(0) = \delta^{-1} I, \quad \delta = \text{small positive constant}$$

$$\underline{W}(0) = 0$$

Algorithm	Computational complexity
For $n = 1, 2, \dots$ compute	
$\underline{u}(n) = \lambda^{-1} \Phi^{-1}(n-1) \underline{X}(n)$	$P^2$ (T3.1.1)
$\underline{C}(n) = \frac{\underline{u}(n)}{[1 + \underline{X}^H(n) \underline{u}(n)]}$	$P$ (T3.1.2)
$\eta(n) = d(n) - \underline{W}^H(n-1) \underline{X}(n)$	$P$ (T3.1.3)
$\underline{W}(n) = \underline{W}(n-1) + \underline{C}(n) \eta^*(n)$	$P$ (T3.1.4)
$\Phi^{-1}(n) = \lambda^{-1} \Phi^{-1}(n-1) - \underline{C}(n) \underline{u}^H(n)$	$2P^2$ (T3.1.5)
<b>Total</b>	<b><math>((P^2 + 2P) \text{ divisions}) + 3P^2 + 3P</math></b>

### 3.3 THE QRD-LS ALGORITHM

The RLS algorithm discussed in the previous section has two main drawbacks, namely, its sensitivity to roundoff error and the difficulty in implementing the algorithm in VLSI technology. An alternative approach which is numerically sound is that of orthogonal

triangularization [61]. It is based on updating the upper triangular matrix, which is obtained by QR decomposition of the n-by-P data matrix. Since, the condition number of the data matrix is much smaller than the condition number of the correlation matrix, any algorithm that operates directly on the data is much better conditioned. The added advantage is that these QR decomposition algorithms can be implemented in a highly pipelined manner using systolic arrays.

In the following, we derive the QRD-LS algorithm based on QR updating.

### 3.3-1 Exact Least-Square Error Criterion in the Data Domain

We have till now, defined the exact least-square error criterion in the covariance matrix domain. The exact least-square error criterion defined in eqn.(3.1) can also be expressed in data domain. This is accomplished by defining a n-by-P data matrix  $A(n)$  such that

$$A^H(n) = [\underline{X}(1), \underline{X}(2), \dots, \underline{X}(n)]$$

$$= \begin{bmatrix} x_1(1) & x_1(2) & \dots & x_1(n) \\ x_2(1) & x_2(2) & \dots & x_2(n) \\ \vdots & \vdots & \ddots & \vdots \\ x_p(1) & x_p(2) & \dots & x_p(n) \end{bmatrix} \quad \dots (3.14)$$

Let n-by-1 vectors  $\underline{e}(n)$  and  $\underline{b}(n)$ , respectively, denote the error vector and the desired response vector, which are defined by

$$\underline{e}^H(n) = [e(1), e(2), \dots, e(n)] \quad \dots (3.15)$$

$$\underline{b}^H(n) = [d(1), d(2), \dots, d(n)] \quad \dots (3.16)$$

The index of performance can now be redefined as

$$\xi(n) = \underline{\varepsilon}^H(n) \Lambda(n) \underline{\varepsilon}(n) \quad \dots (3.17)$$

where  $\Lambda(n)$  is the n-by-n exponential weighting matrix

$$\Lambda(n) = \text{diagonal} \left[ \lambda^{n-1}, \lambda^{n-2}, \dots, 1 \right] \quad \dots (3.18)$$

The inclusion of  $\Lambda(n)$ , the exponential weighting matrix, has the effect of progressively weighting against the preceding column of the data matrix  $A^H(n)$  in favour of the last column. The last column of the  $A^H(n)$  corresponds to the input vector  $\underline{X}(n)$  at time n, for which the weighting factor is unity.

The index of performance may then be redefined as

$$\xi(n) = \left\| \Lambda^{1/2}(n) \underline{\varepsilon}(n) \right\|^2 \quad \dots (3.19)$$

and is in the form of squared Euclidean norm. The problem we have to solve is to find the least-squares value of the weight vector that minimizes the performance index  $\xi(n)$ .

### 3.3-2 QRD-LS Algorithm

Since the norm of the weight vector is unaffected by premultiplication by an unitary matrix  $Q(n)$ , the index of performance  $\xi(n)$  can also be expressed as

$$\xi(n) = \left\| Q(n) \Lambda^{1/2}(n) \underline{\varepsilon}(n) \right\|^2 \quad \dots (3.20)$$

The error vector  $\underline{\varepsilon}(n)$  is given by

$$\underline{\varepsilon}(n) = \underline{b}(n) - A(n) \underline{W}(n) \quad \dots (3.21)$$

Hence, the vector  $Q(n) \wedge^{1/2}(n) \underline{\varepsilon}(n)$  in (3.20) may be expressed as

$$Q(n) \wedge^{1/2}(n) \underline{\varepsilon}(n) = Q(n) \wedge^{1/2}(n) \underline{b}(n) - Q(n) \wedge^{1/2}(n) A(n) \underline{W}(n) \quad \dots (3.22)$$

The orthogonal matrix  $Q(n)$  is generated such that it applies an orthogonal triangularization to the weighted data matrix and transforms it to upper triangular form.

$$Q(n) \wedge^{1/2}(n) A(n) = \begin{bmatrix} Q_1(n) \\ Q_2(n) \end{bmatrix} \wedge^{1/2}(n) A(n) = \begin{bmatrix} R(n) \\ 0 \end{bmatrix} \quad \dots (3.23)$$

Here  $Q_1(n)$  contains first  $P$  rows of  $Q(n)$  and  $Q_2(n)$  contains the remaining  $(n-P)$  rows. An orthogonal matrix can always be constructed such that  $R(n)$  is a  $P$ -by- $P$  upper triangular matrix with nonnegative diagonal elements and '0' is an  $(n-P)$ -by- $(n-P)$  null matrix. This factorization is referred to in the literature as the QR decomposition.

Rather than solve the normal equations, the QR method uses  $Q(n)$  from (3.23) to rotate (3.22) into

$$\begin{aligned} \begin{bmatrix} Q_1(n) \\ Q_2(n) \end{bmatrix} \wedge^{1/2}(n) \underline{\varepsilon}(n) &= \begin{bmatrix} Q_1(n) \\ Q_2(n) \end{bmatrix} \wedge^{1/2}(n) \underline{b}(n) - \begin{bmatrix} Q_1(n) \\ Q_2(n) \end{bmatrix} \wedge^{1/2}(n) A(n) \underline{W}(n) \\ &= \begin{bmatrix} \underline{P}(n) \\ \underline{V}(n) \end{bmatrix} - \begin{bmatrix} R(n) \\ 0 \end{bmatrix} \underline{W}(n) \quad \dots (3.24) \end{aligned}$$

Here  $\underline{P}(n) = Q_1(n) \wedge^{1/2}(n) \underline{b}(n)$ , contains the first  $P$  elements of the rotated desired response vector, while  $\underline{V}(n)$  contains the



remaining  $(n-P)$  elements, i.e.,  $Q_2(n) \wedge^{1/2}(n) \underline{b}(n)$ .

To solve the LS problem, we choose the weight vector  $\underline{W}(n)$  so as to minimize the performance index  $\xi(n)$ . If  $\underline{W}_{opt}(n)$  denotes this optimum value of the weight vector, it is evident from eqn.(3.24) that, the squared norm of  $Q(n) \wedge^{1/2}(n) \underline{e}(n)$  is a minimum, when

$$R(n) \underline{W}_{opt}(n) = \underline{P}(n) \quad \dots(3.25)$$

Correspondingly, the minimum value of the performance index is given by

$$\xi_{min}(n) = \|\underline{V}(n)\|^2 \quad \dots(3.26)$$

### 3.3-3 Recursive Implementation

To develop a recursive implementation of the above procedure, we assume that at time  $(n-1)$ , an  $(n-1)$ -by- $(n-1)$  matrix  $Q(n-1)$  is known such that

$$Q(n-1) \wedge^{1/2}(n-1) A(n-1) = \begin{bmatrix} R(n-1) \\ 0 \end{bmatrix} \quad \dots(3.27)$$

where  $R(n-1)$  is a  $P$ -by- $P$  upper triangular matrix and '0' denotes the  $(n-1-P)$ -by- $P$  null matrix.

At time  $n$ , the data matrix  $A(n)$  and the desired response vector  $\underline{b}(n)$  can be partitioned as

$$A(n) = \begin{bmatrix} A(n-1) \\ \underline{x}^H(n) \end{bmatrix} \quad \dots(3.28)$$

$$\underline{b}(n) = \begin{bmatrix} \underline{b}(n-1) \\ \hline d^*(n) \end{bmatrix} \quad \dots (3.29)$$

Correspondingly, the n-by-n exponential weighting matrix  $\Lambda(n)$  satisfies the recursion

$$\Lambda(n) = \begin{bmatrix} \lambda \Lambda(n-1) & \underline{0} \\ \underline{0}^T & 1 \end{bmatrix} \quad \dots (3.30)$$

To compute the QR decomposition of the updated data matrix  $A(n)$ , we first define an n-by-n orthogonal matrix  $\bar{Q}(n-1)$  which is related to  $Q(n-1)$  as

$$\bar{Q}(n-1) = \begin{bmatrix} Q(n-1) & \underline{0} \\ \underline{0}^T & 1 \end{bmatrix} \quad \dots (3.31)$$

Therefore,

$$\bar{Q}(n-1) \Lambda^{1/2}(n) A(n) = \begin{bmatrix} \lambda^{1/2} Q(n-1) \Lambda^{1/2}(n-1) A(n-1) \\ \hline \underline{X}^H(n) \end{bmatrix} = \begin{bmatrix} \lambda^{1/2} R(n-1) \\ \hline \underline{0} \\ \hline \underline{X}^H(n) \end{bmatrix} \quad \dots (3.32)$$

The n-by-P matrix on the right hand side of eqn.(3.32) is partially triangularized in that only the last row of the matrix consists of non-zero elements.

Similarly,

$$\begin{aligned} \bar{Q}(n-1) \Lambda^{1/2}(n) \underline{b}(n) &= \begin{bmatrix} \lambda^{1/2} Q(n-1) \Lambda^{1/2}(n-1) \underline{b}(n-1) \\ \hline d^*(n) \end{bmatrix} \\ &= \begin{bmatrix} \lambda^{1/2} \underline{P}(n-1) \\ \hline \lambda^{1/2} \underline{V}(n-1) \\ \hline d^*(n) \end{bmatrix} \quad \dots (3.33) \end{aligned}$$

The orthogonal triangularization can be completed by using an update matrix  $\hat{Q}(n)$  which rotates the bottom row into upper triangular portion of the matrix. Mathematically, this takes the form

$$\hat{Q}(n) \bar{Q}(n-1) \lambda^{1/2}(n) A(n) = \begin{bmatrix} \underline{R}(n) \\ 0 \end{bmatrix} \quad \dots (3.34)$$

Similarly, we can write

$$\hat{Q}(n) \bar{Q}(n-1) \lambda^{1/2}(n) \underline{b}(n) = \hat{Q}(n) \begin{bmatrix} \lambda^{1/2} \underline{P}(n-1) \\ \lambda^{1/2} \underline{V}(n-1) \\ d^*(n) \end{bmatrix} = \begin{bmatrix} \underline{P}(n) \\ \underline{V}(n) \end{bmatrix} \quad \dots (3.35)$$

Having computed the updated matrix  $R(n)$  from eqn.(3.34) and  $\underline{P}(n)$  from eqn.(3.35), we may use back substitution in eqn.(3.25) to compute the corresponding updated value,  $\underline{W}(n)$ , of the least-squares weight vector. Equations(3.25), (3.34) and (3.35) constitute the algorithm which is referred to as the QR decomposition least-square (QRD-LS) algorithm. A summary of this algorithm is presented in Table 3.2 [22].

### 3.3-4 Givens Rotations

The orthogonal triangularization process may be carried out using the Givens rotation procedure. Through successive application of Givens rotations, we may develop a very efficient algorithm for solving the linear least-square problem, where by orthogonal triangularization of the data matrix is recursively updated as each new set of data enters the computation [51].

**Table-3.2**  
**The Recursive QRD-LS Algorithm**

1. Initialize the orthogonal triangularization procedure

$$\underline{R}(0) = \underline{0} \quad \text{and} \quad \underline{P}(0) = \underline{0} \quad \dots (\text{T.3.2.1})$$

the exact initialization occupies the period  $0 \leq n \leq P$ .

2. For  $n > P$ , perform

- (a) update the P-by-P matrix  $\underline{R}(n)$  using the recursion

$$\begin{bmatrix} \underline{R}(n) \\ 0 \end{bmatrix} = \hat{Q}(n) \begin{bmatrix} \lambda^{1/2} \underline{R}(n-1) \\ 0 \\ \underline{X}^H(n) \end{bmatrix} \quad \dots (\text{T.3.2.2})$$

- (b) update the P-by-1 vector  $\underline{P}(n)$  using the recursion

$$\begin{bmatrix} \underline{P}(n) \\ \underline{V}(n) \end{bmatrix} = \hat{Q}(n) \begin{bmatrix} \lambda^{1/2} \underline{P}(n-1) \\ \lambda^{1/2} \underline{V}(n-1) \\ d^*(n) \end{bmatrix} \quad \dots (\text{T.3.2.3})$$

- (c) compute the least-square weight vector  $\underline{W}(n)$

$$\underline{W}(n) = \underline{R}^{-1}(n) \underline{P}(n) \quad \dots (\text{T.3.2.4})$$

and the minimum value of the sum of weighted squared error

$$\xi_{\min}(n) = \|\underline{V}(n)\|^2 \quad \dots (\text{T.3.2.5})$$

In this method, the update matrix,  $\hat{Q}(n)$ , which rotates the bottom row into upper triangular portion of the matrix in eqn.(3.34), is formed as the product of P Givens rotations.

$$\hat{Q}(n) = \hat{Q}_P(n) \dots \hat{Q}_1(n) \quad \dots (3.36)$$

Each rotation matrix is of the form

$$\hat{Q}_i(n) = \begin{bmatrix} I_{i-1} & & & \\ & C_i & & S_i^* \\ & & I_{n-i-1} & \\ & -S_i & & C_i \end{bmatrix} \quad \dots (3.37)$$

where

$$C_i = \text{Cos } \theta_i(n)$$

$$S_i = \text{Sin } \theta_i(n) e^{j\beta}$$

When  $\hat{Q}(n)$  is applied to eqn.(3.34), the proper selection of rotation angles  $\{\theta_i(n)\}$  will annihilate the non zero elements in the last row of the partially triangularized matrix

$$\begin{bmatrix} \lambda^{1/2} R(n-1) \\ \hline 0 \\ \hline \underline{X}^H(n) \end{bmatrix}$$

For illustrating the procedure, let us consider the application of  $\hat{Q}(n)$  to eqn.(3.32), which becomes

$$\hat{Q}(n) \bar{Q}(n-1) \lambda^{1/2}(n) A(n) = \hat{Q}_1(n) \begin{bmatrix} \lambda^{1/2} R(n-1) \\ \hline 0 \\ \hline \underline{X}^H(n) \end{bmatrix} \quad \dots (3.38)$$

using (3.37) in (3.38), we get

$$\begin{bmatrix} C_1 & S_1^* \\ & I_{n-2} \\ -S_1 & C_1 \end{bmatrix} \begin{bmatrix} \lambda^{1/2} R_{11}^{(n-1)} & \lambda^{1/2} R_{12}^{(n-1)} & \dots & \lambda^{1/2} R_{1P}^{(n-1)} \\ & \lambda^{1/2} R_{22}^{(n-1)} & \dots & \lambda^{1/2} R_{2P}^{(n-1)} \\ & & \dots & \dots \\ & & & \lambda^{1/2} R_{PP}^{(n-1)} \\ 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & 0 \\ x_1^*(n) & & x_2^*(n) & x_P^*(n) \end{bmatrix} \dots (3.39)$$

We denote the last row of the partially triangularized matrix by  $g_1^{(1)}(n)$ .

$$g_1^{(1)}(n) = [x_1^*(n), x_2^*(n), \dots, x_P^*(n)] \dots (3.40)$$

The element  $x_1^*(n)$  on the lower left corner of the product will become zero, if

$$x_1^*(n) C_1 - \lambda^{1/2} R_{11}^{(n-1)} S_1 = 0 \dots (3.41)$$

For eqn.(3.41) to hold, the real and imaginary parts of the expression on the left hand side must individually equal zero. Thus, recognizing that

$$|S_1|^2 + C_1^2 = 1 \dots (3.42)$$

and solving eqn.(3.41) for the parameters of the transformation, we get

$$C_1 = \frac{|\lambda^{1/2} R_{11}^{(n-1)}|}{\sqrt{[|x_1(n)|^2 + \lambda |R_{11}^{(n-1)}|^2]}} \quad \dots (3.43a)$$

and

$$S_1 = \frac{x_1^*(n)}{\sqrt{[|x_1(n)|^2 + \lambda |R_{11}^{(n-1)}|^2]}} \quad \dots (3.43b)$$

The Givens rotation as described above operates on the first and  $n^{\text{th}}$  rows of matrix  $R(n)$  to annihilate the first element  $x_1^*(n)$  in the last row. In this process, both the first and last rows of the matrix  $R(n)$  are modified. In particular, the element  $\lambda^{1/2} R_{11}^{(n-1)}$  is replaced by the new value

$$C_1 \lambda^{1/2} R_{11}^{(n-1)} + S_1^* x_1^*(n) = \frac{\lambda^{1/2} |R_{11}^{(n-1)}|}{\lambda^{1/2} |R_{11}^{(n-1)}|} \sqrt{|x_1(n)|^2 + \lambda |R_{11}^{(n-1)}|^2} \quad \dots (3.44)$$

We denote the modified last row as  $g_1^{(2)}(n)$ , which is given by

$$g_1^{(2)}(n) = [0, *, *, \dots, *] \quad \dots (3.45)$$

Where the symbol '\*' denotes the modified values. It may be noted that other elements of  $R(n)$  remain unaffected by this transformation.

We now choose  $\hat{Q}_2(n)$  to rotate the second element of the resulting bottom row,  $g_1^{(2)}(n)$ , into the (2,2) element of the upper triangular matrix and so on, until all the elements of the bottom row

are rotated into the triangular portion, thereby producing  $R(n)$ . This procedure is then repeated for the next snapshot of the data vector.

In addition, it is also easy to deduce that

$$\hat{Q}(n) \begin{bmatrix} \lambda^{1/2} \underline{P}(n-1) \\ \lambda^{1/2} \underline{V}(n-1) \\ d^*(n) \end{bmatrix} = \begin{bmatrix} \underline{P}(n) \\ \underline{V}(n) \end{bmatrix}. \quad \dots (3.46)$$

Thus,  $\underline{P}(n)$  can be updated using the same sequence of Givens rotations. The least-square weight vector  $\underline{W}(n)$  is then obtained by solving eqn. (3.25).

### 3.3-5 Direct Extraction of Residuals

In adaptive beamforming applications, the main objective is to compute the least-squares error residual since the corresponding weight vector is not of direct interest. Previous work by McWhirter [62] has described a modified version of the Q-R recursive least-square algorithm in which the least-squares residual is produced directly at every stage of the recursive process, without any need to derive the weight vector explicitly.

This technique may be summarized as follows.

We can rewrite eqn. (3.24) as

$$\hat{Q}(n) \lambda^{1/2}(n) \underline{\epsilon}(n) = \begin{bmatrix} \underline{P}(n) \\ \lambda^{1/2} \underline{V}(n-1) \\ \gamma(n) \end{bmatrix} - \begin{bmatrix} R(n) \\ 0 \end{bmatrix} \underline{W}(n) \quad \dots (3.47)$$

where  $\gamma(n)$  is the last element of the vector  $\underline{V}(n)$ .



The weight vector must satisfy the eqn.(3.25). Hence, the residual vector  $\underline{e}(n)$  is given by

$$Q(n) \Lambda^{1/2}(n) \underline{e}(n) = \hat{Q}(n) \bar{Q}(n-1) \lambda^{1/2}(n) \begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(n) \end{bmatrix} = \begin{bmatrix} 0 \\ \hline \lambda^{1/2} \underline{v}(n-1) \\ \hline \gamma(n) \end{bmatrix} \quad \dots (3.48)$$

But  $\hat{Q}(n)$  is unitary, and so we have

$$\bar{Q}(n-1) \Lambda^{1/2}(n) \begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(n) \end{bmatrix} = \hat{Q}^H(n) \begin{bmatrix} 0 \\ \hline \lambda^{1/2} \underline{v}(n-1) \\ \hline \gamma(n) \end{bmatrix} \quad \dots (3.49)$$

Considering only the  $n^{\text{th}}$  element of the vector in (3.49), it is possible to deduce that the current residual  $e(n)$  is given by

$$e(n) = \alpha(n) \gamma(n) \quad \dots (3.50)$$

where

$$\alpha(n) = \prod_{i=1}^P C_i \quad \dots (3.51)$$

is the product of all the cosine parameters generated during the sequence of Givens rotations. Eqn.(3.49) follows from the fact that  $\hat{Q}(n)$  is simply the product of 'P' elementary rotations given in eqn.(3.32). The parameter  $\alpha(n)$  may be readily computed during the recursive update of the matrix  $R(n)$  while the scalar quantity  $\gamma(n)$  is available as a direct by product of the corresponding update for

vector  $\underline{P}(n)$ . The current residual  $e(n)$  may, therefore, be evaluated in a very cost effective manner.

In the foregoing, the QRD-LS algorithm based on Givens rotation has been derived on the assumption that the Hermitian transposed data vector  $\underline{X}^H(n)$  and  $d^*(n)$  are available. Since, in real time applications,  $\underline{X}(n)$  and  $d(n)$  are available, the algorithm summarized in Table-3.3 has been written under this assumption.

**Table-3.3**  
**The QRD-LS Algorithm Using Givens Rotations**

Input definitions		
$\alpha_1 = 1.0$	$x_{P+1}(n) = d(n)$	(T 3.3.1)
$q_j^{(1)} = x_j(n),$	$j = 1, 2, \dots, P+1$	(T 3.3.2)
Algorithm	Complexity	
For $n = 1, 2, \dots, do$		
For $i = 1, 2, \dots, P+1$		
$r_{ii}(n) = \sqrt{\left[ \left  \lambda^{1/2} r_{ii}(n-1) \right ^2 + \left  q_j^{(i)}(n) \right ^2 \right]}$	3(P+1)	(T 3.3.3)
If $\left  r_{ii}(n) \right  > 0,$	$C_i = \left  r_{ii}(n-1) \right  / r_{ii}(n)$	(T 3.3.4)
	$S_i = q_j^{(i)}(n) / r_{ii}(n)$	(T 3.3.5)
	else $C_i = 1, \quad S_i = 0$	
$\alpha_{i+1}(n) = \alpha_i C_i$	(P+1)	(T 3.3.6)
For $j = i+1, \dots, P+1 do$		
$r_{ij}(n) = C_i \lambda^{1/2} r_{ij}(n-1) + S_i^* q_j^{(i)}(n)$	3P(P+1)/2	(T 3.3.7)
$q_j^{(i+1)}(n) = -S_i r_{ij}(n-1) \lambda^{1/2} + C_i q_j^{(i)}(n)$	2P(P+1)/2	(T 3.3.8)
Total 2(P+1) divisions+(P+1) square root operations)+2.5P +4.5(P+1) <sup>2</sup>		

As seen in Table 3.3, the QRD-LS algorithm using Givens rotations has a computational complexity of  $(2.5P^2+4.5(P+1))$ . In addition to this, it needs  $2(P+1)$  divisions and  $(P+1)$  square-root operations per time sample. Since, the square-root operations are computationally expensive, a square-root free Givens rotations has been proposed in the literature [34]. Though the computational complexity in this case, is reduced to  $P^2+7P$  operations per time sample, the method suffers from overflow problem.

### 3.3-6 Systolic Array Implementation

Fig.3.1 shows the systolic array structure for implementing the recursive QRD-LS algorithm described in Table-3.2. The systolic array operates directly on the input data that are represented by the matrix  $A^H(n)$  and the desired response vector  $\underline{b}^H(n)$ . Accordingly, the output of the array is  $\underline{w}^H(n)$ .

The systolic array structure consists of two distinct sections : a triangular systolic array and a linear systolic array[22]. The entire structure is controlled by a single clock. Each section of the array consists of two types of cells : internal cells (represented by circles) and boundary cells (represented by two concentric circles). Each cell receives its input data from the directions indicated for one clock cycle, performs specified arithmetic functions and then on the next cycle, delivers the resulting output values to the neighboring cells as indicated in Fig.3.1. The triangular systolic array section implements the Givens rotations part of the QRD-LS algorithm and the linear array section computes the Hermitian transposed weight vector,  $\underline{w}^H(n)$ , at the end of

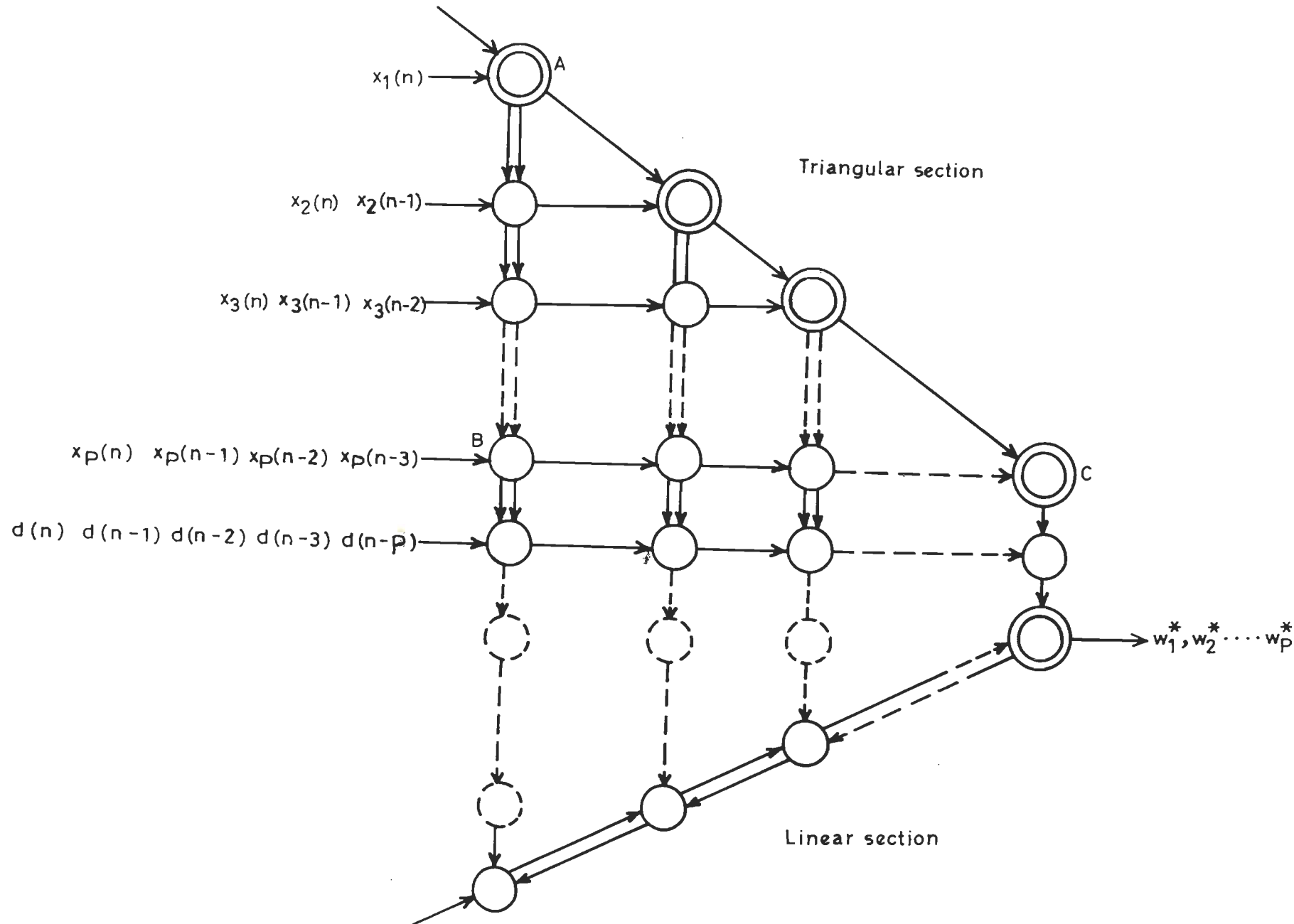


Fig.3.1. Systolic array implementation of the recursive QRD-LS algorithm using Givens rotations.

the recursions.

Consider first the operation of the triangular systolic array labeled ABC in Fig.3.1. The boundary cells and internal cells of this section are shown in Fig.3.2. The internal cells perform only the additions and multiplications while the boundary cells perform square-root and reciprocal operations as shown in the Fig.3.2(b). These operations follow directly from the discussion presented in the previous subsection. Each cell of the triangular systolic array section stores a particular element of the upper triangular matrix  $R^H(n)$  which, at the outset of the adaptive beamforming problem, is initialized to zero. The function of each column of processing cells in the triangular array section is to rotate one column of the stored triangular matrix with a vector of data received, in such a way that the leading element of the received data vector is annihilated. The reduced data vector is then passed to the right to the next column of cells. The boundary cell in each column of this section computes the pertinent rotation parameters and then passes them downward to the next clock cycle. The internal cells, subsequently, apply the same rotation to all other elements in the received data vector. Since a delay of one clock cycle per cell is incurred in passing the rotation parameters downward along a column, it is necessary that the input data vectors enter the triangular systolic array in a skewed order, as illustrated in Fig.3.1. This arrangement ensures that as each column vector  $\underline{X}(n)$  of the data matrix  $A^H(n)$  propagates through the array, it interacts with the previously stored triangular matrix  $R^H(n-1)$  and, thereby, undergoes the sequence of Givens rotations  $\hat{Q}(n)$  as required. Accordingly, all the elements of the column vector  $\underline{X}(n)$  are

annihilated, one by one, and an updated lower triangular matrix  $R^H(n)$  is produced and stored in the process.

As the orthogonal triangularization process is being performed by the triangular section labeled ABC, at the same time, the vector  $\underline{p}^H(n)$  is computed by the appended bottom row of internal cells. In effect, this computation is made by treating the desired response vector  $\underline{p}^H(n)$  as an additional row that is appended to the data matrix  $A^H(n)$ .

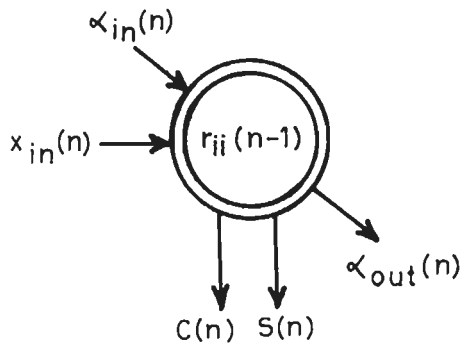
When the entire orthogonal triangularization process is complete, each particular row of the lower triangular matrix  $R^H(n)$ , or the associated 1-by-P vector  $\underline{p}^H(n)$ , is clocked out to the linear systolic array for subsequent processing. This section computes the Hermitian transposed least-square weight vector,  $\underline{w}^H(n)$ , by using the backward substitution method for solving the triangular system of equations,

$$\begin{bmatrix} r_{11}(n) & r_{12}(n) & \dots & r_{1P}(n) \\ & r_{22}(n) & \dots & r_{2P}(n) \\ & & \dots & \dots \\ & & & r_{PP}(n) \end{bmatrix} \begin{bmatrix} w_1^*(n) \\ w_2^*(n) \\ \vdots \\ w_P^*(n) \end{bmatrix} = \begin{bmatrix} P_1^*(n) \\ P_2^*(n) \\ \vdots \\ P_P^*(n) \end{bmatrix} \quad \dots (3.52)$$

In particular, the elements of the vector  $\underline{w}^H(n)$  are computed using the equations[22]

$$\begin{aligned} Z_i^{(P)} &= 0 \\ Z_i^{(k-1)} &= Z_i^{(k)} + r_{ik}(n)w_k^*(n) \end{aligned} \quad \dots (3.53)$$

$$w_i^*(n) = \frac{P_i^*(n) - Z_i^{(1)}}{r_{ii}(n)}$$



(a)

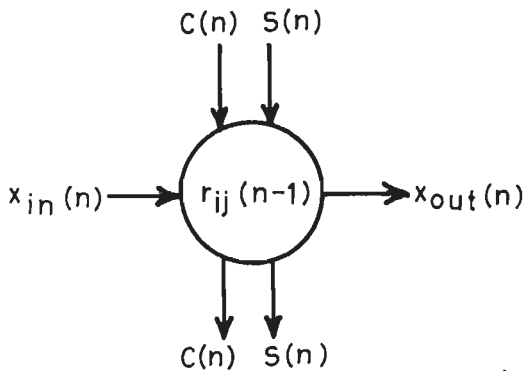
If  $x_{in} = 0$   
 $C(n) \leftarrow 1$   
 $S(n) \leftarrow 0$

$$r_{ii}(n) = \sqrt{(|x_{in}(n)|^2 + |r_{ii}(n-1)|^2)}$$

$$C(n) = r_{ii}(n-1) / r_{ii}(n)$$

$$S(n) = x_{in}(n) / r_{ii}(n)$$

$$\alpha_{out}(n) = \alpha_{in}(n) \cdot C(n)$$



(b)

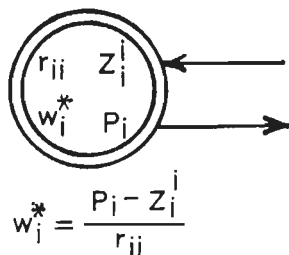
$$x_{out}(n) = -S(n) r_{ij}(n-1) + C(n) x_{in}(n)$$

$$r_{ij}(n) = C(n) r_{ij}(n-1) + S^*(n) x_{in}(n)$$

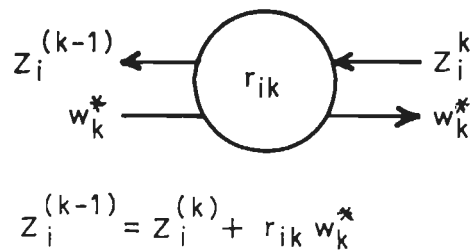
$$C(n) = C(n)$$

$$S(n) = S(n)$$

Fig.3.2. Cells for the recursive QRD-LS algorithm :  
 (a) boundary cell, (b) internal cell .



(a)



(b)

Fig.3.3. Cells for linear systolic array: (a) boundary cell, (b) internal cell .

Where  $z_i^{(k)}$  are the intermediate variables,  $r_{ik}(n)$  are elements of the upper triangular matrix  $R^H(n)$ ,  $P_i^*(n)$  are the elements of vector  $\underline{P}^H(n)$ , and  $w_k^*(n)$  are the elements of the weight vector  $\underline{W}^H(n)$ . The linear systolic array section consists of one boundary cell and  $(P-1)$  internal cells. The arithmetic functions performed by these cells are defined in Fig. 3.3 and are in accordance with eqn.(3.53). The elements of the weight vector  $\underline{W}^H(n)$  appear at the output of the boundary cell at different clock cycles, with  $w_P^*(n)$  leaving the cell first, followed by  $w_{P-1}^*(n)$  and so on.

### 3.4 THE RECURSIVE MODIFIED GRAM-SCHMIDT [RMGS] ALGORITHM

As discussed in the previous section, the QRD-LS algorithm based on Givens rotations is computationally expensive. Alternatively, QRD-LS problem can also be solved using the Gram-Schmidt procedure, which, however, has very poor numerical characteristics. A rearrangement of steps of the Gram-Schmidt procedure, known as Modified Gram-Schmidt [MGS] procedure yields a method which is computationally sound [29]. However, the MGS procedure is designed for block processing and is not efficient when it is implemented in time recursive form. Ling et al [36] have presented a time recursive form of the MGS transformations, viz, the recursive modified Gram-Schmidt [RMGS] algorithm and its error feedback form, for least-square estimation. Since their derivation is for real valued data while the adaptive beamforming problem involves complex valued data, we derive here the complex recursive modified Gram-Schmidt algorithm.



### 3.4-1 The Modified Gram-Schmidt Procedure

The data matrix defined in eqn.(3.14) can be rewritten as

$$A(n) = \begin{bmatrix} x_1^*(1) & x_2^*(1) \dots \dots x_P^*(1) \\ x_1^*(2) & x_2^*(2) \dots \dots x_P^*(2) \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ x_1^*(n) & x_2^*(n) \dots \dots x_P^*(n) \end{bmatrix} \quad \dots (3.54)$$

Premultiplying by the square-root of the exponential weighting matrix  $\lambda(n)$  defined in eqn.(3.18), we get

$$\begin{aligned} \tilde{A}(n) &= \lambda^{1/2}(n)A(n) \\ &= \begin{bmatrix} \lambda^{(n-1)/2} x_1^*(1) & \lambda^{(n-1)/2} x_2^*(1) \dots \dots \lambda^{(n-1)/2} x_P^*(1) \\ \lambda^{(n-2)/2} x_1^*(2) & \lambda^{(n-2)/2} x_2^*(2) \dots \dots \lambda^{(n-2)/2} x_P^*(2) \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ x_1^*(n) & x_2^*(n) \dots \dots x_P^*(n) \end{bmatrix} \quad \dots (3.55) \end{aligned}$$

Next, we define a set of P vectors of dimension n, as

$$\underline{x}_i(n) = \left[ \lambda^{(n-1)/2} x_i^*(1), \lambda^{(n-2)/2} x_i^*(2), \dots, x_i^*(n) \right]^T \quad i = 1, 2, \dots, P \quad \dots (3.56)$$

Using eqn.(3.56), we can express  $\tilde{A}(n)$  as

$$\tilde{A}(n) = \left[ \underline{x}_1(n), \underline{x}_2(n), \dots, \underline{x}_P(n) \right] \quad \dots (3.57)$$

where  $\underline{x}_1(n), \dots, \underline{x}_p(n)$  are linearly independent vectors.

Next, the n-by-1 desired response vector in eqn.(3.16) is rewritten as

$$\underline{b}(n) = \left[ d^*(1), d^*(2), \dots, d^*(n) \right]^T \quad \dots (3.58)$$

Premultiplying  $\underline{b}(n)$  by  $\lambda^{1/2}(n)$ , we obtain

$$\underline{B}(n) = \lambda^{1/2}(n) \underline{b}(n) = \left[ \lambda^{(n-1)/2} d^*(1), \lambda^{(n-2)/2} d^*(2), \dots, d^*(n) \right]^T \quad (3.59)$$

The error vector,  $\underline{\varepsilon}(n)$ , in eqn.(3.21) can now be written as

$$\underline{\varepsilon}(n) = \underline{B}(n) - \underline{A}(n) \underline{w}(n) \quad \dots (3.60)$$

This problem can be solved using the MGS procedure[29], where we combine the exponentially weighted data matrix  $\underline{A}(n)$  and the n-by-1 desired response vector  $\underline{B}(n)$  to form an augmented matrix.

$$\bar{\underline{A}}(n) = \left[ \underline{A}(n), \underline{B}(n) \right] = \left[ \underline{x}_1(n), \underline{x}_2(n), \dots, \underline{x}_p(n), \underline{B}(n) \right] \quad \dots (3.61)$$

Applying the MGS procedure to the matrix  $\bar{\underline{A}}(n)$ , we get

$$\bar{\underline{A}}(n) = \underline{Q}(n) \bar{\underline{K}}(n) \quad \dots (3.62)$$

Where  $\underline{Q}(n)$  is an n-by-P orthogonal matrix and  $\bar{\underline{K}}(n)$  is a (P+1)-by-(P+1) upper triangular matrix defined respectively, as

$$\underline{Q}(n) = \left[ \underline{q}_1(n), \underline{q}_2(n), \dots, \underline{q}_p(n), \bar{\underline{e}}(n) \right] \quad \dots (3.63)$$

and

$$\bar{K}(n) = \begin{bmatrix} 1 & K_{12}(n) \dots\dots K_{1P}(n) & K_1^d(n) \\ & 1 & \dots\dots K_{2P}(n) & K_2^d(n) \\ & & \dots\dots\dots\dots\dots\dots & \\ & & & 1 & K_P^d(n) \\ & & & & 1 \end{bmatrix} \dots (3.64)$$

The vectors  $\{g_1(n), \dots, g_P(n)\}$  and  $\bar{e}(n)$  in eqn. (3.63) are a set of (P+1) mutually orthogonal vectors spanning the same (P+1) dimensional space as the vectors  $\{X_1(n), X_2(n), \dots, X_P(n)\}$  and  $\underline{B}(n)$  of the augmented matrix  $\bar{A}(n)$  in eqn. (3.61). The  $n^{\text{th}}$  element of the vector  $\bar{e}(n)$  is, in fact,  $e(n)$  in eqn. (3.15) [36].

The upper triangular matrix with unit diagonal elements  $\bar{K}(n)$  in eqn. (3.62) can be written as

$$\bar{K}(n) = \begin{bmatrix} K(n) & \underline{K}^d(n) \\ \underline{0}^T & 1 \end{bmatrix} \dots (3.65)$$

The elements  $g_i(n)$ ,  $\bar{e}(n)$  and the elements of upper triangular matrix  $K_{ij}(n)$  and  $K_i^d(n)$  are determined using the modified Gram-Schmidt algorithm given in Table 3.4 [36].

The weight vector  $\underline{W}(n)$  is obtained by solving

$$K(n) \underline{W}(n) = \underline{K}^d(n). \dots (3.66)$$

Since the MGS algorithm is a block processing scheme, the vectors  $X_i(n)$  and  $\underline{B}(n)$ ,  $n = 1$  through  $N$ , are involved in the

Table-3.4

## Modified Gram-Schmidt [MGS] Algorithm.

Initialization

$$\underline{q}_i^{(1)}(n) = \underline{X}_i(n), \quad i = 1, \dots, P \quad (T 3.4.1)$$

$$\underline{e}_i^{(1)}(n) = \underline{B}(n), \quad (T 3.4.2)$$

For  $i = 1$  to  $P$  do

$$\underline{q}_i(n) = \underline{q}_i^{(1)}(n) \quad (T 3.4.3)$$

$$r_{ii}(n) = \underline{q}_i^H(n) \underline{q}_i(n) \quad (T 3.4.4)$$

For  $j = i + 1$  to  $P$  do

$$r_{ij}(n) = \left[ \underline{q}_j^{(i)}(n) \right]^H \underline{q}_i(n) \quad (T 3.4.5)$$

$$K_{ij}(n) = r_{ij}(n) / r_{ii}(n) \quad (T 3.4.6)$$

$$\underline{q}_j^{(i+1)}(n) = \underline{q}_j^{(i)}(n) - K_{ij}(n) \underline{q}_i(n) \quad (T 3.4.7)$$

$$r_i^d(n) = \left[ \underline{e}_i^{(1)}(n) \right]^H \underline{q}_i(n) \quad (T 3.4.8)$$

$$K_i^d(n) = r_i^d(n) / r_{ii}(n) \quad (T 3.4.9)$$

$$\underline{e}_i^{(i+1)}(n) = \underline{e}_i^{(1)}(n) - K_i^d(n) \underline{q}_i(n) \quad (T 3.4.10)$$

$$\underline{e}(n) = \underline{e}^{(P+1)}(n) \quad (T 3.4.11)$$

computation of error  $e(n)$ , or the weight vector  $\underline{W}(n)$ . Therefore, the computational complexity will increase as 'n' increases. Hence, the use of MGS algorithm in real time application is inefficient.

### 3.4-2 The Complex Recursive Modified Gram-Schmidt[RMGS] Algorithm

It can be easily seen from Table 3.4 that the  $n^{\text{th}}$  components of the vectors  $q_j^{(i)}(n)$ ,  $q_1(n)$  and  $\bar{e}^{(i)}(n)$  namely,  $q_1^{(i)}(n,n)$ ,  $q_1(n,n)$  and  $e^{(i)}(n,n)$ , satisfy the same order recursive equations as their corresponding vectors.

Therefore,

$$q_1(n,n) = q_1^{(1)}(n,n) \quad \dots (3.67)$$

$$q_j^{(i+1)}(n,n) = q_j^{(i)}(n,n) - K_{ij}(n) q_1(n,n) \quad \dots (3.68)$$

$$e^{(i+1)}(n,n) = e^{(i)}(n,n) - K_1^d(n) q_1(n,n) \quad \dots (3.69)$$

$$e(n) = e^{(P+1)}(n,n) \quad \dots (3.70)$$

In order to obtain the time recursive form of the complex MGS algorithm [26], have to derive only the time update formulae for the coefficients  $r_{ij}(n)$  and  $r_1^d(n)$ . These time update formulae have been derived in Appendix-B and are given by the following equations.

$$r_{ij}(n) = \lambda r_{ij}(n-1) + q_j^{(i)*}(n,n) q_1(n,n)/\alpha_1(n) \quad \dots (3.71)$$

and

$$r_1^d(n) = \lambda r_1^d(n-1) + \left[ e^{(i)}(n,n) \right]^* q_1(n,n)/\alpha_1(n) \quad \dots (3.72)$$

for  $i = 1$  to  $P$  and  $j = i+1$  to  $P$ .

In the above equations,  $\alpha_i(n)$  is a scalar quantity whose magnitude is close to, but less than unity. It is calculated using the order recursive equation which is derived in Appendix-C and given by

$$\alpha_{i+1}(n) = \alpha_i(n) - \frac{|q_i(n,n)|^2}{r_{ii}(n)} \quad \dots (3.73)$$

In the MGS algorithm (Table-3.4), the computation of  $q_i(n), q_j^{(i+1)}(n), e(n), r_{ij}(n)$  and  $r_{id}(n)$  involves vector operations. Replacing these equations by their corresponding time recursive form, we obtain the complex RMGS algorithm which is presented in Table-3.5.

The complex RMGS algorithm has a computational complexity of  $1.5P^2 + 4.5P$  per time sample. In addition, it needs  $P^2 + 3P$  divisions per time sample. Using the error feedback form of the RMGS algorithm, which is obtained by using an a priori error form [53] and incorporating the error feedback formula [38], the number of divisions is reduced to  $0.5P^2 + 1.5P$  per time sample. The error feedback form of RMGS algorithm is summarized in Table-3.6. A distinct feature of this algorithm is that the error  $e^{(i+1)}(n,n)$  and  $\alpha_i(n)$  are feedback to time update the elements  $K_{ij}(n)$ , of the upper triangular matrix and the elements of the vector  $K^d(n)$ . Therefore, the algorithm exhibits better numerical accuracy and is more robust to round off errors as compared to RMGS algorithm without error feedback [36].

Table-3.5

The Complex Recursive Modified Gram-Schmidt Algorithm

Input definitions

$$\alpha_1(n) = 1 \quad r_{11}(0) = \delta \quad (T\ 3.5.1)$$

$$q_i^{(1)}(n) = x_i(n), \quad i = 1, 2, \dots, P, \quad e^{(1)}(n, n) = d(n) \quad (T\ 3.5.2)$$

Algorithm

Complexity

For  $i = 1$  to  $P$  do

$$q_i(n, n) = q_i^{(i)}(n, n) \quad (T\ 3.5.3)$$

$$r_{ii}(n) = \lambda r_{ii}(n-1) + \frac{|q_i(n, n)|^2}{\alpha_i(n)} \quad 2P \quad (T\ 3.5.4)$$

$$\alpha_{i+1}(n) = \alpha_i(n) - \frac{|q_i(n, n)|^2}{r_{ii}(n)} \quad P \quad (T\ 3.5.5)$$

For  $j = i+1$  to  $P$  do

$$\left[ \begin{array}{l} r_{ij}(n) = \lambda r_{ij}(n-1) + q_j^{(i)}(n, n) q_i^*(n, n) / \alpha_i(n) \quad P(P-1) \quad (T\ 3.5.6) \\ K_{ij}(n) = r_{ij}(n) / r_{ii}(n) \quad (T\ 3.5.7) \\ q_j^{(i+1)}(n, n) = q_j^{(i)}(n, n) - K_{ij}(n) q_i(n, n) \quad P(P-1)/2 \quad (T\ 3.5.8) \end{array} \right.$$

$$r_i^d(n) = \lambda r_i^d(n-1) + e^{(i)}(n, n) q_i^*(n, n) / \alpha_i(n) \quad 2P \quad (T\ 3.5.9)$$

$$K_i^d(n) = r_i^d(n) / r_{ii}(n) \quad (T\ 3.5.10)$$

$$e^{(i+1)}(n, n) = e^{(i)}(n, n) - K_i^d(n) q_i(n, n) \quad P \quad (T\ 3.5.11)$$

$$e(n) = e^{(P+1)}(n, n) \quad (T\ 3.5.12)$$

Total  $(P^2 + 3P)$  divisions +  $1.5P^2 + 4.5P$

\* denotes complex conjugate operation.

Table-3.6

The complex RMGS Algorithm Using Error Feedback

Input definitions		
$\alpha_1(n) = 1,$	$r_{11}(0) = \delta$	(T 3.6.1)
$q_i^{(1)}(n,n) = x_i(n),$	$i = 1, 2, \dots, P$	$e^{(1)}(n,n) = d(n)$ (T 3.6.2)
Algorithm		Complexity
For $i = 1$ to $P$ do		
$q_i(n,n) = q_i^{(i)}(n,n)$		(T 3.6.3)
$r_{ii}(n) = \lambda r_{ii}(n-1) + \alpha_i(n)  q_i(n,n) ^2$	$3P$	(T 3.6.4)
$\alpha_{i+1}(n) = \alpha_i(n) - \frac{\alpha_i^2(n)  q_i(n,n) ^2}{r_{ii}(n)}$	$3P$	(T 3.6.5)
For $j = i+1$ to $P$ do		
$q_j^{(i+1)}(n,n) = q_j^{(i)}(n,n) - K_{ij}(n-1)q_i(n,n)$	$P(P-1)/2$	(T 3.6.6)
$K_{ij}(n) = K_{ij}(n-1) + \alpha_i(n)q_j^{(i+1)}(n,n)q_i^*(n,n)/r_{ii}(n)$		(T 3.6.7)
	$P(P-1)$	
$e^{(i+1)}(n,n) = e^{(i)}(n,n) - K_i^d(n-1)q_i(n,n)$	$P$	(T 3.6.8)
$K_i^d(n) = K_i^d(n-1) + \alpha_i(n)e^{(i+1)}(n,n)q_i^*(n,n)/r_{ii}(n)$	$2P$	(T 3.6.9)
$e(n) = e^{(i+1)}(n,n)$		(T 3.6.10)
Total	$(0.5P^2 + 1.5P)$ divisions + $1.5P^2 + 7.5P$	



### 3.4-3 Systolic Array Implementation

As in the case of QRD-LS algorithm using Givens rotations, the RMGS algorithm can be implemented using systolic array structure as shown in Fig.3.4. The arrangement of the structure shown in Fig.3.1 has been retained for ease of comparison. The systolic array operates on the data that are represented by  $A^H(n)$  and desired response vector  $\underline{b}^H(n)$ . Accordingly the output of the array is  $\underline{w}^H(n)$ .

The triangular array section labeled ABC in Fig.3.4 implements the RMGS algorithm. The boundary cells process equations of type (T 3.5.4) and (T 3.5.5), whereas, the internal cells process equations of type (T 3.5.6) and (T 3.5.8) of Table 3.5. Both boundary and internal cells of this section perform only arithmetic functions, viz , addition, subtraction, multiplication and division. The boundary cells are initiated to a small value,  $\delta$  to avoid a division by zero. Each cell of the triangular array section stores a particular element of the lower triangular matrix  $R^H(n)$  which are updated every clock cycle. The function of each column of cells is to orthogonalize the leading element of the data vector with respect to its other elements. This process of orthogonalization produces an orthogonal vector of random variables whose dimension is one less than that of input data vector and a set of constants which are stored as the elements of the upper triangular matrix  $R^H(n)$ . The orthogonal vector of random variables is then passed to the right to the next column of cells. The process repeats till the last column of cells is reached so that all the elements of the input data vector are orthogonalized with each other, thereby, decorrelating all the signals in the data vector. The boundary cell in each column of this section, that computes  $r_{11}$  and

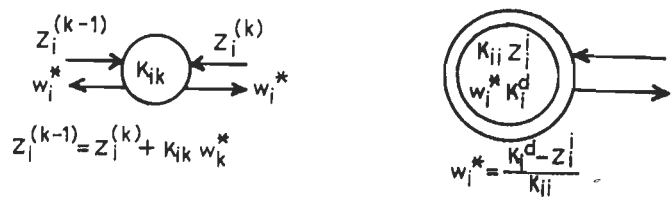
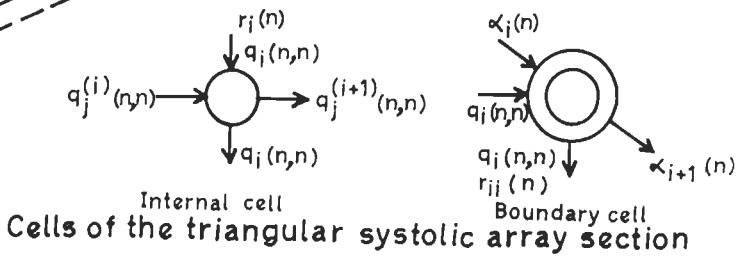
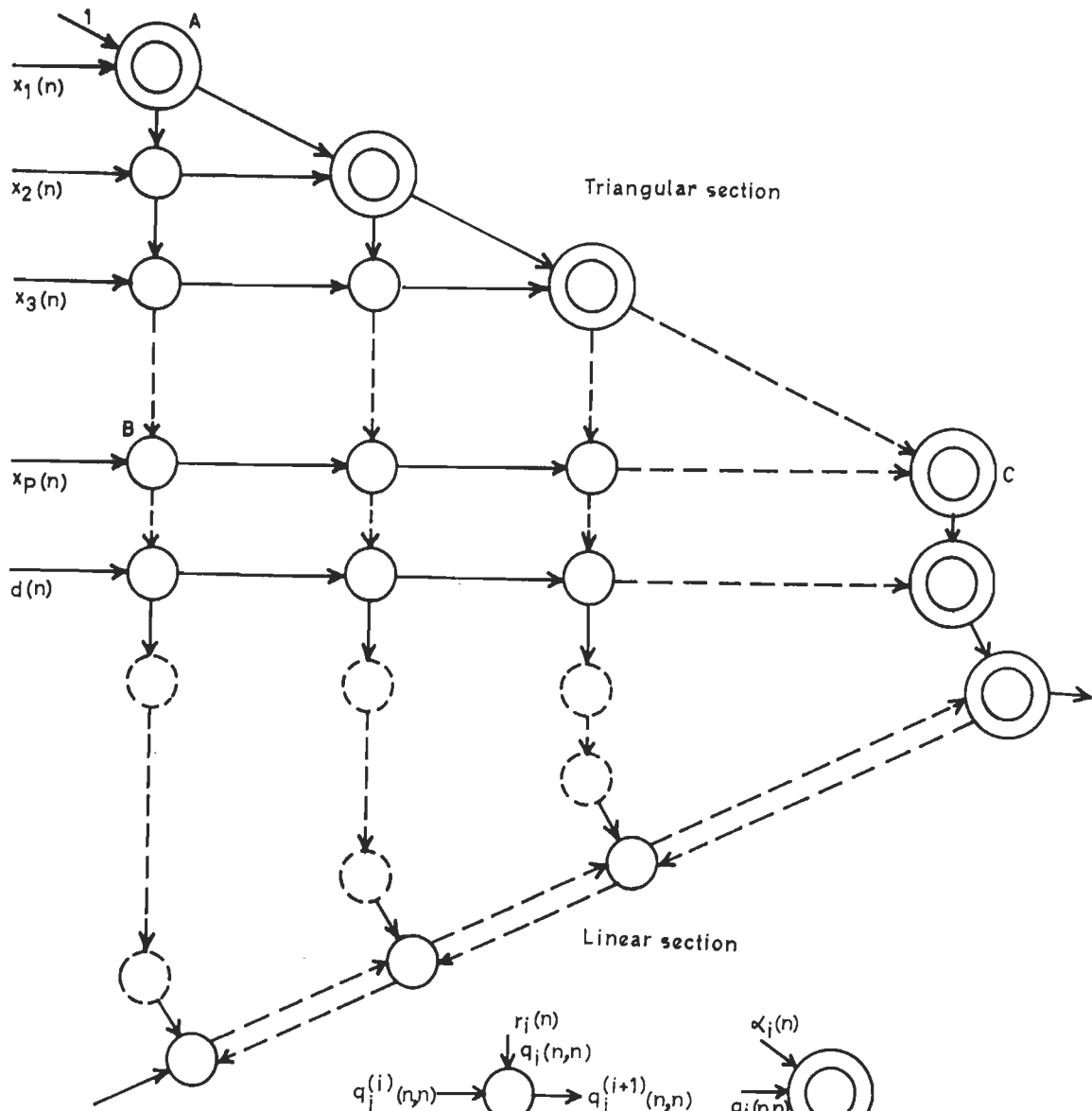


Fig.3.4. Systolic array implementation of the RMGS algorithm.

$\alpha_{i+1}$ , stores the computed  $r_{11}$  and passes  $\alpha_{i+1}$  diagonally to the next column of cells. The internal cells subsequently apply the operations given by eqns. (T 3.5.6) and (T 3.5.8) to the other elements of the data vector to compute  $r_{1j}$  and the elements for the next data vector. The elements of the upper triangular matrix are normalized with  $r_{11}$ , so that all the diagonals are equal to unity. The vector  $[\underline{K}^d(n)]^H$  is computed by treating  $d(n)$  as an appended bottom row of cells. The bottom row of the internal cells compute equations of type (T 3.5.9) and (T 3.5.11).

When the entire orthogonal triangularization process is completed, each row of the triangular matrix  $R^H(n)$  or the associated 1-by-P vector  $[\underline{K}^d(n)]^H$  is clocked out to the linear systolic array to compute the Hermitian transposed weight vector  $\underline{W}^H(n)$ .

### 3.5 BROADBAND ADAPTIVE ARRAYS

The broadband adaptive beamformer model introduced in chapter-2, is only an extension of the narrowband beamformer, in that, a tapped delay line is added behind each sensor as shown in Fig.2.4. That is, the dimensions of the signal vector and weight vector increase from 'P' to 'PM', where M is the number of taps in each delay line.

Let  $\underline{X}_{-PM}$  and  $\underline{W}_{-PM}$  denote the signal vector and the weight vector given, respectively, by

$$\underline{X}_{PM}(j) = \begin{bmatrix} x_1(j), x_2(j), \dots, x_p(j), \\ x_1(j-1), x_2(j-1), \dots, x_p(j-1), \\ \dots \\ x_1(j-M+1), x_2(j-M+1), \dots, x_p(j-M+1) \end{bmatrix}^T \quad \dots (3.74)$$

and

$$\underline{W}_{PM}(j) = \begin{bmatrix} w_{11}, w_{21}, \dots, w_{p1}, \\ w_{12}, w_{22}, \dots, w_{p2}, \\ \dots \\ w_{1M}, w_{2M}, \dots, w_{pM} \end{bmatrix}^T \quad \dots (3.75)$$

where the subscript PM denotes the dimension of the vector.

The least-square problem to be solved is to find the weight vector  $\underline{W}_{PM}$ , which minimizes the exponentially weighted sum of squared errors given by,

$$\sum_{j=1}^n \lambda^{n-j} |e_M(j)|^2 \quad \dots (3.76)$$

where  $e_M(j)$  is the error in the estimation of the desired signal  $d(j)$ , which can be written as

$$e_M(j) = d(j) - \underline{W}_{PM}^H(j) \underline{X}_{PM}(j) \quad \dots (3.77)$$

The tap weight vector  $\underline{W}_{PM}(n)$  which minimizes the quantity in eqn. (3.76), is obtained by differentiating the expression with respect to  $\underline{W}_{PM}(n)$  and equating the results to zero. That is,

$$\nabla_{\underline{W}_{PM}(n)} \left[ \sum_{j=1}^n \lambda^{n-j} |e_M(j)|^2 \right] = 0 \quad \dots (3.78)$$

The solution of the above equation is

$$\Phi_{PM}(n) \underline{W}_{PM}(n) = \underline{\theta}_{PM}(n) \quad \dots (3.79)$$

where

$$\Phi_{PM}(n) = \sum_{j=1}^n \lambda^{n-j} \underline{X}_{PM}(j) \underline{X}_{PM}^H(j) \quad \dots (3.80)$$

and

$$\underline{\theta}_{PM}(n) = \sum_{j=1}^n \lambda^{n-j} \underline{X}_{PM}(j) d^*(j) \quad \dots (3.81)$$

Therefore, the exact least-square algorithms can be applied to the broadband beamforming problem as well. However, the increase in the dimensions of the signal vector and weight vector leads to an increase in the computational complexity of these algorithms to the order of  $O(P^2M^2)$ . In particular, PM square-root operations will have to be performed per time sample, in the case of Givens rotation based QRD-LS algorithm.

### 3.6 SIMULATION RESULTS

In order to demonstrate the numerical properties, convergence characteristics and nulling abilities of the exact least-square algorithms considered in this chapter, the output residual powers and voltage patterns of the four adaptive beamforming techniques, viz, the RLS algorithm, the RMGS algorithm and its error feedback form (RMGSEF) and the QRD-LS algorithm have been compared using several computer simulated examples.

### 3.6-1 Narrowband Arrays

A 6-element uniform linear array has been assumed in the first three examples, with the signal environment consisting of a desired signal and four interferences. A larger array of twelve elements in a scenario of a desired signal and as many as ten interferences has been assumed in the fourth example.

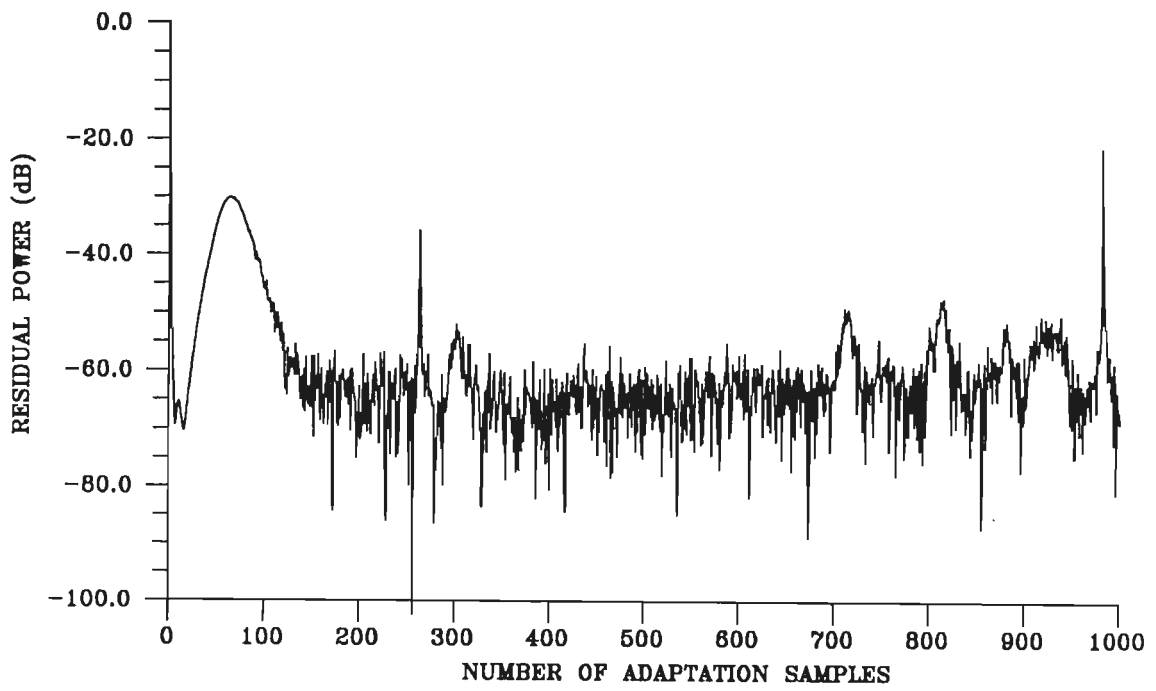
**Example 3.6-1.1** In this example, the signal environment has been modelled to have a desired signal and four interferences which are narrowband. The desired signal arrives from the direction of broadside to the array and is of strength 0.1. The interference parameters are given in Table 3.7.

**Table 3.7**

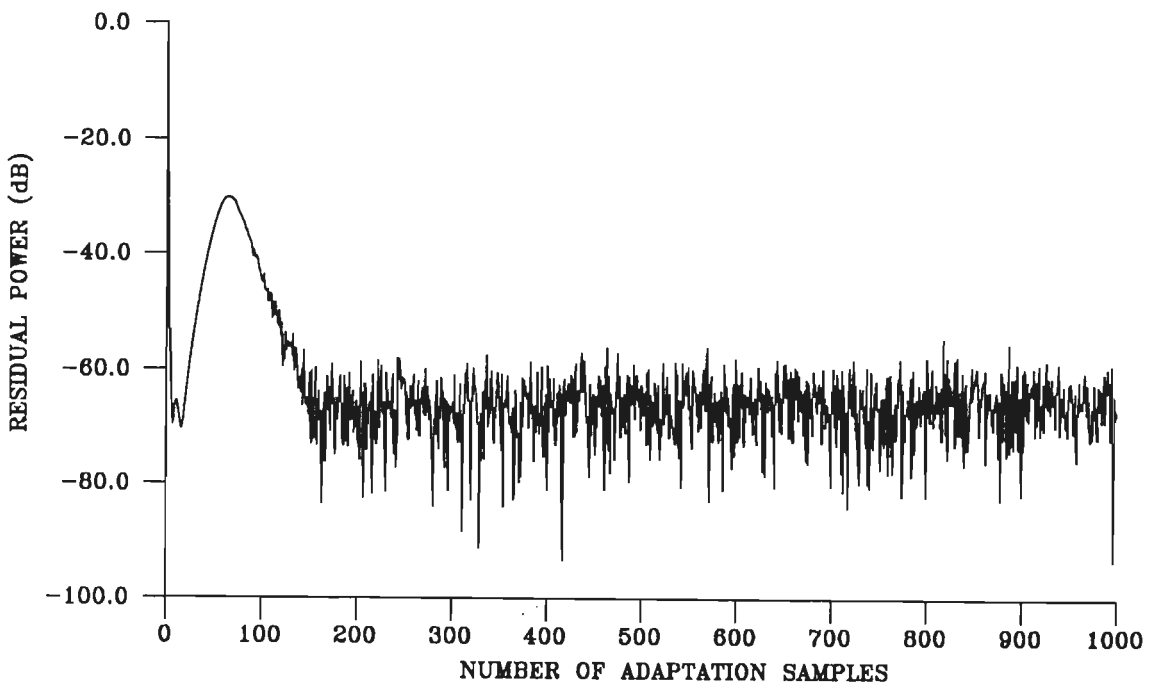
**Parameters of the Interferences in Example 3.6-1.1**

Parameter	Interference 1	Interference 2	Interference 3	Interference 4
$S_1$	10.0	10.0	10.0	10.0
$\theta_1$	$30^\circ$	$-30^\circ$	$60^\circ$	$-60^\circ$
$\omega_1$	1.1	0.9	1.2	0.8

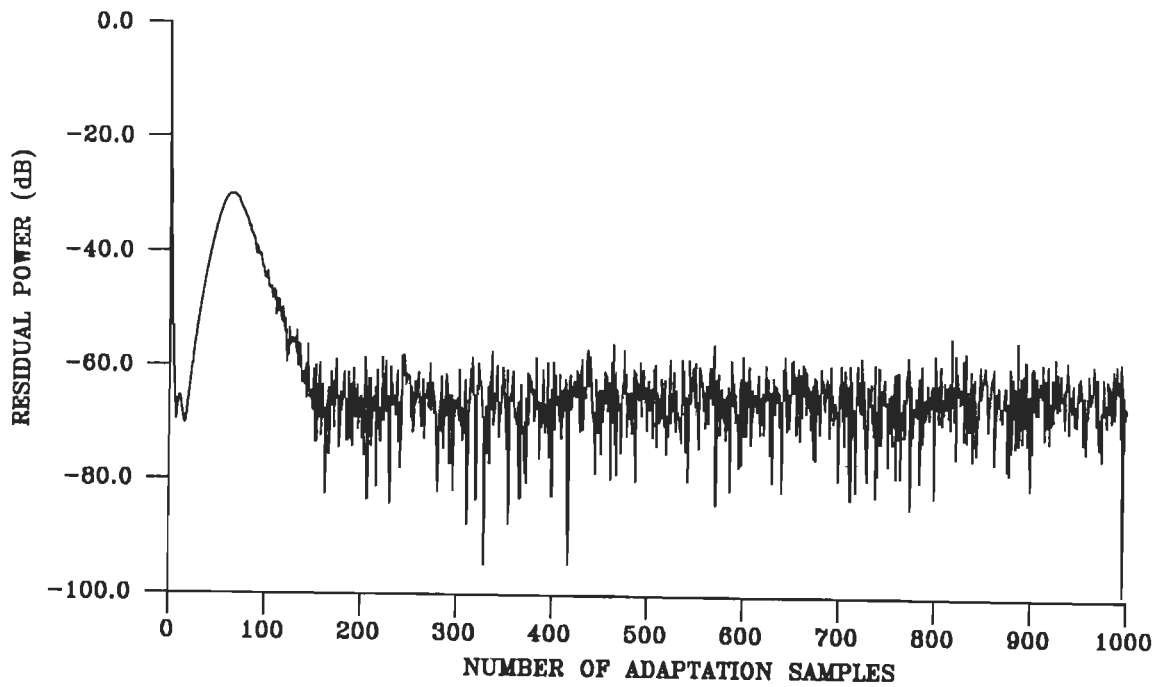
Fig.3.5 shows the residual power as a function of the number of adaptation samples for RLS, RMGS, RMGSEF and QRD-LS algorithm based beamformers. An inspection of Fig.2.3 clearly shows that in comparison to these beamformers, the LMS beamformer has a much lower convergence speed and, also, a larger residual power. Of the four beamformers considered here, the QRD-LS array is found to exhibit fastest convergence speed. It attains convergence, in the present interference environment, in about 40 iterations (Fig.3.5(d)). The other three



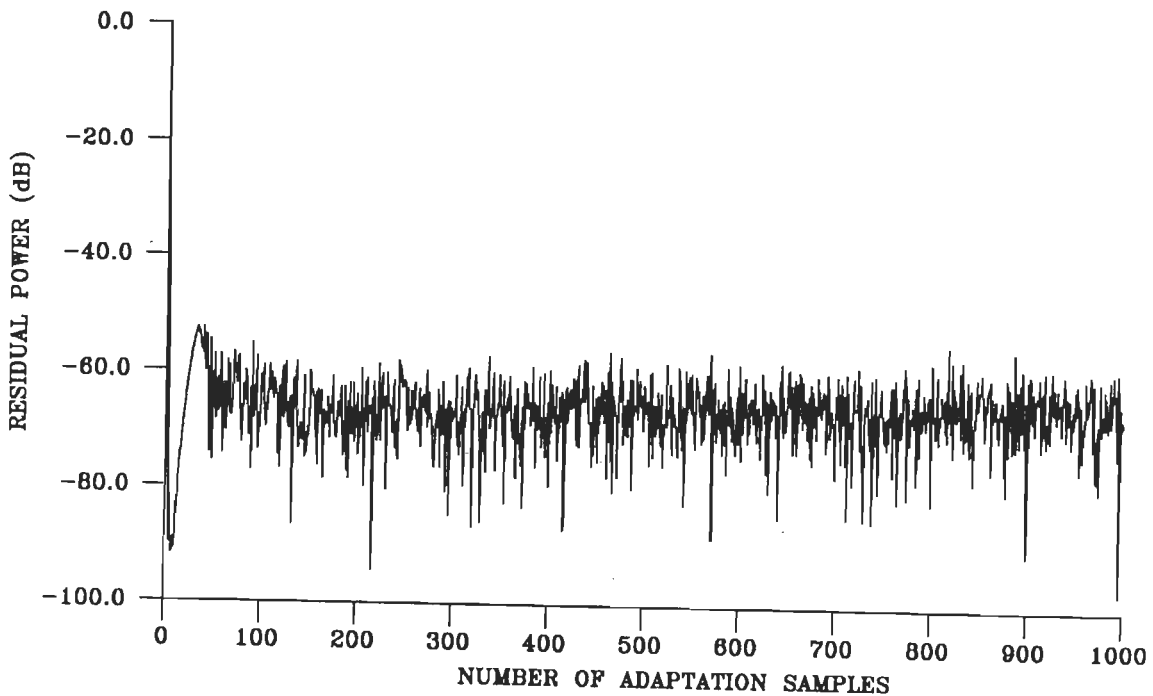
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER



(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

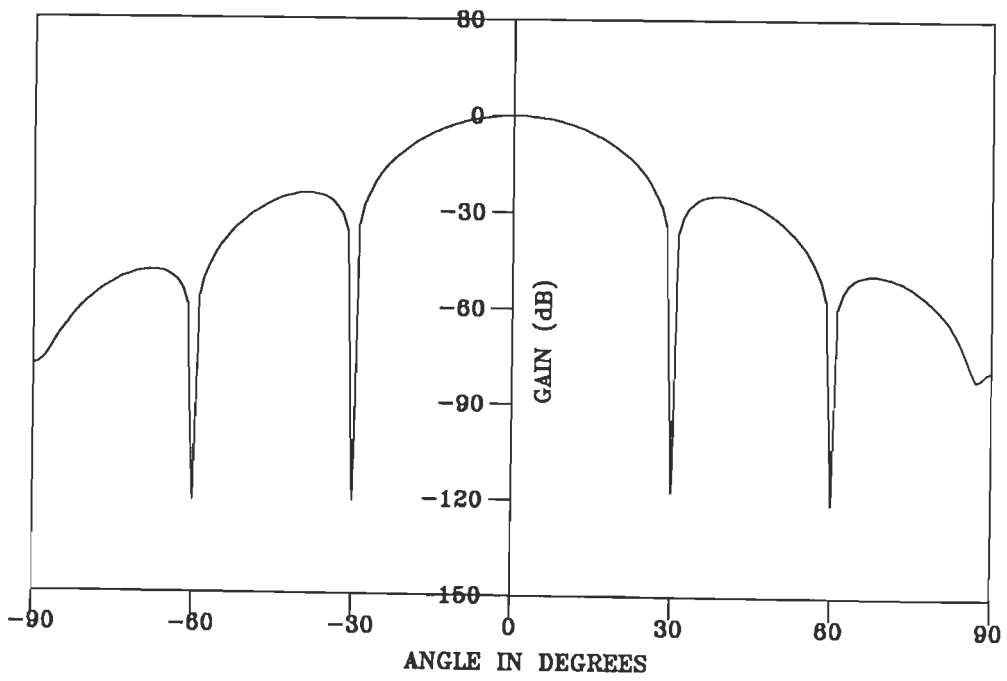
FIG.3.5 CONVERGENCE CHARACTERISTICS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ, -30^\circ, 60^\circ$  AND  $-60^\circ$ .



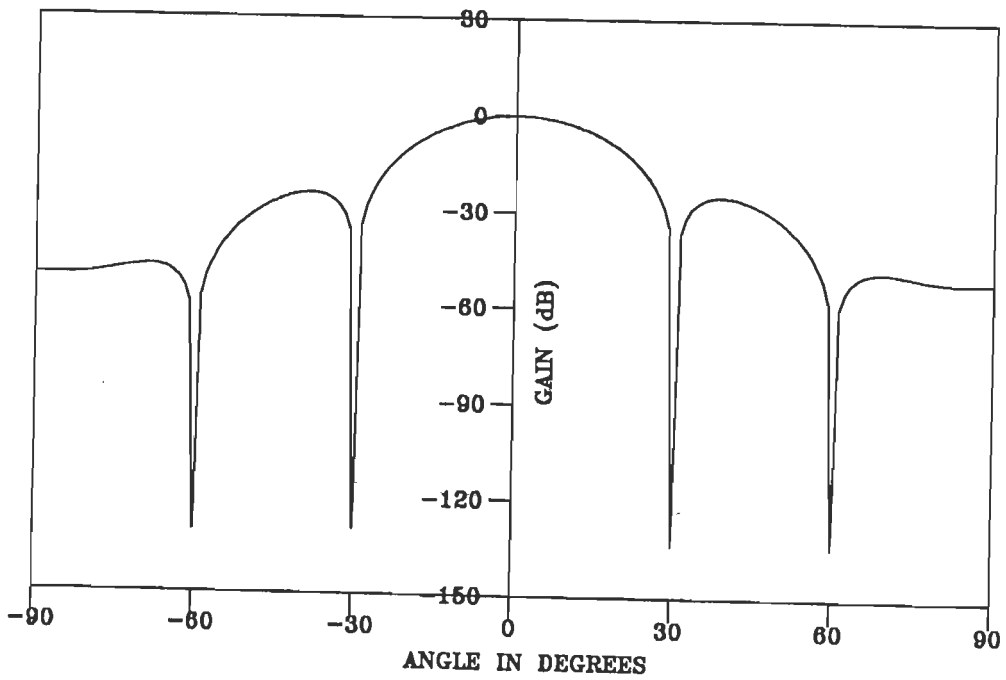
beamformers, however, take about 150 samples to converge. All these beamformers, except RLS beamformer exhibit the same residual power of -60dB after attaining convergence. In the case of RLS beamformer, however, it is found that the residual power continuously exhibits sharp fluctuations and after about 700 samples, it has an increasing trend (Fig.3.5(a)). This phenomenon may be attributed to the numerical instability of the RLS algorithm.

The weight coefficients of these beamformers at the end of 1000 samples are listed in Table.3.8. It is found that the four beamformers converge to nearly the same weight vector. This is not surprising because, all the beamformers are based upon the same exact least-square error criterion. It is also observed that the imaginary parts of the weight vector are negligibly small. This is because, the desired signal and the interferences have been assumed to be of the same form  $S_1 e^{j(\omega_1 t + \phi_1)}$ . Since the interferences have been assumed to have identical amplitudes and initial phase, and are symmetrically located, the weight vectors are expected to be real.

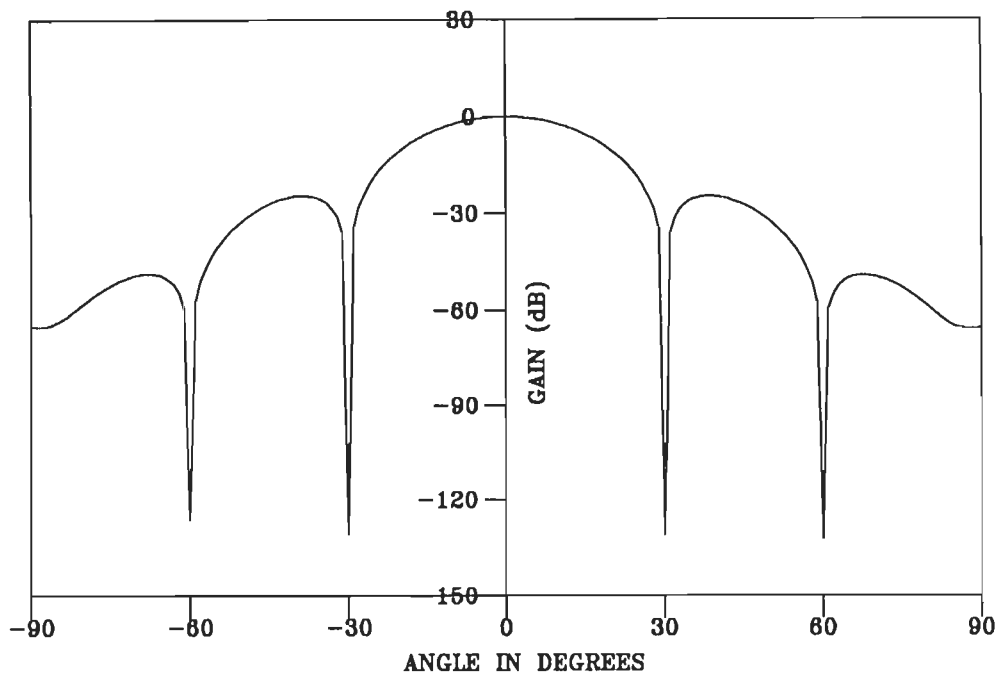
The voltage patterns of these beamformers are shown in Fig.3.6 and the null depths are listed in Table.3.9. On comparing the voltage patterns with that of the LMS beamformer (Fig.2.2), it can be seen that the exact least-square beamformers produce much deeper nulls in the direction of interferences. The RLS beamformer produces nulls which are of slightly lesser depth as compared to the other three beamformers.



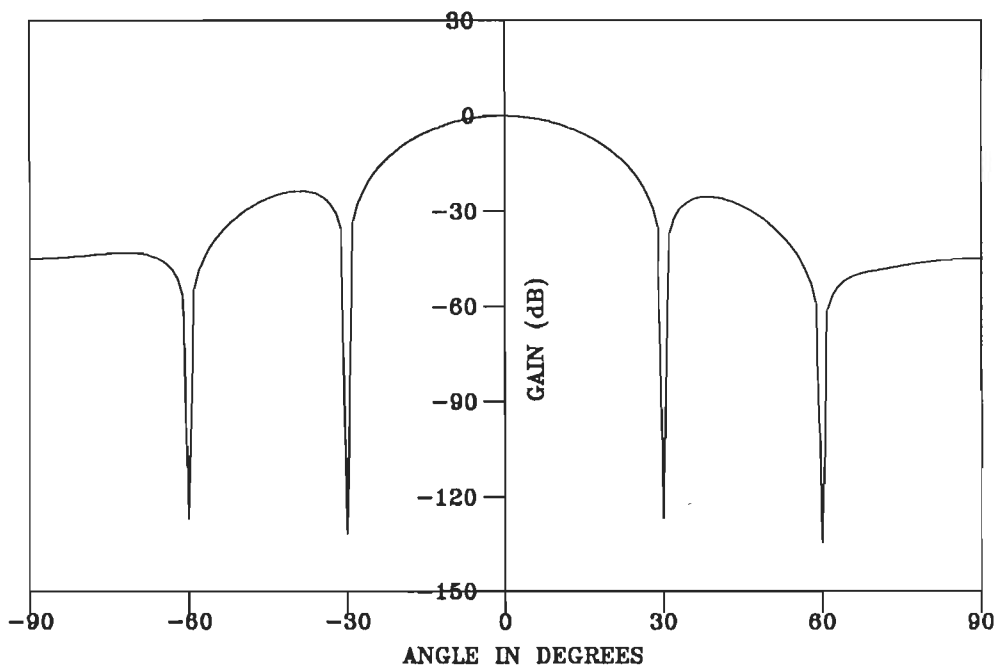
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER



(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

FIG.3.6 VOLTAGE PATTERNS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ$ ,  $-30^\circ$ ,  $60^\circ$  &  $-60^\circ$ .

**Table-3.8**

**Weight Coefficients of the Four Beamformers**

Weight Coefficients	RLS	RMGS	RMGSEF	QRD-LS
$w_1$	0.65265+j0.00006	0.69004+j0.00005	0.66073+j0.00012	0.714327+j0.00008
$w_2$	1.846827+j0.00001	1.888371+j0.00008	1.851891+j0.00019	1.92534+j0.00002
$w_3$	2.500289+j0.00002	2.508312+j0.00002	2.502833+j0.00006	2.512530+j0.00007
$w_4$	2.530421+j0.00001	2.490292+j0.00001	2.498551+j0.00006	2.42127+j0.00007
$w_5$	1.82325+j0.00001	1.813210+j0.00001	1.840683+j0.00001	1.78683+j0.00007
$w_6$	0.64430+j0.00001	0.613382+j0.00002	0.624320+j0.00018	0.578134+j0.00001

**Table-3.9**

**Null Depths Produced by the Beamformers in dB**

$\theta_1$	RLS	RMGS	RMGSEF	QRD-LS
$30^\circ$	-117	-130	-130	-129
$-30^\circ$	-120	-134	-134	-132
$60^\circ$	-120	-136	-136	-135
$-60^\circ$	-120	-131	-131	-128

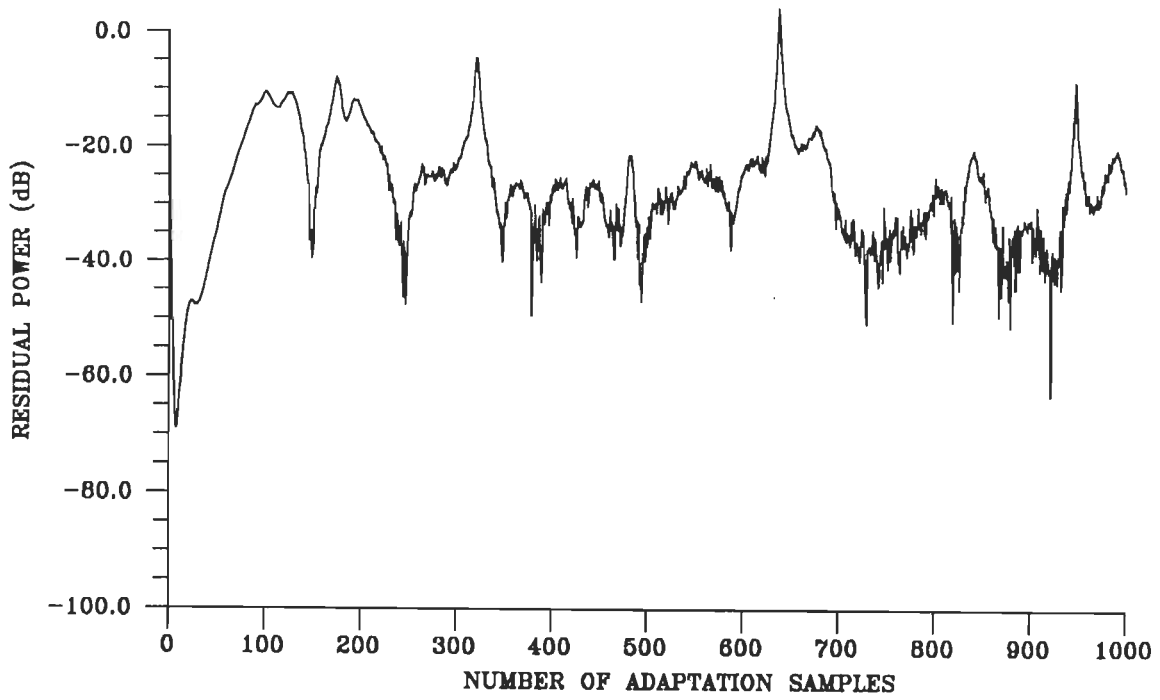
**Example 3.6-1.2** In this example, two interferences are modelled to arrive from directions very close to the desired signal ( $5^\circ$  and  $-5^\circ$ ) while the other two arrive from near endfire directions ( $85^\circ$  and  $-85^\circ$ ). The remaining parameters of the interferences are the same as

given in Table-3.7.

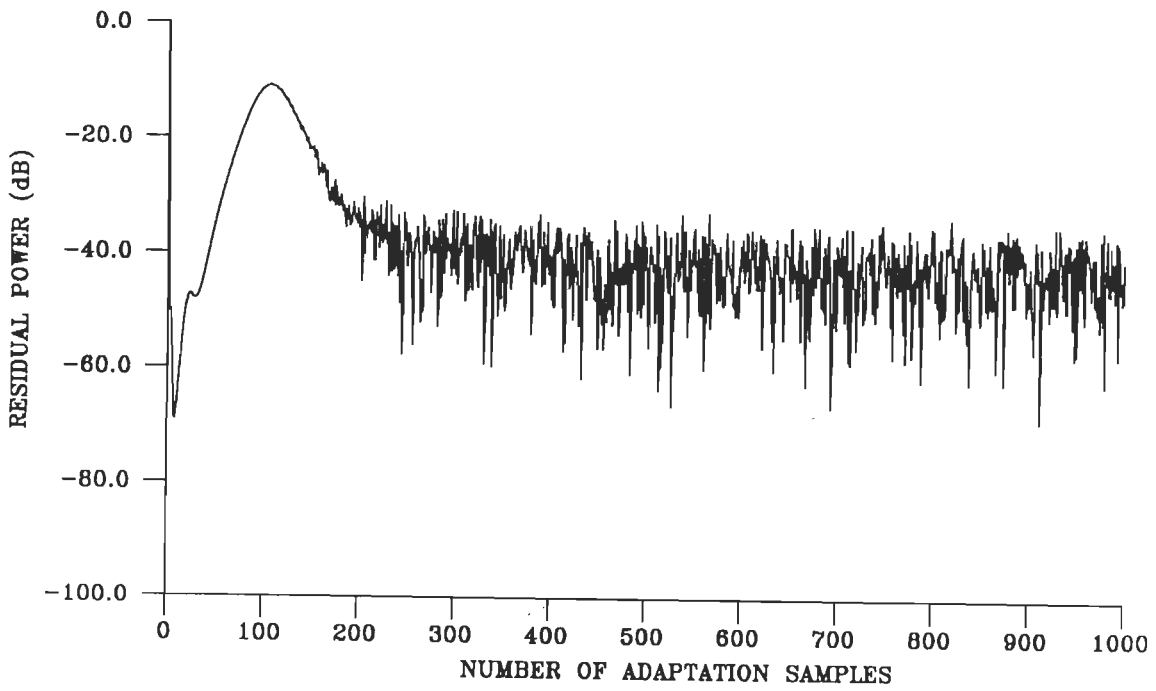
The output residual power characteristics for the four beamformers are shown in Fig.3.7. The problem of instability, which was noted in the previous example, can be seen more clearly here. The RLS beamformer (Fig.3.7(a)) does not show any signs of convergence even after 1000 samples and further, the residual power is also very large. In particular, the residual power is greater than 0dB at about 650 samples. The RMGS and RMGSEF array (Fig.3.7(b) and (c)) converge in about 200 samples and have a residual power of about -40dB after convergence is achieved. The QRD-LS beamformer (Fig.3.7(d)) exhibits much faster convergence and attains convergence in about 80 samples. A comparison of Fig.3.7 with Fig.3.5 shows that when interferences arrive from near endfire as well as near broadside directions, these beamformers require more number of samples to converge.

The corresponding voltage patterns are shown in Fig.3.8. All the four beamformers succeed in placing nulls in the direction of interferences arriving very close to the desired signal, ie, at  $+5^{\circ}$  and  $-5^{\circ}$ . However, deep nulls are obtained in the case of RMGS, RMGSEF and the QRD-LS beamformers which are on the order of -100dB. Since the two interferences are very close to the desired signal and fall within the main beam of the array, the grating lobes invariably appear.

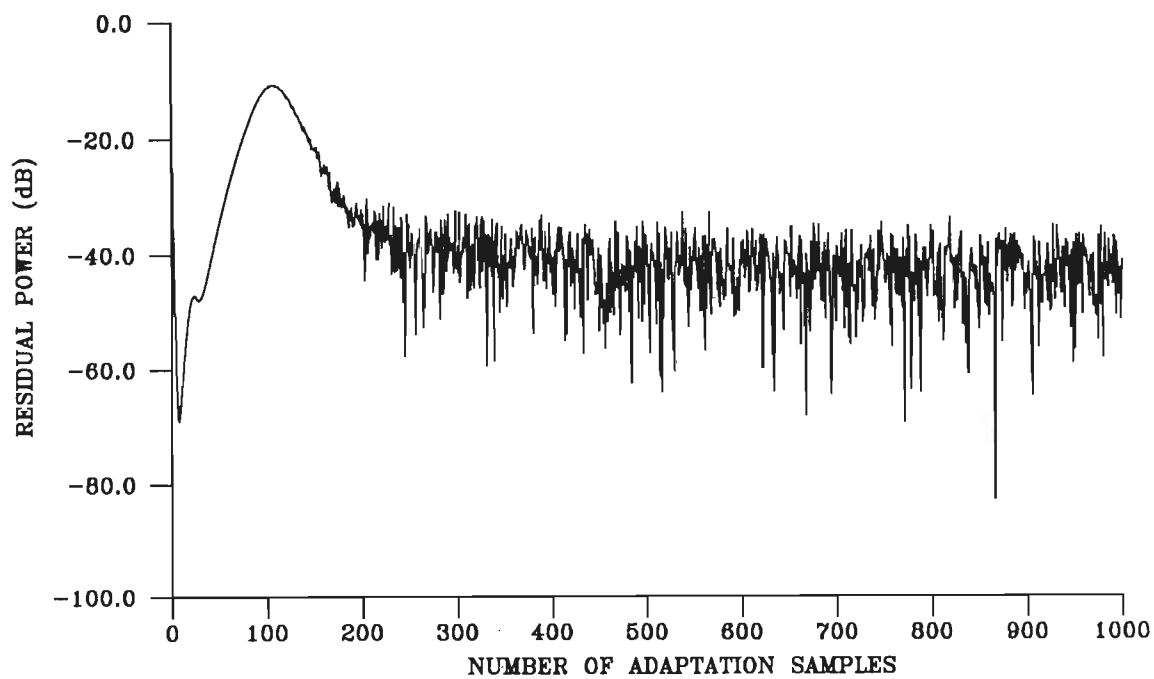
It can be seen from Fig.3.8(a), that the RLS beamformer fails to produce nulls in the direction of near endfire interferences. This is possibly because the RLS beamformer has very poor convergence characteristics for this interference environment and the residual power is also very large. On the other hand, the other three



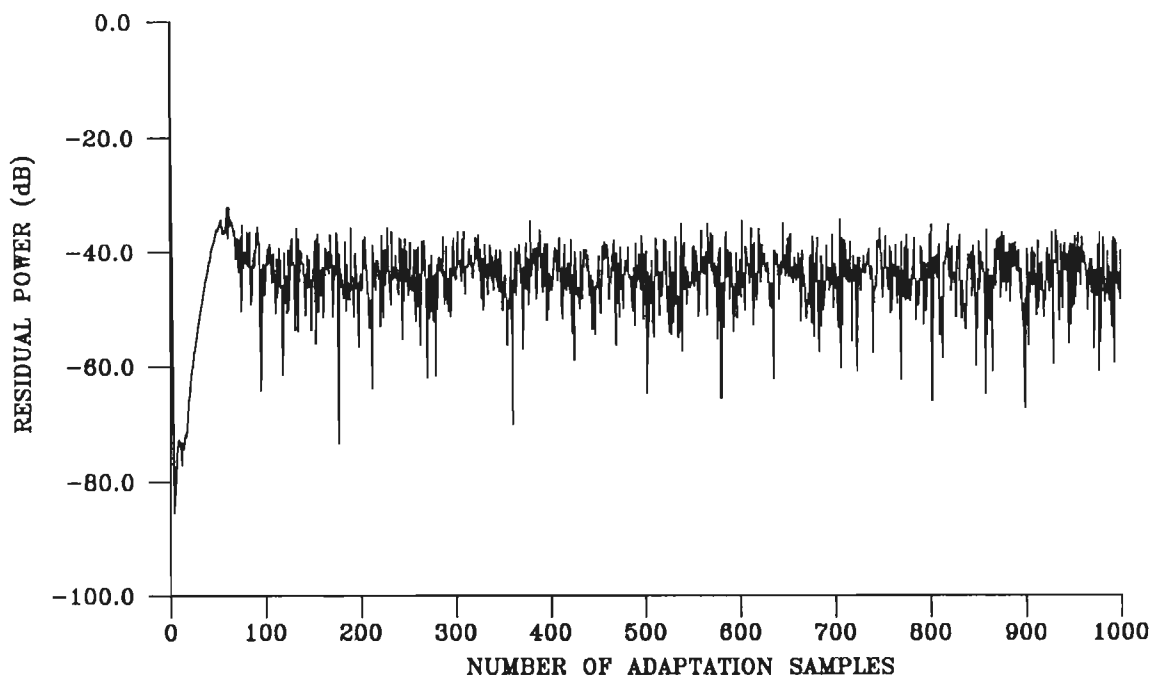
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER

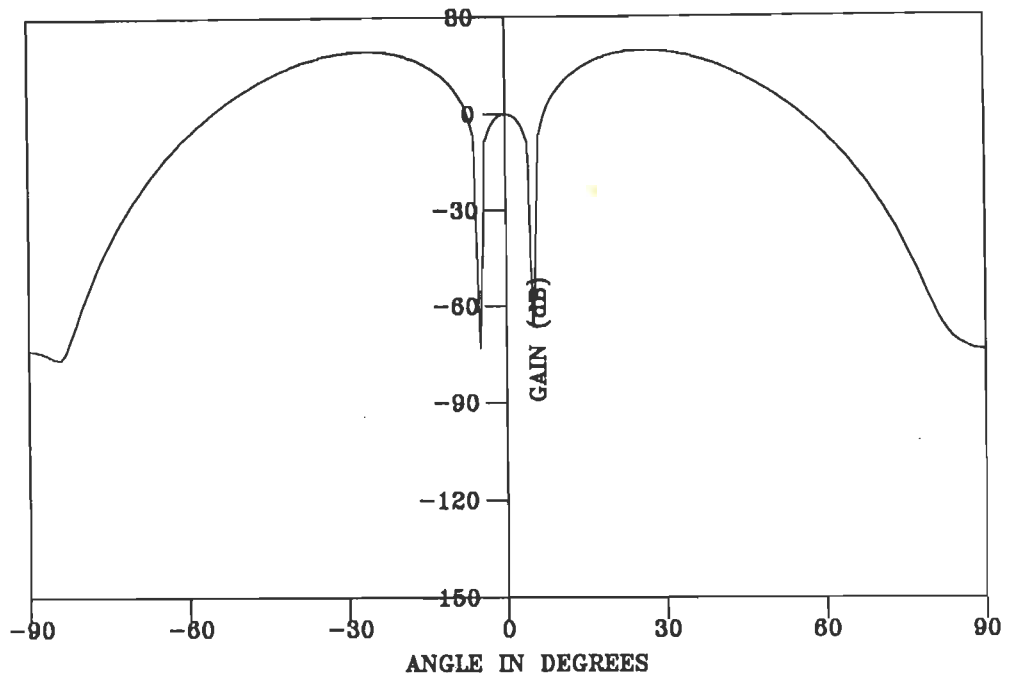


(c) RMGSEF BEAMFORMER

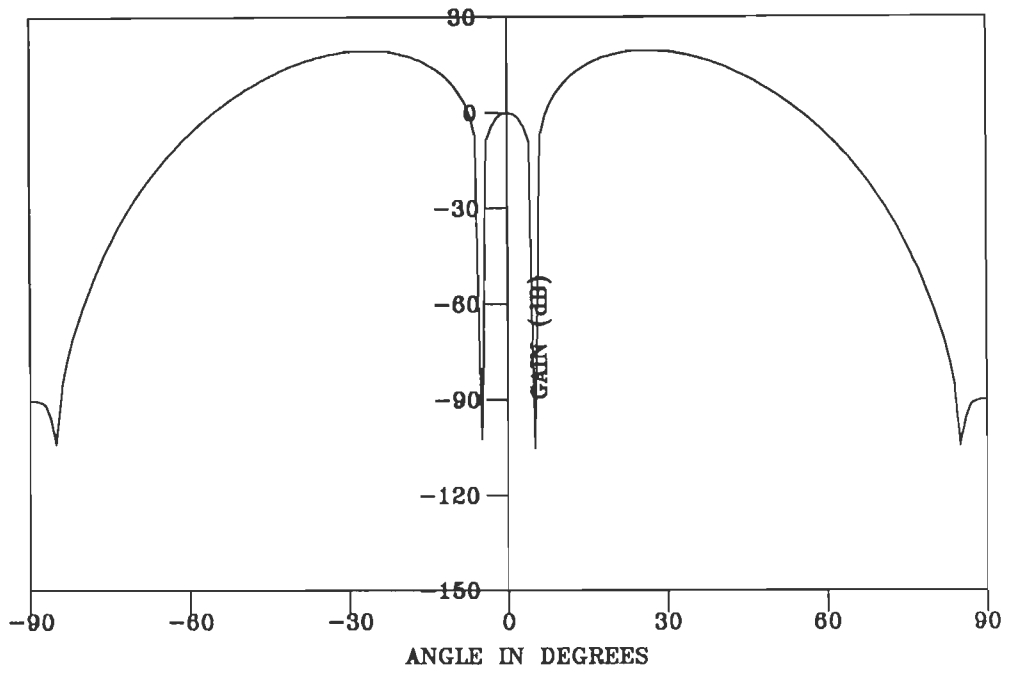


(d) QRD-LS BEAMFORMER

FIG.3.7 CONVERGENCE CHARACTERISTICS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $5^\circ, -5^\circ, 85^\circ$  AND  $-85^\circ$ .

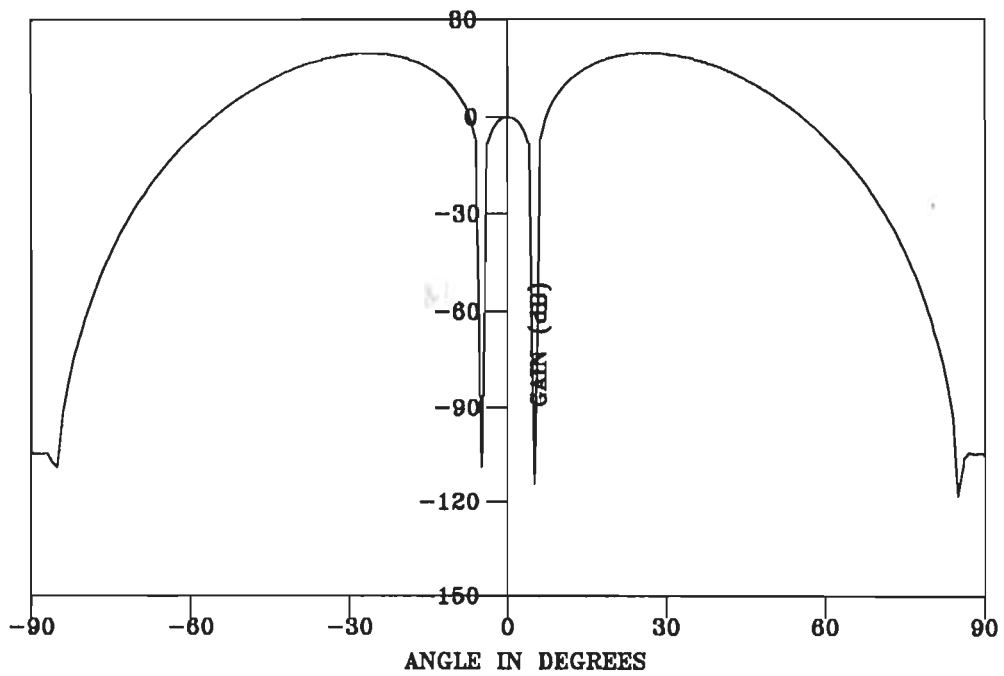


(a) RLS BEAMFORMER

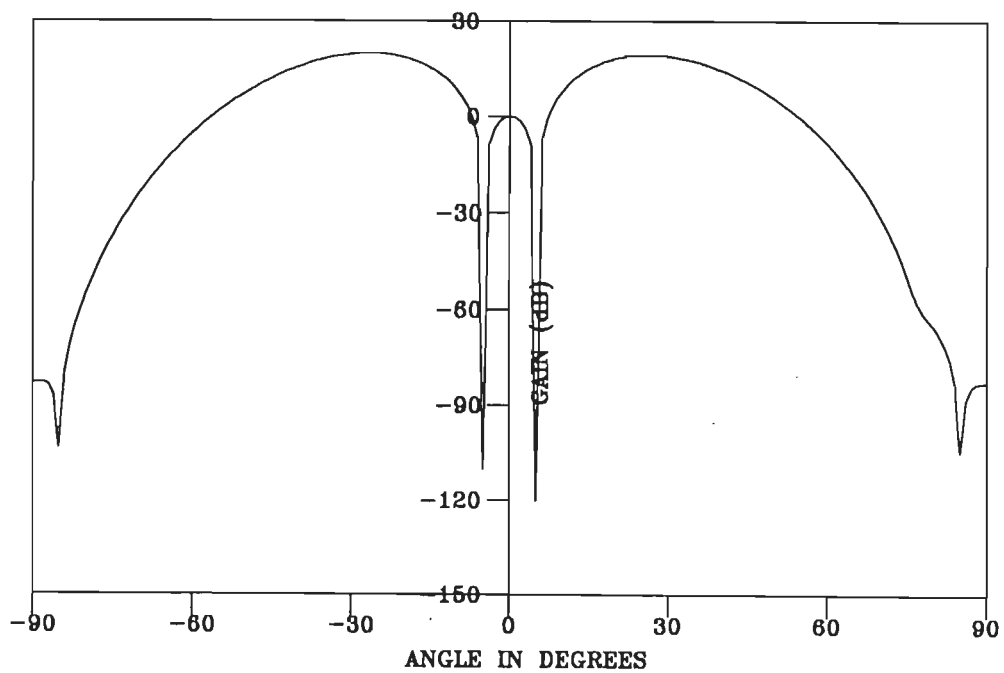


(b) RMGS BEAMFORMER





(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

FIG.3.8 VOLTAGE PATTERNS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $5^\circ, -5^\circ, 85^\circ$  &  $-85^\circ$ .

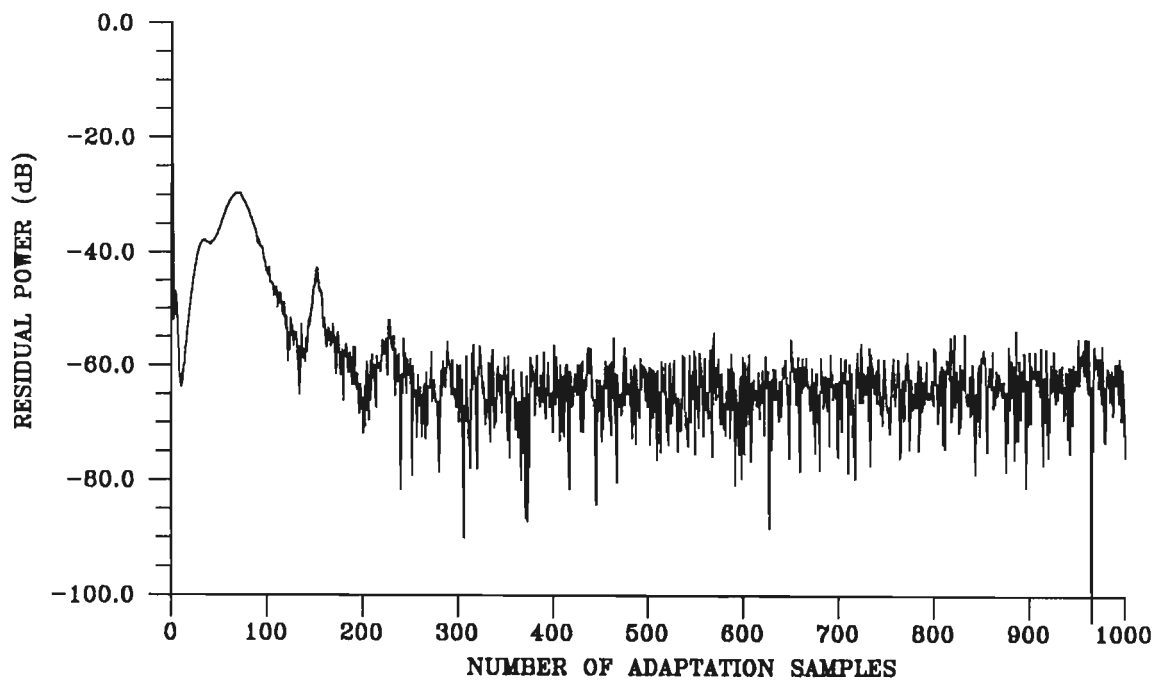
beamformers place well defined nulls of -100dB or more depth in the direction of endfire interferences.

**Example 3.6-1.3** In this example, an interference environment consisting of two near endfire interferences arriving at  $+80^\circ$  and  $-80^\circ$  has been assumed. The other two interferences have been assumed to arrive at  $+60^\circ$  and  $-60^\circ$ . The remaining parameters are same as in Table-3.7.

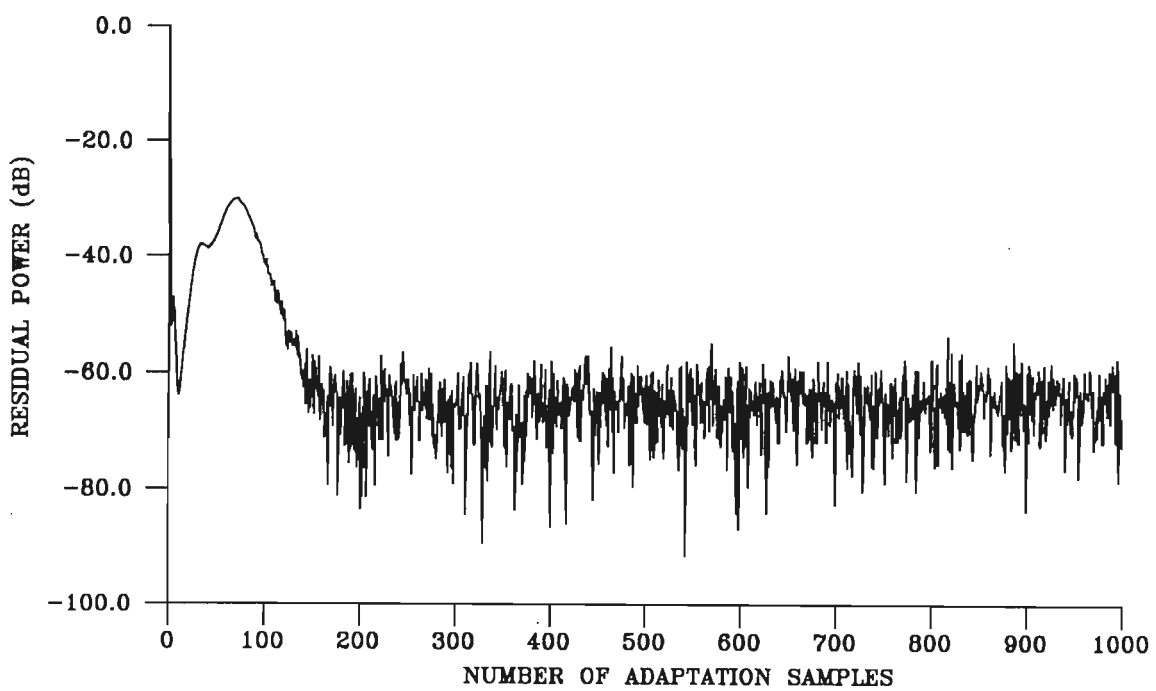
Fig.3.9 shows the residual power as a function of the number of adaptation samples for the four beamformers. The RLS beamformer has the lowest convergence speed and converges after about 250 samples while the RMGS and RMGSEF beamformers converge in about 150 samples. The QRD-LS beamformer exhibits fastest convergence and converges in about 40 samples.

From the voltage patterns of the four beamformers shown in Fig.3.10, it is found that all the beamformers place deep nulls in the direction of interferences arriving from  $+60^\circ$  and  $-60^\circ$ . However, inspite of achieving convergence, the RLS beamformer again fails to produce nulls in the direction of endfire interferences arriving from  $+80^\circ$  and  $-80^\circ$ . The other three beamformers, viz, the RMGS, RMGSEF and the QRD-LS beamformers (Fig.3.10(b)-(d)) place sharp nulls in the direction of  $+80^\circ$  and  $-80^\circ$ .

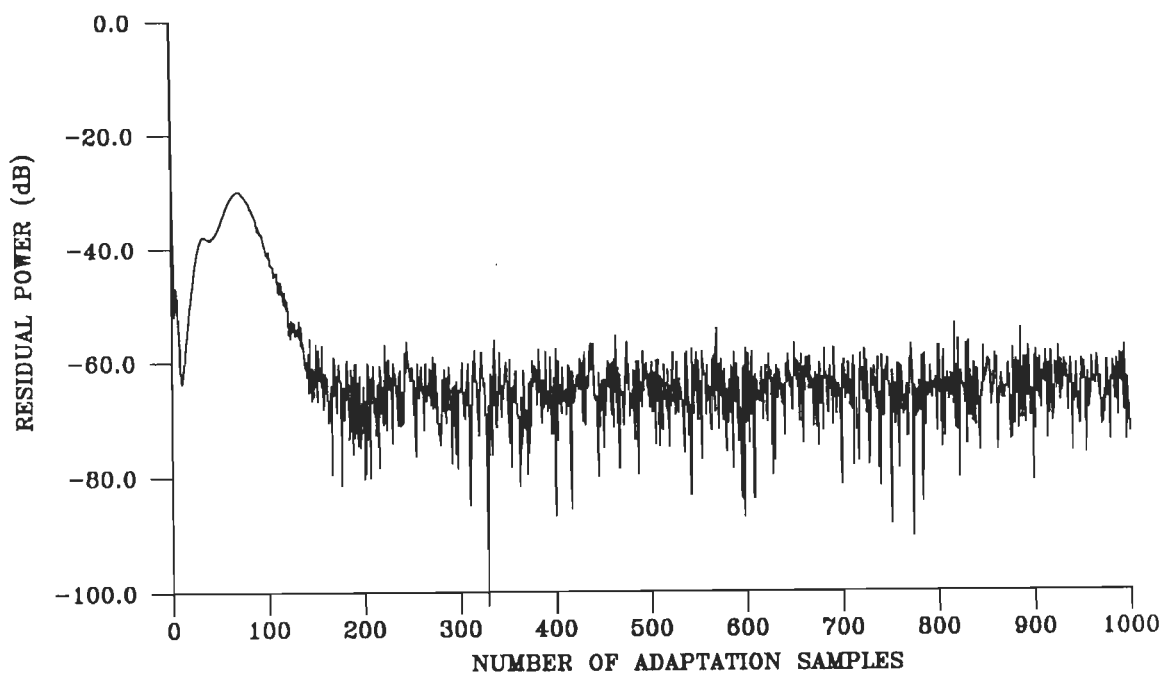
**Example 3.6-1.4** As a final example for the narrowband case, we consider a total of 10 interferences whose parameters are given in Table-3.10.



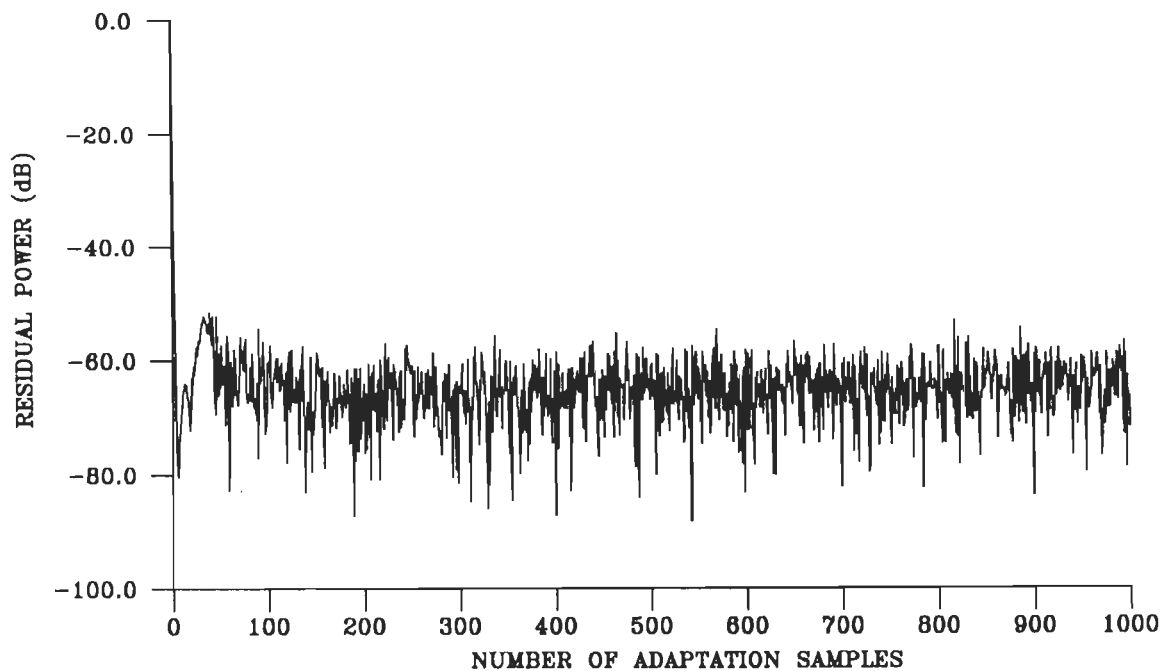
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER

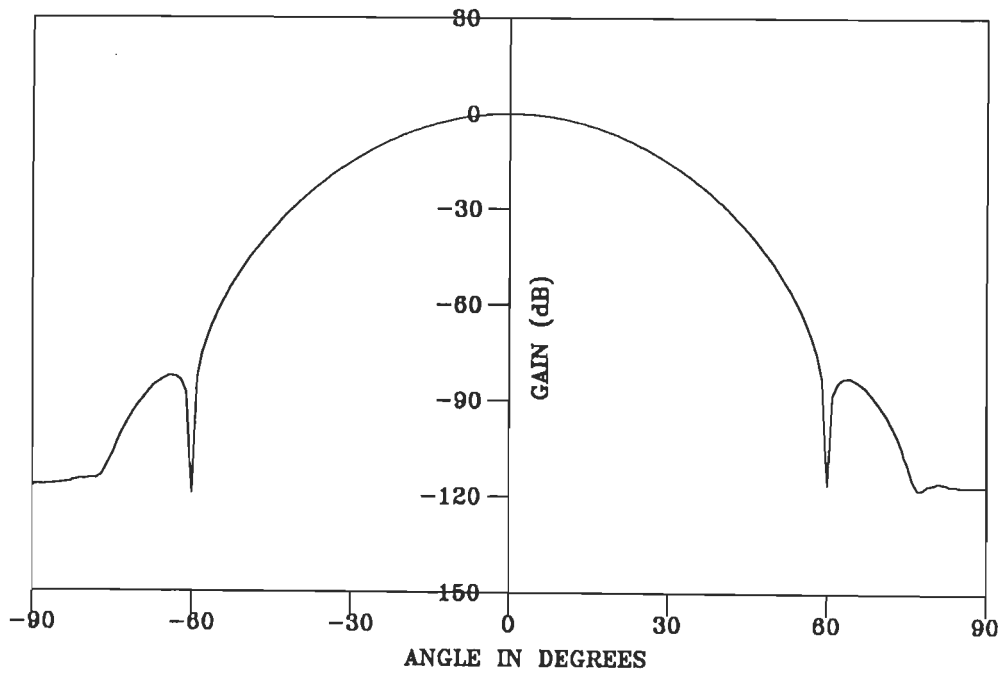


(c) RMGSEF BEAMFORMER

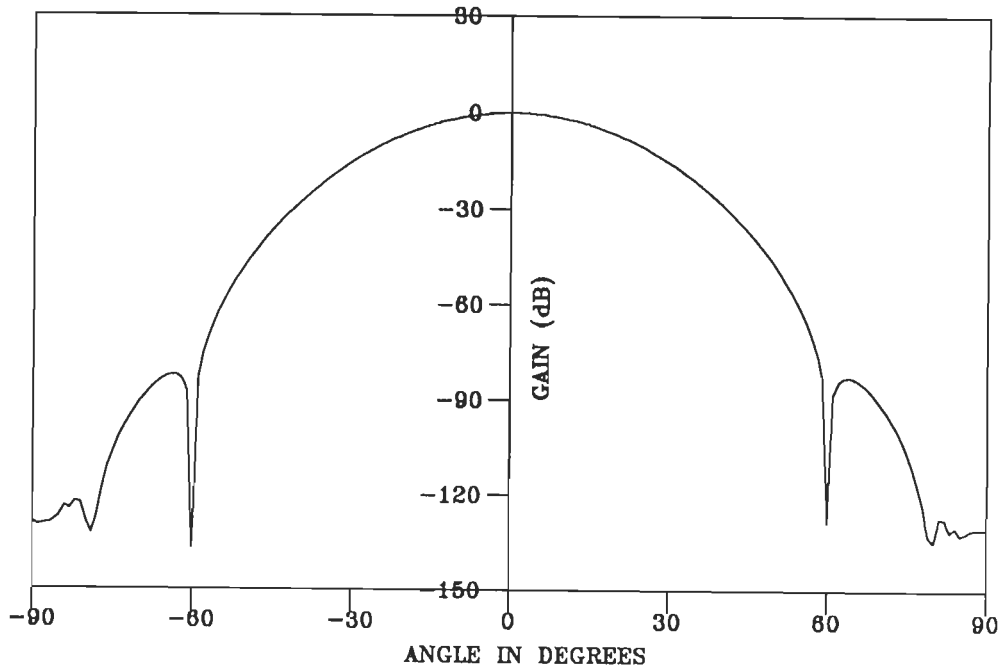


(d) QRD-LS BEAMFORMER

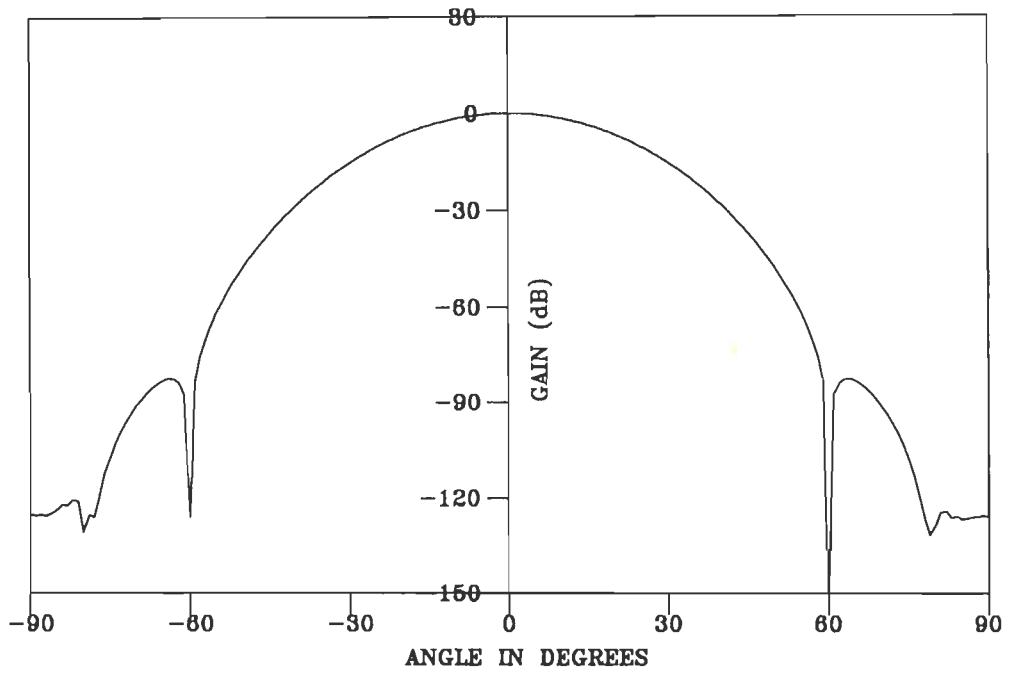
FIG.3.9 CONVERGENCE CHARACTERISTICS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $60^\circ$ ,  $-60^\circ$ ,  $80^\circ$  AND  $-80^\circ$ .



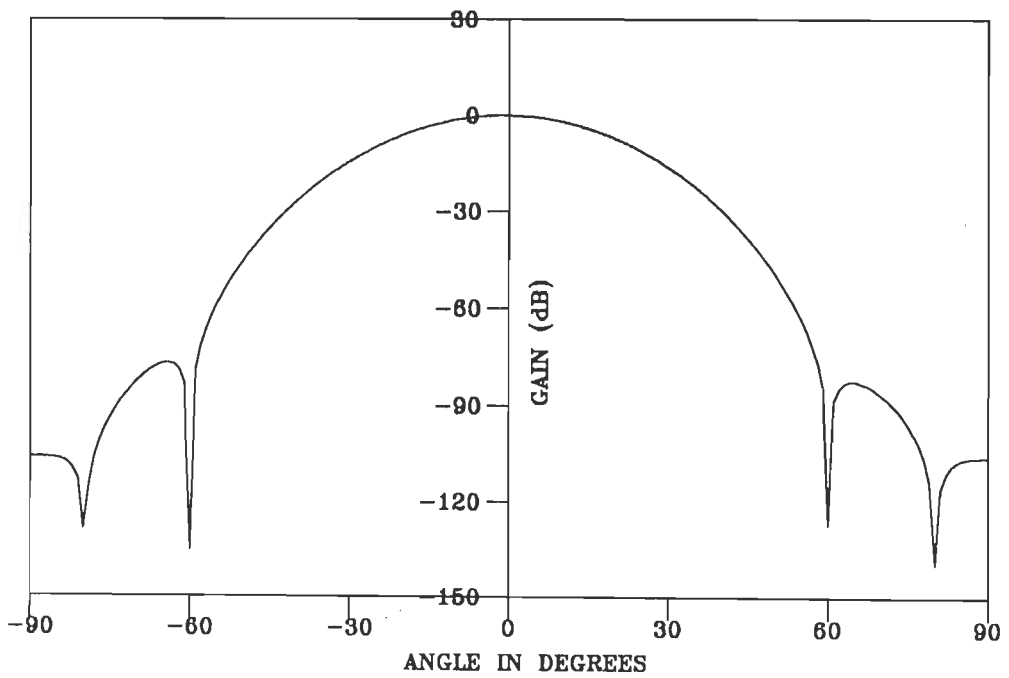
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER



(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

FIG.3.10 VOLTAGE PATTERNS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $60^\circ$ ,  $-60^\circ$ ,  $80^\circ$  &  $-80^\circ$ .

**Table-3.10**  
**Parameters of Interferences in Example 3.6-1.4**

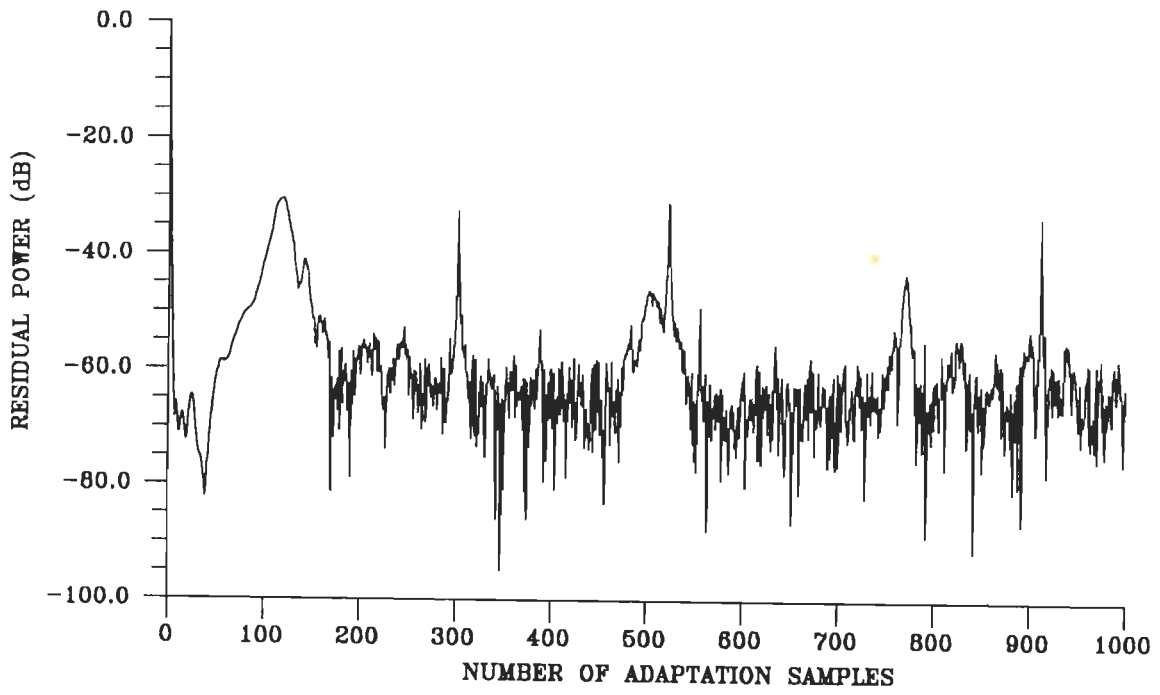
Parameter	$I_1$	$I_2$	$I_3$	$I_4$	$I_5$	$I_6$	$I_7$	$I_8$	$I_9$	$I_{10}$
$S_i$	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
$\theta_i$	$15^\circ$	$-15^\circ$	$30^\circ$	$-30^\circ$	$45^\circ$	$-45^\circ$	$60^\circ$	$-60^\circ$	$75^\circ$	$-75^\circ$
$\omega_i$	1.1	0.9	1.2	0.8	1.3	0.7	1.4	0.6	1.5	0.5

The residual power characteristics for this problem are shown in Fig.3.11. As can be seen from Fig.3.11(a), the RLS beamformer exhibits poor convergence in this case also. Though the beamformer converges in about 200 samples, the residual power characteristics exhibits undesirable overshoots at regular intervals. The other three beamformers exhibit much better convergence characteristics. The RMGS and RMGSEF beamformers converge in around 200 samples while the QRD-LS beamformer (Fig.3.11(d)) once again converges quickly. All these three beamformers exhibit approximately the same amount of residual power.

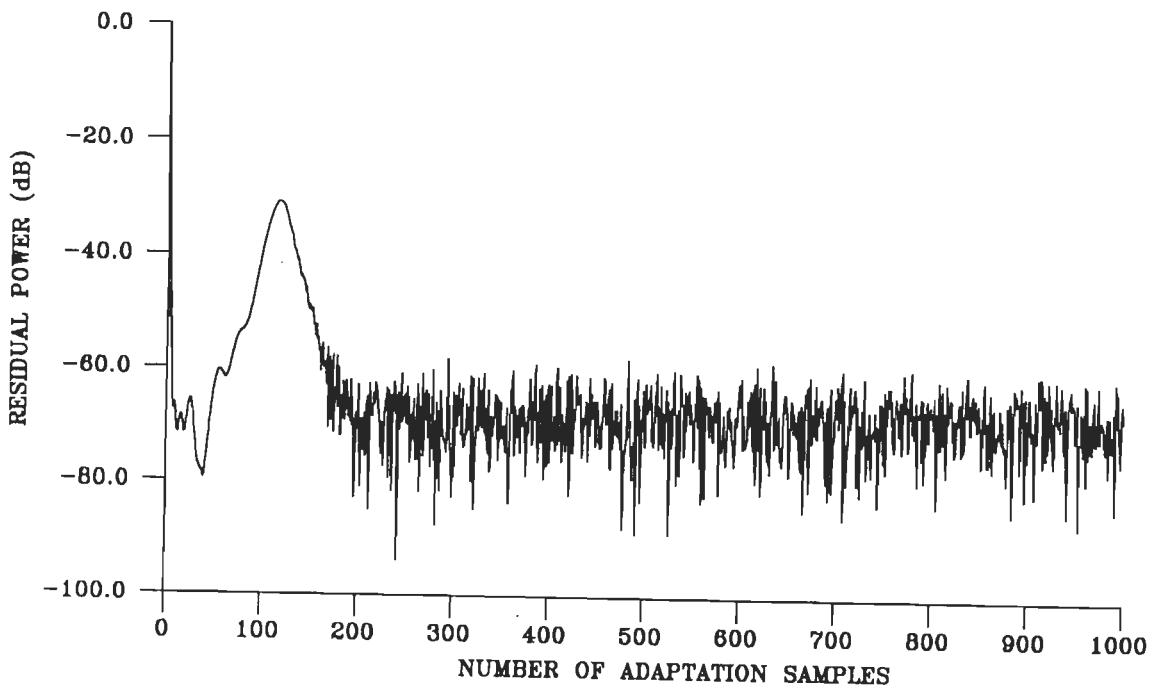
From the voltage pattern plots in Fig.3.12, it can be seen that the RLS beamformer produces a spurious null at  $85^\circ$ . Also, the null depths in the direction of interferences is small as compared to those produced by the other three beamformers. The RMGS, RMGSEF and the QRD-LS beamformers produce nulls, more or less the same depth ( $>120\text{dB}$ ) in the direction of interferences.

### 3.6-2 Broadband Arrays

For the broadband case, we consider a 6-element uniform linear array with four taps behind each element and a tap delay of

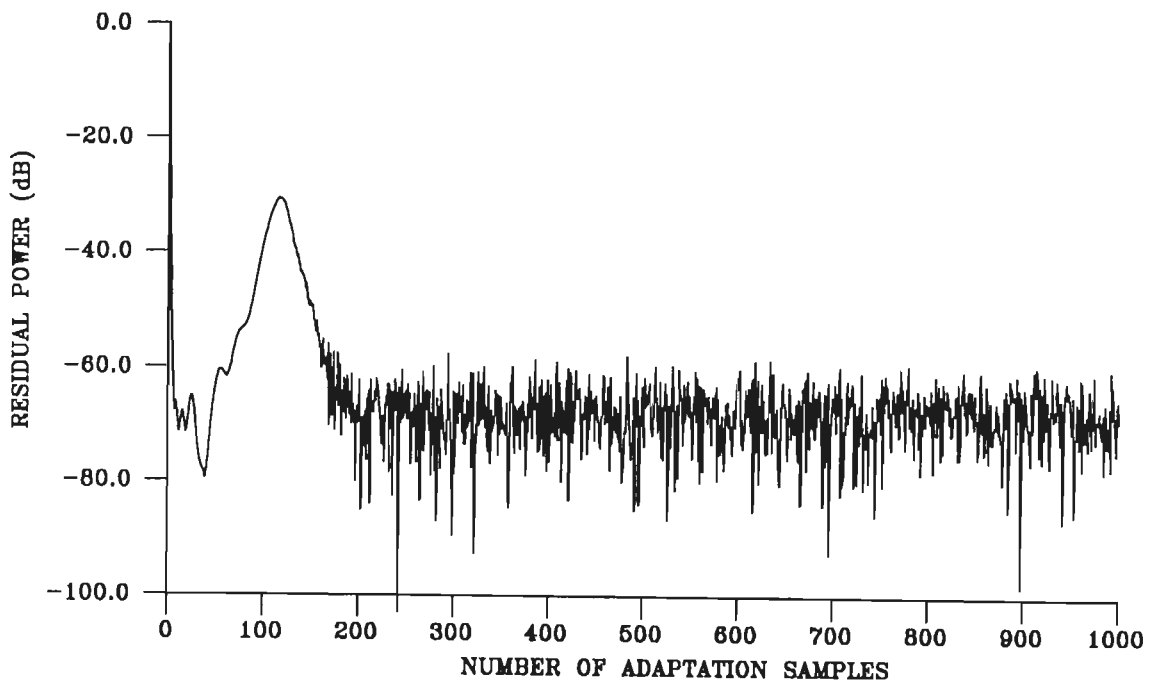


(a) RLS BEAMFORMER

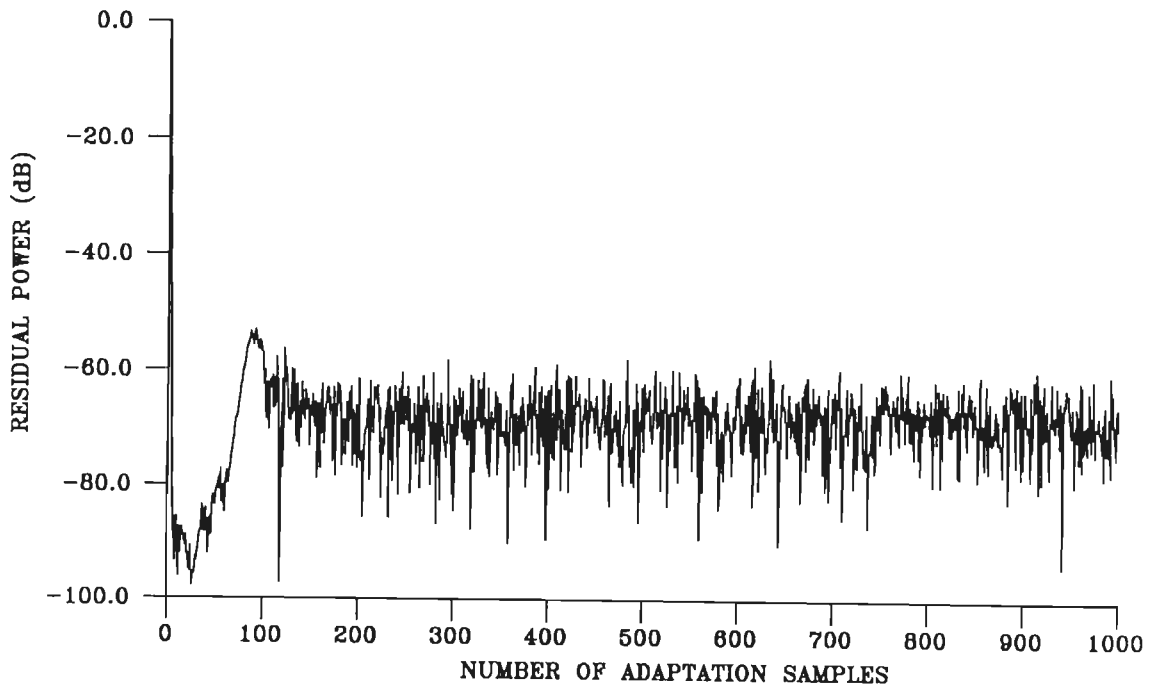


(b) RMGS BEAMFORMER



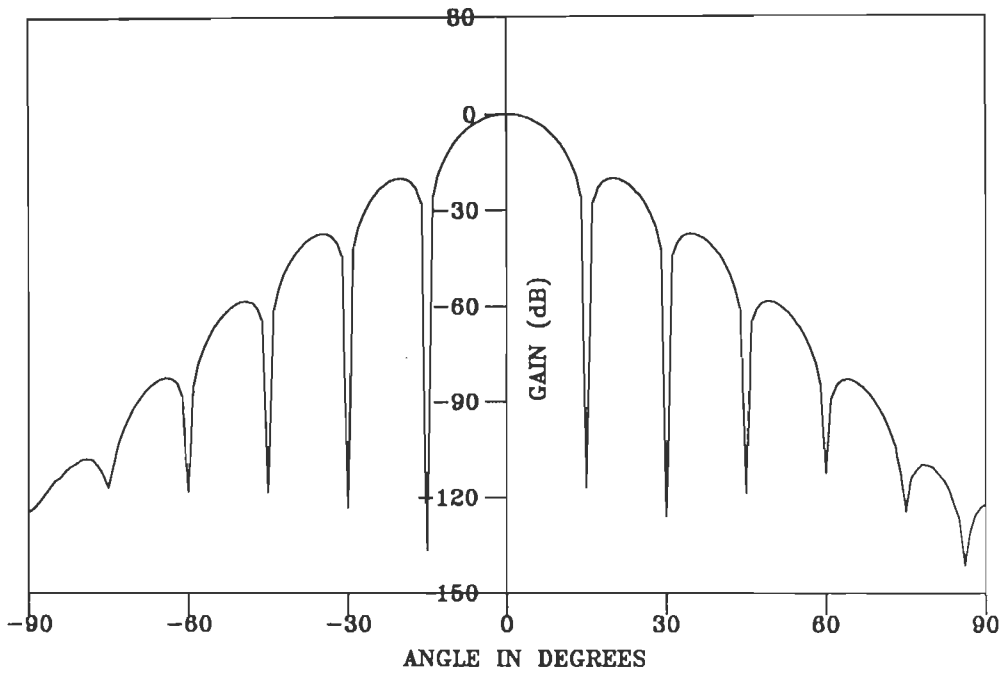


(c) RMGSEF BEAMFORMER

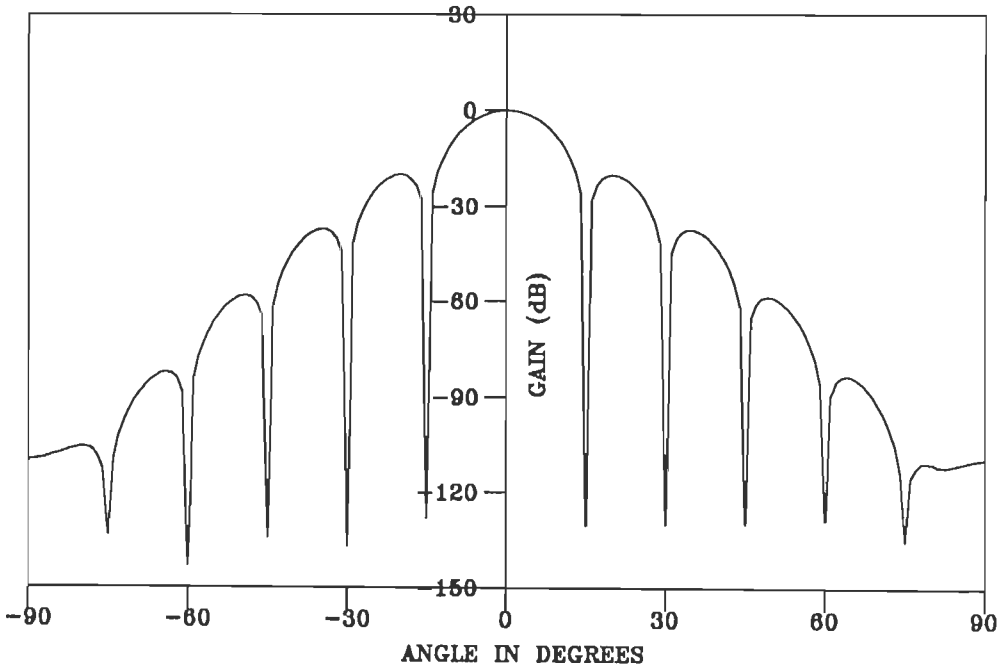


(d) QRD-LS BEAMFORMER

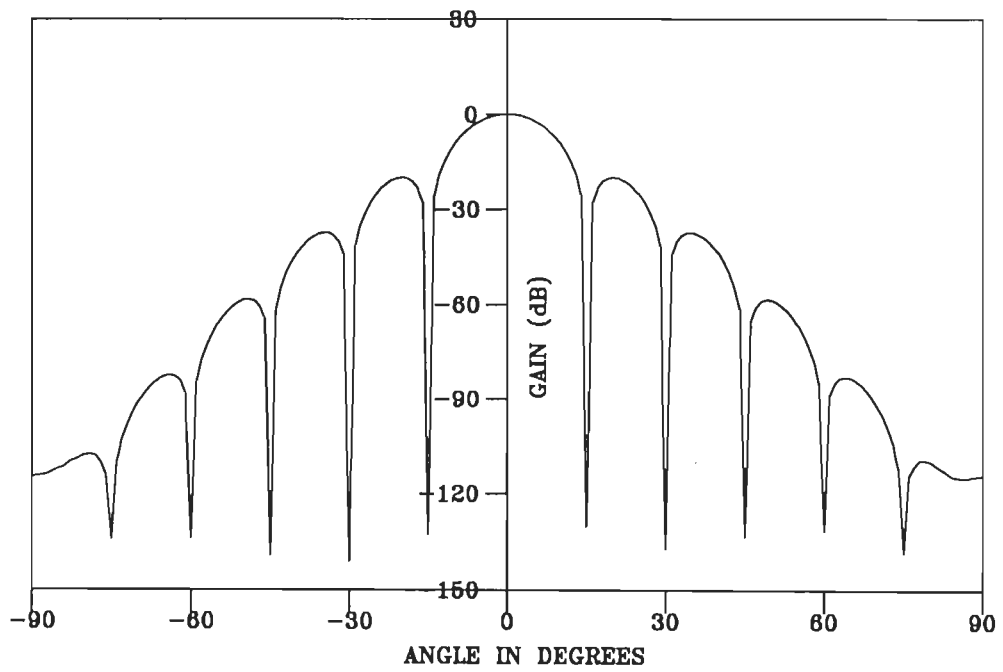
FIG.3.11 CONVERGENCE CHARACTERISTICS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $15^\circ, -15^\circ, 30^\circ, -30^\circ, 45^\circ, -45^\circ, 60^\circ, -60^\circ, 75^\circ$  &  $-75^\circ$ .



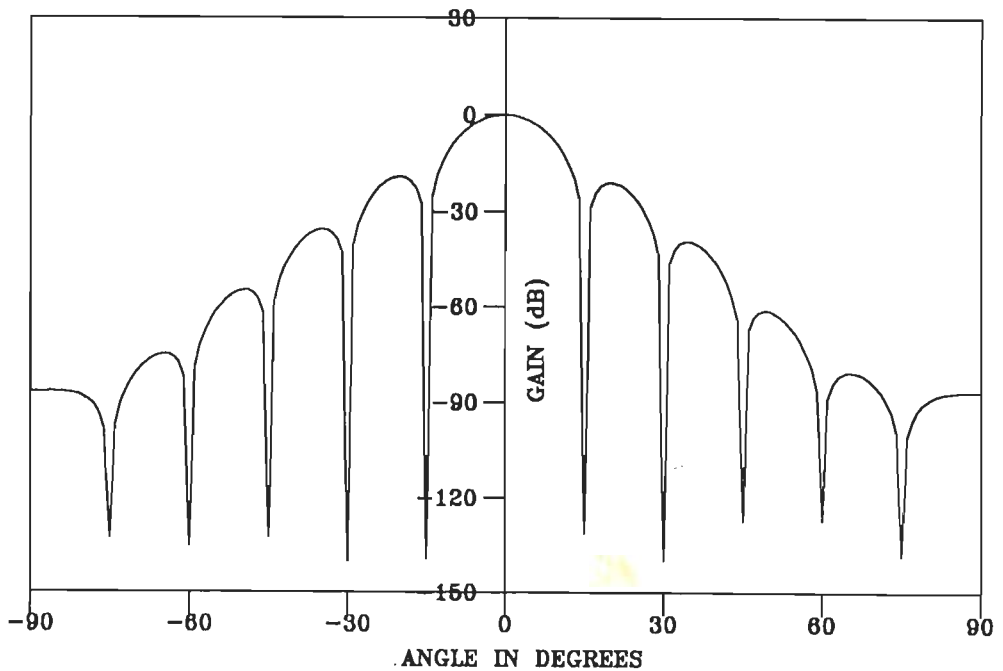
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER



(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

FIG.3.12 VOLTAGE PATTERNS OF NARROWBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $15^\circ, -15^\circ, 30^\circ, -30^\circ, 45^\circ, -45^\circ, 60^\circ, -60^\circ, 75^\circ$  &  $-75^\circ$ .

$1/4\omega_0$  (0.25) each. For the first four examples the signal environment consists of a desired signal and two interferences while four interferences have been assumed to be present for the fifth example.

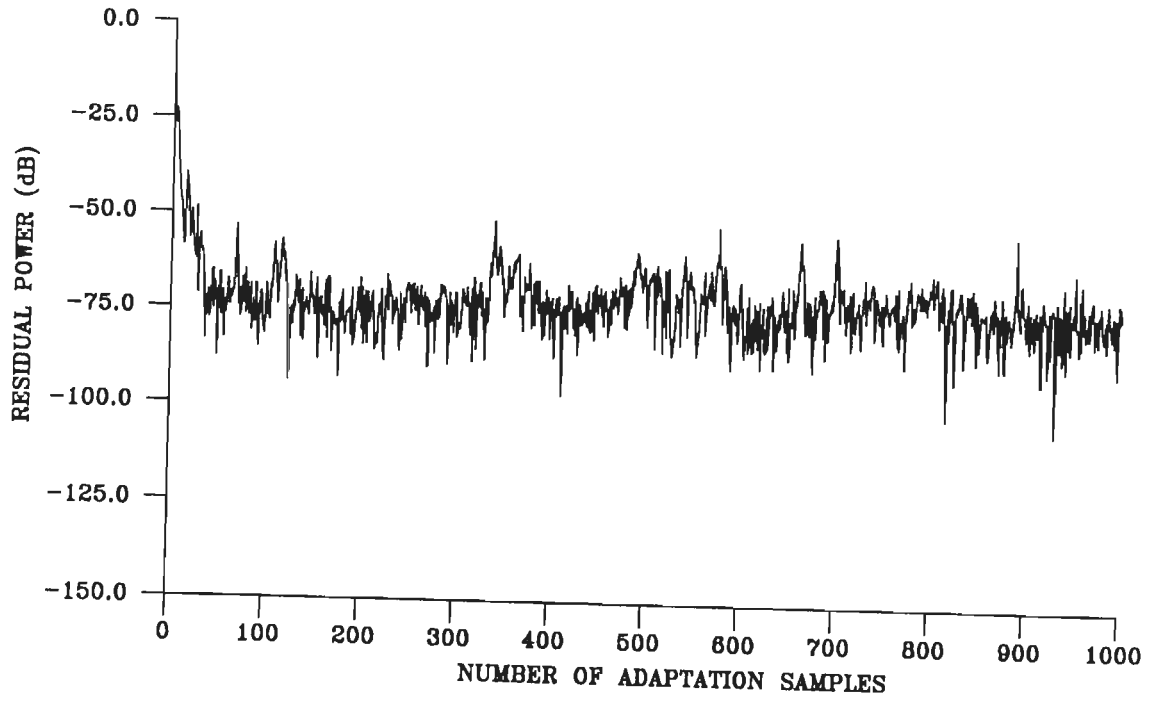
**Example 3.6-2.1** In this example, the signal environment has been modelled to consist of two broadband interferences whose parameters are given in Table-3.11.

Table-3.11

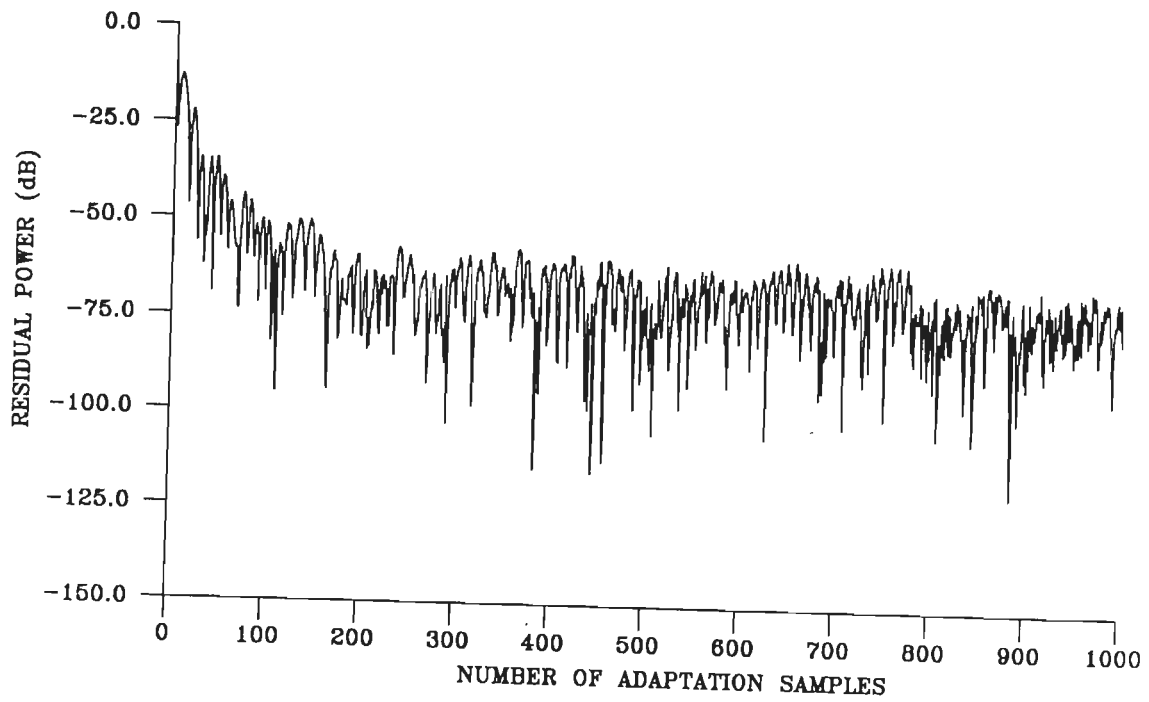
Parameters of the Broadband Interferences in Example 3.6-2.1

Parameter	Interference 1	Interference 2
$S_1$	10.0	10.0
$\theta_1$	$60^\circ$	$-60^\circ$
$\omega_1$	1.0	1.2
$\Delta_1$	0.8	1.0

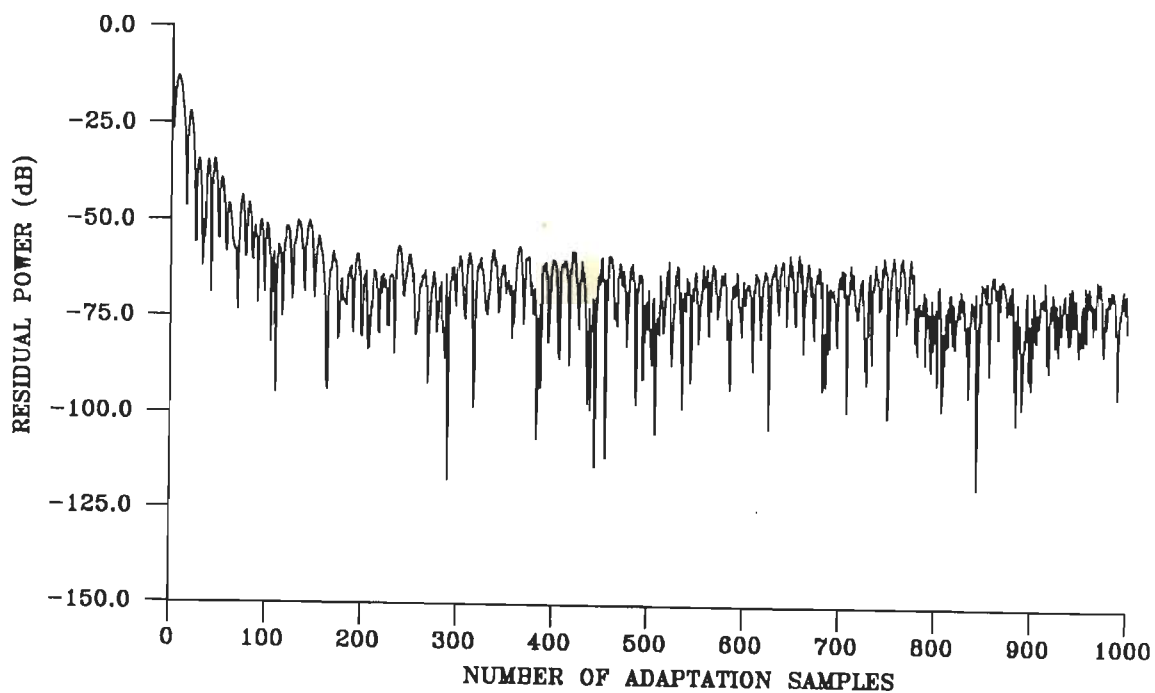
Fig.3.13 shows the residual power characteristics of the four beamformers. It is found that the RLS and the QRD-LS beamformers converge in about 50 samples which is approximately equal to twice the number of weights in the array. The RMGS and the RMGSEF beamformers (Fig.3.13(b) and(c)) on the other hand, exhibit a relatively lower convergence speed and take about 200 samples to converge. It may be noted that although the RLS beamformer has a faster convergence speed it exhibits undesirable fluctuations at regular intervals. The RLS, RMGS and the RMGSEF beamformers have nearly the same amount of residual power (-75dB) while the QRD-LS beamformer has the residual power of about -80dB.



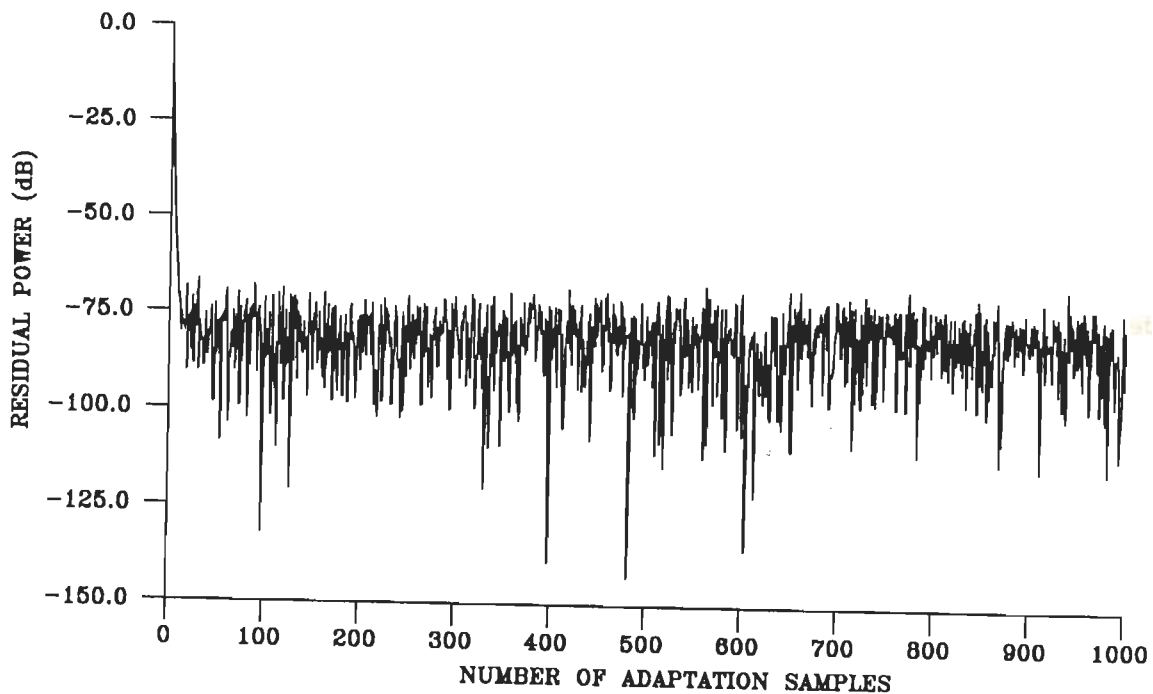
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER



(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

FIG.3.13 CONVERGENCE CHARACTERISTICS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $60^\circ$  AND  $-60^\circ$ .

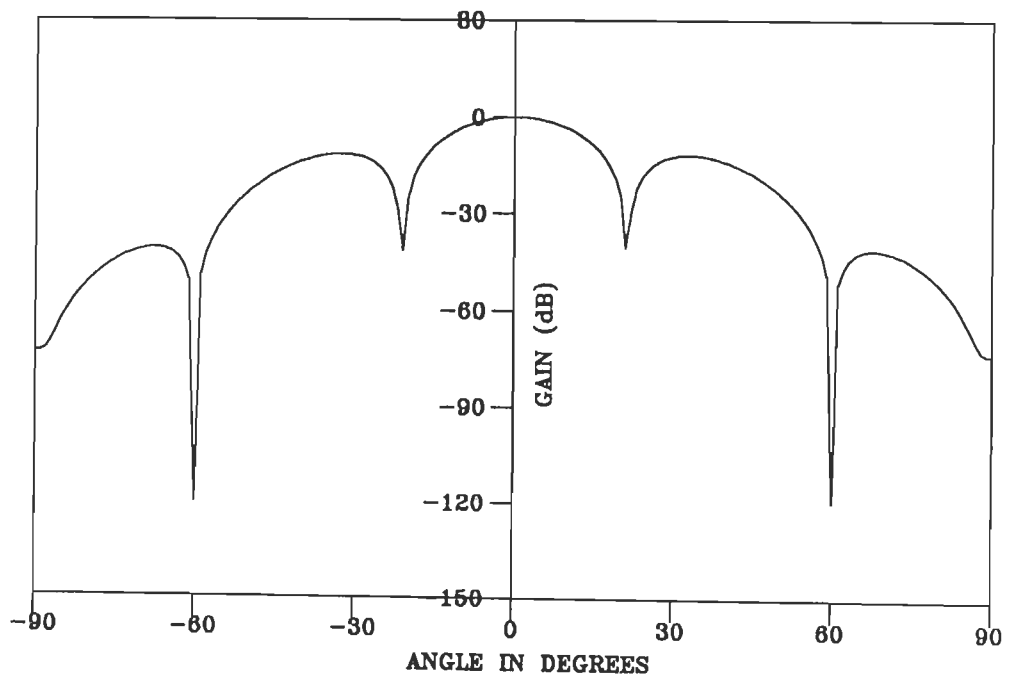
The voltage patterns and the output signal waveforms of these beamformers are shown in Fig.3.14 and Fig.3.15, respectively. It can be seen that the RLS, RMGS and the RMGSEF beamformers produce 120dB nulls depth in the directions of two interferences. The QRD-LS beamformer exhibits better null depth, on the order of 135dB. Further, a comparison of Fig.3.15 with Fig.2.7 shows that all the beamformers track the desired signal satisfactorily.

**Example 3.6-2.2** In this example, the interferences are modelled to arrive close to each other but far away from the desired signal. The interference arrival angles are  $50^\circ$  and  $60^\circ$ . The remaining parameters are as in Table 3.11.

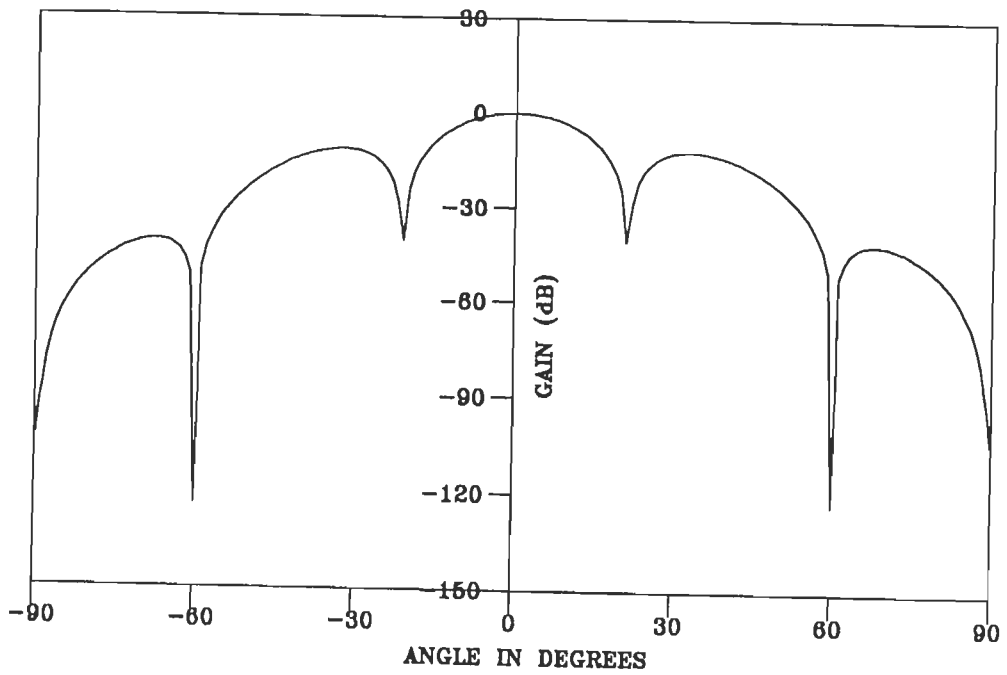
Fig.3.16 and 3.17, respectively, show the residual power and the voltage patterns for the four beamformers. In this situation also, all the beamformers perform satisfactorily with QRD-LS being marginally better.

**Example 3.6-2.3** In this example, the two interferences have been modelled to arrive from directions very close to the desired signal, i.e.  $+5^\circ$  and  $-5^\circ$ . The remaining parameters of the interferences are as in Table.3.11.

From the residual power characteristics shown in Fig.3.18, it can be seen that all the beamformers exhibit a larger residual output power as compared to the residual power produced in the previous two examples. Further, unlike the previous two cases, the convergence of RLS beamformer, is slow and it requires about 400 samples to converge. The remaining three beamformers, however, exhibit

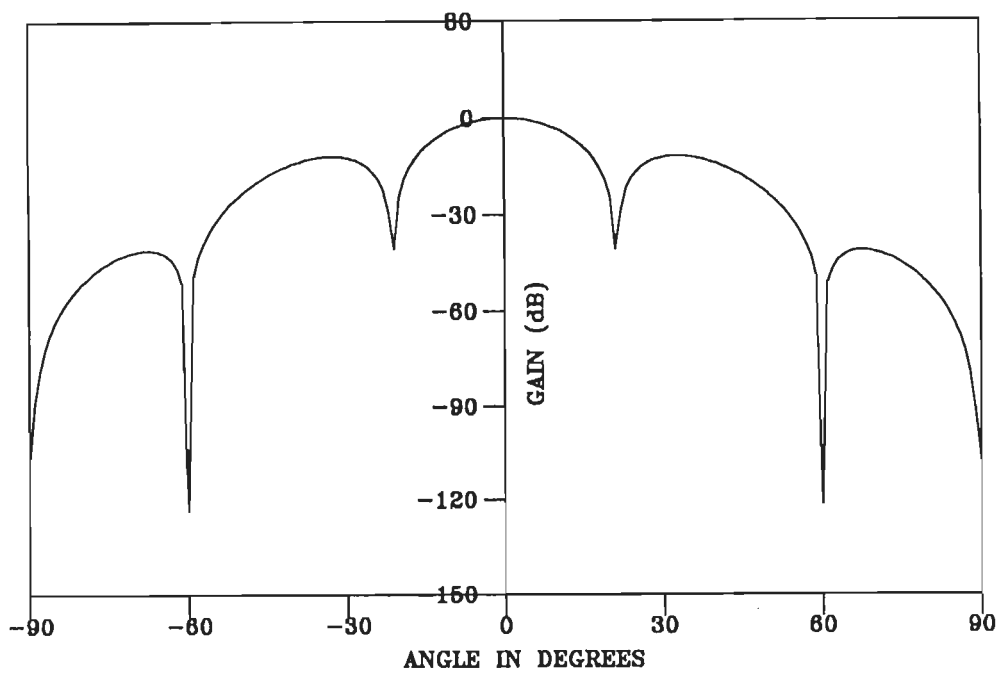


(a) RLS BEAMFORMER

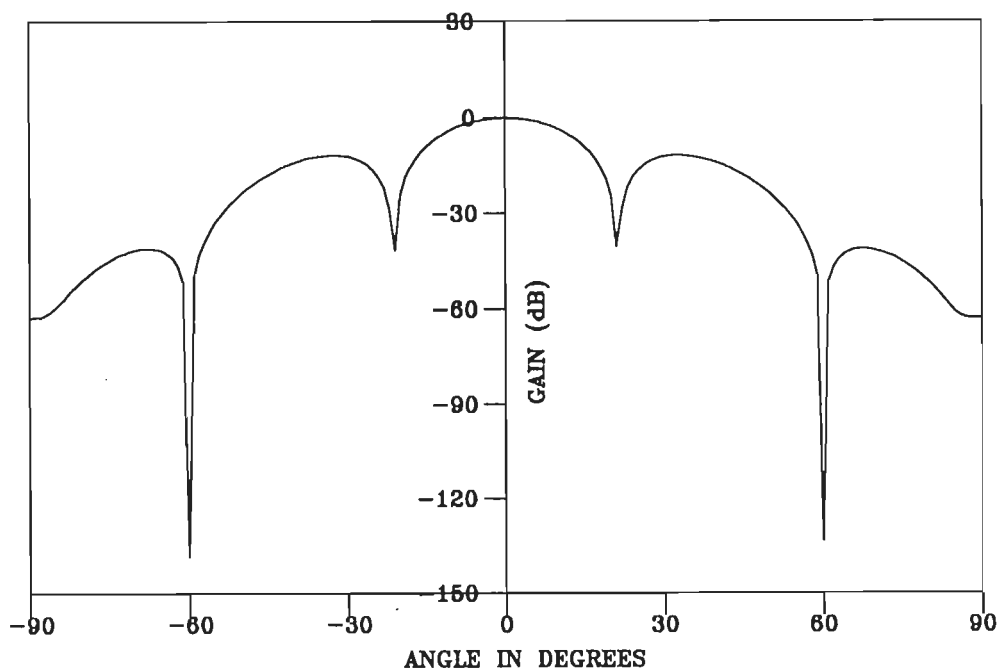


(b) RMGS BEAMFORMER



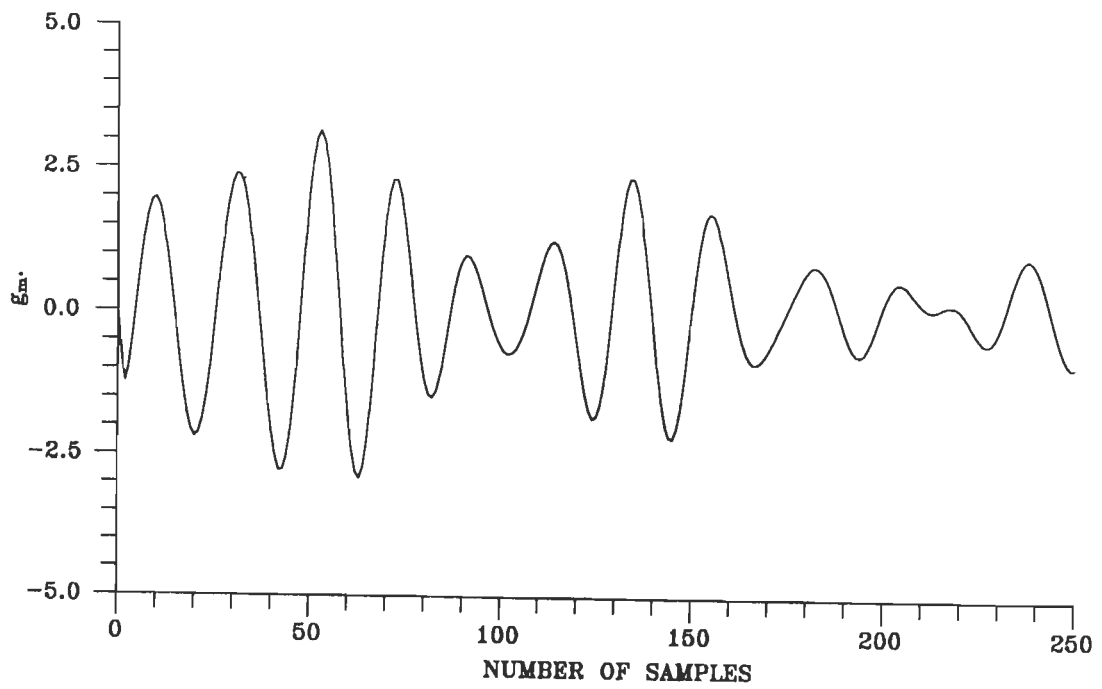


(c) RMGSEF BEAMFORMER

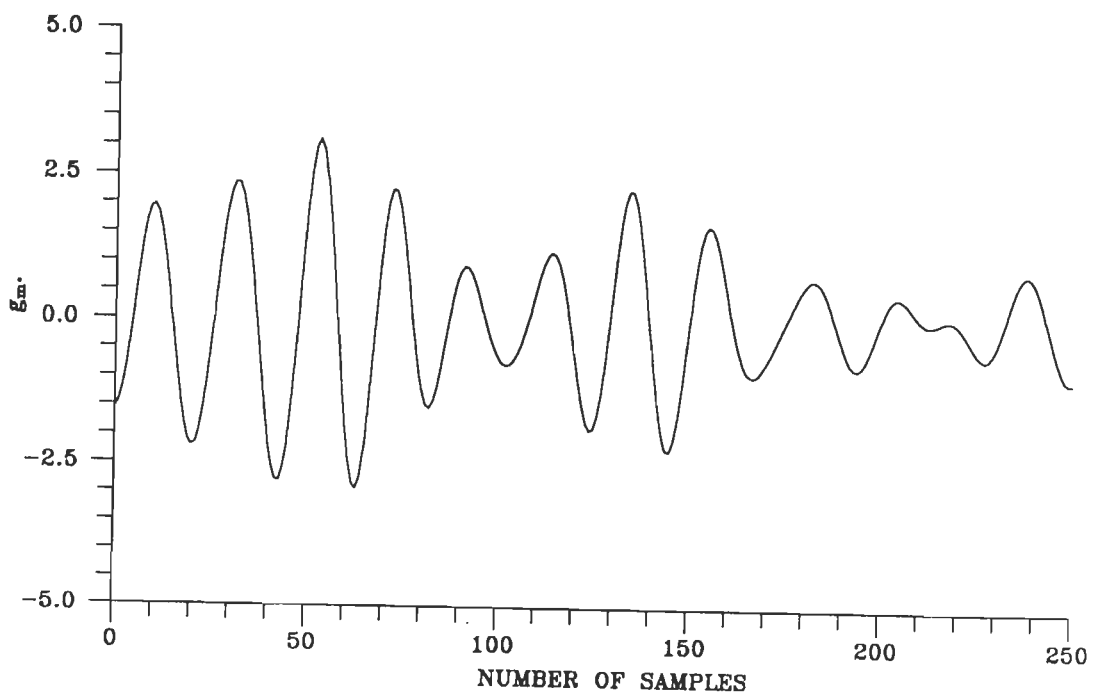


(d) QRD-LS BEAMFORMER

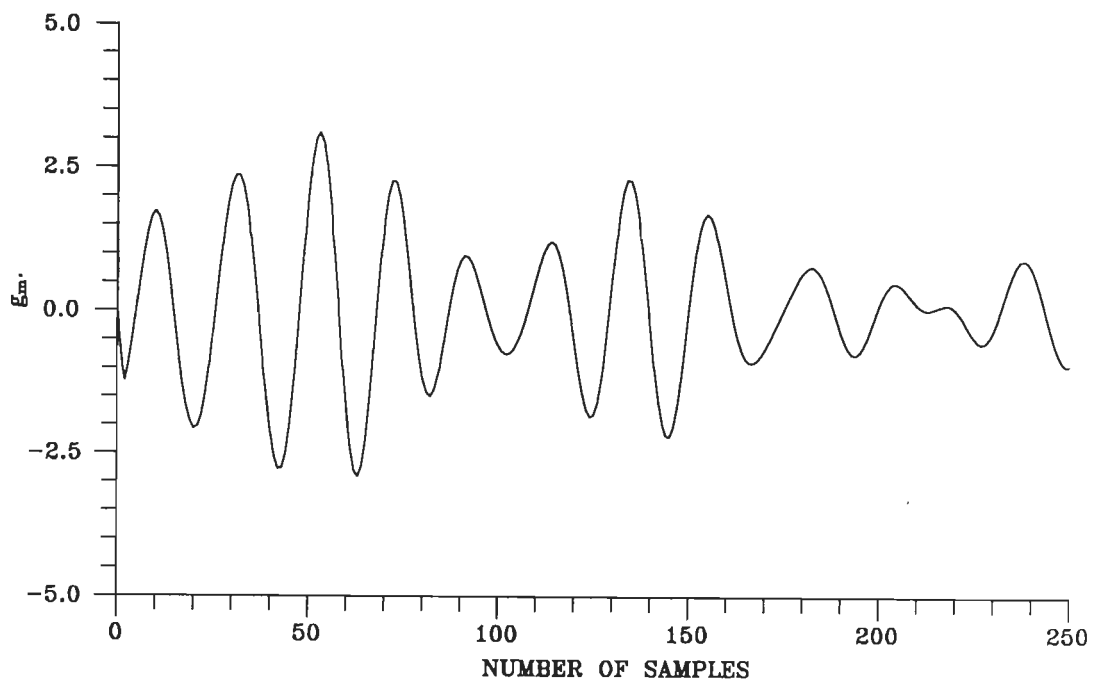
FIG.3.14 VOLTAGE PATTERNS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $60^\circ$  &  $-60^\circ$ .



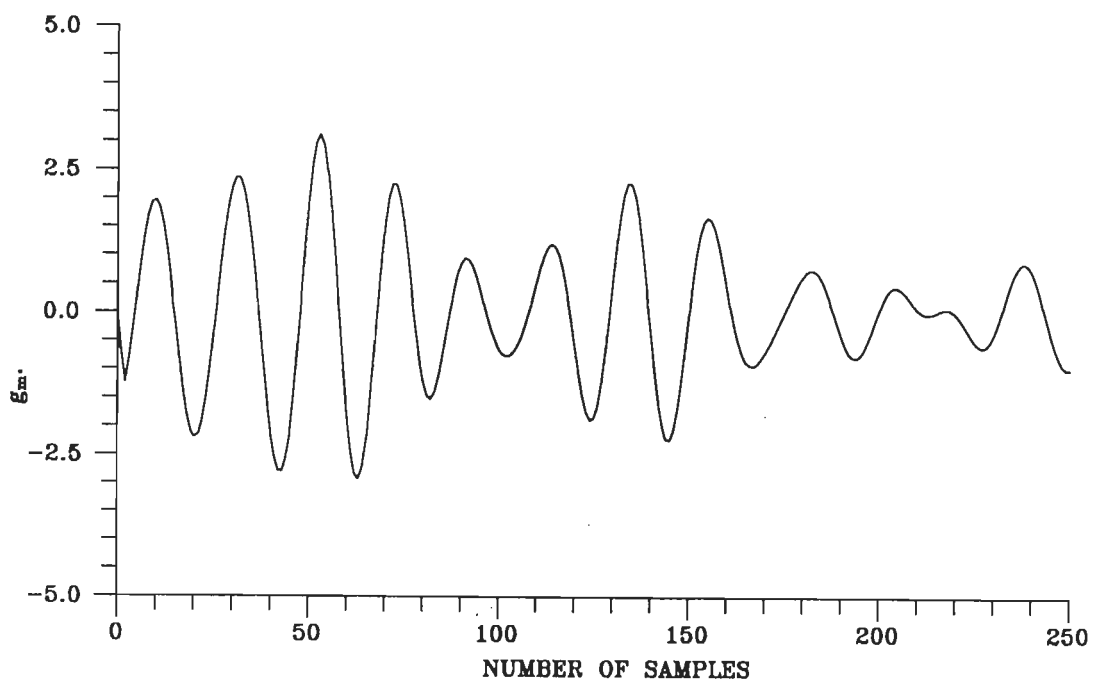
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER

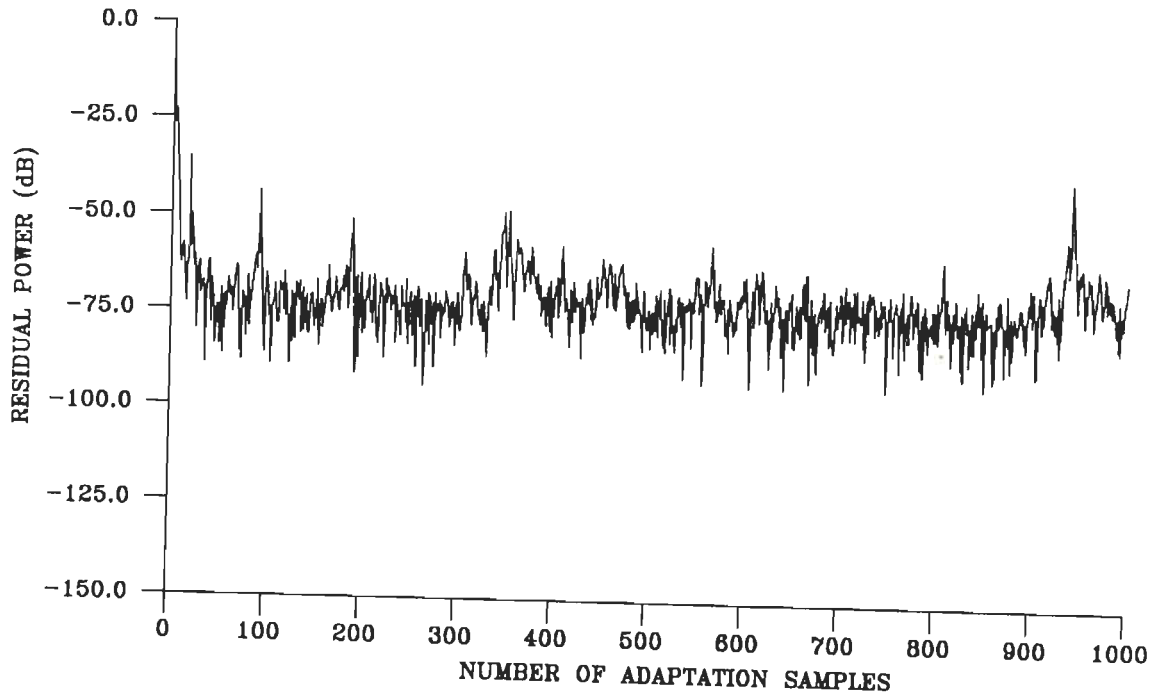


(c) RMGSEF BEAMFORMER

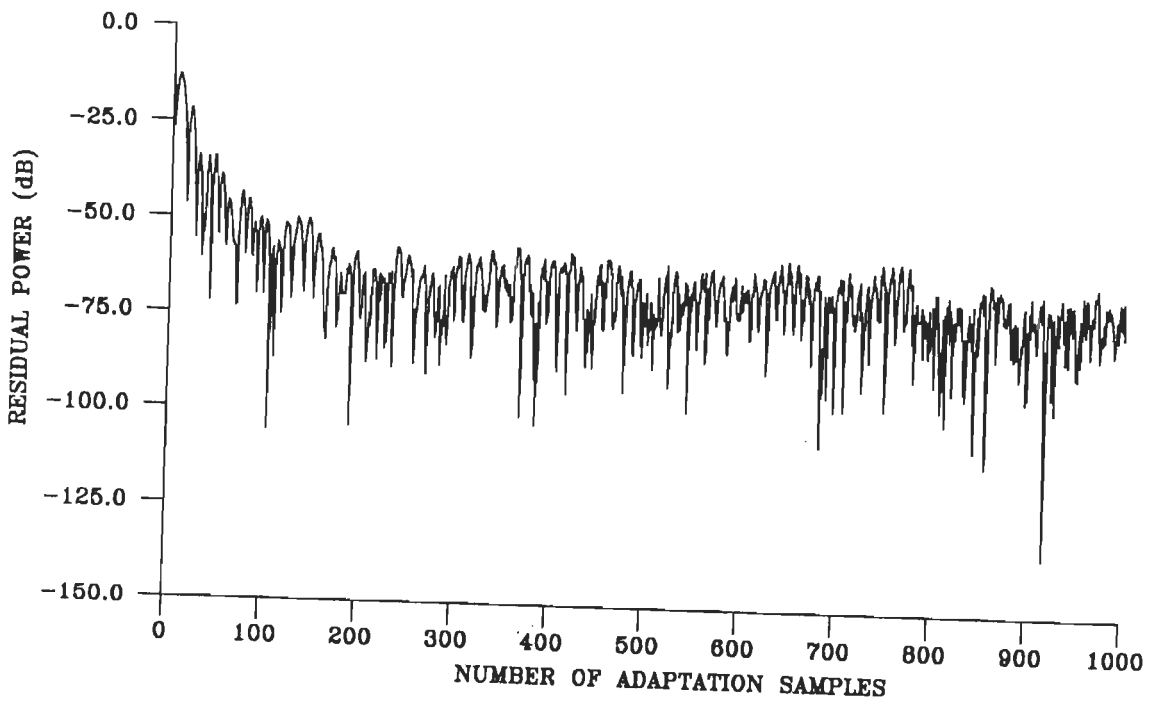


(d) QRD-LS BEAMFORMER

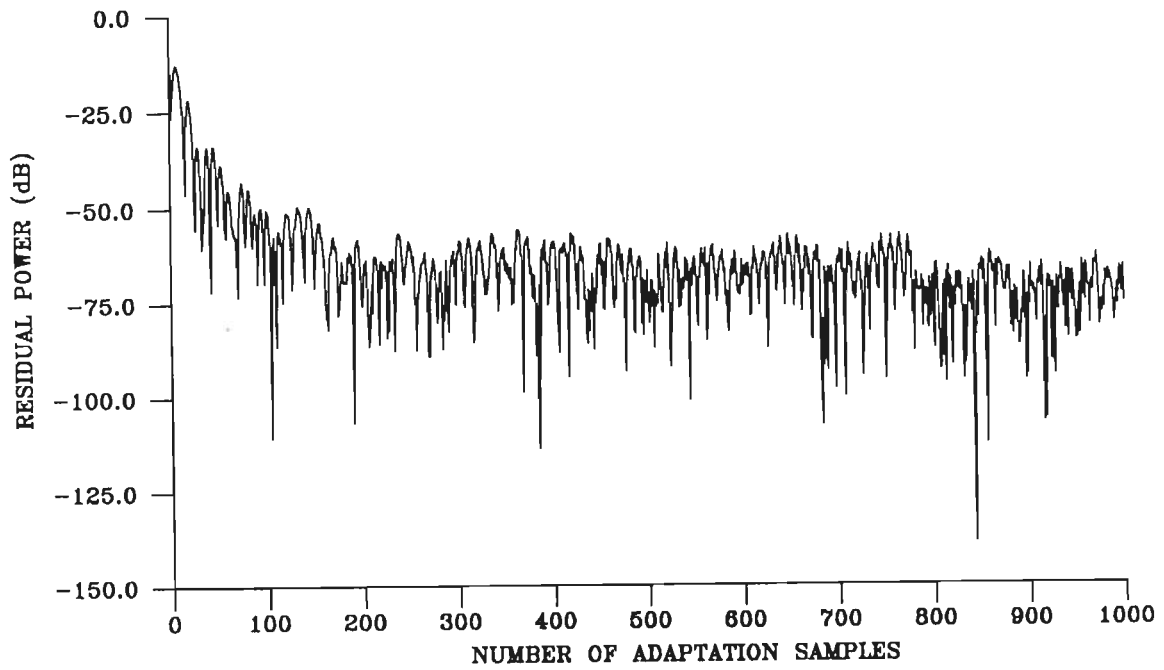
FIG.3.15 OUTPUT WAVEFORMS OF BROADBAND BEAMFORMERS



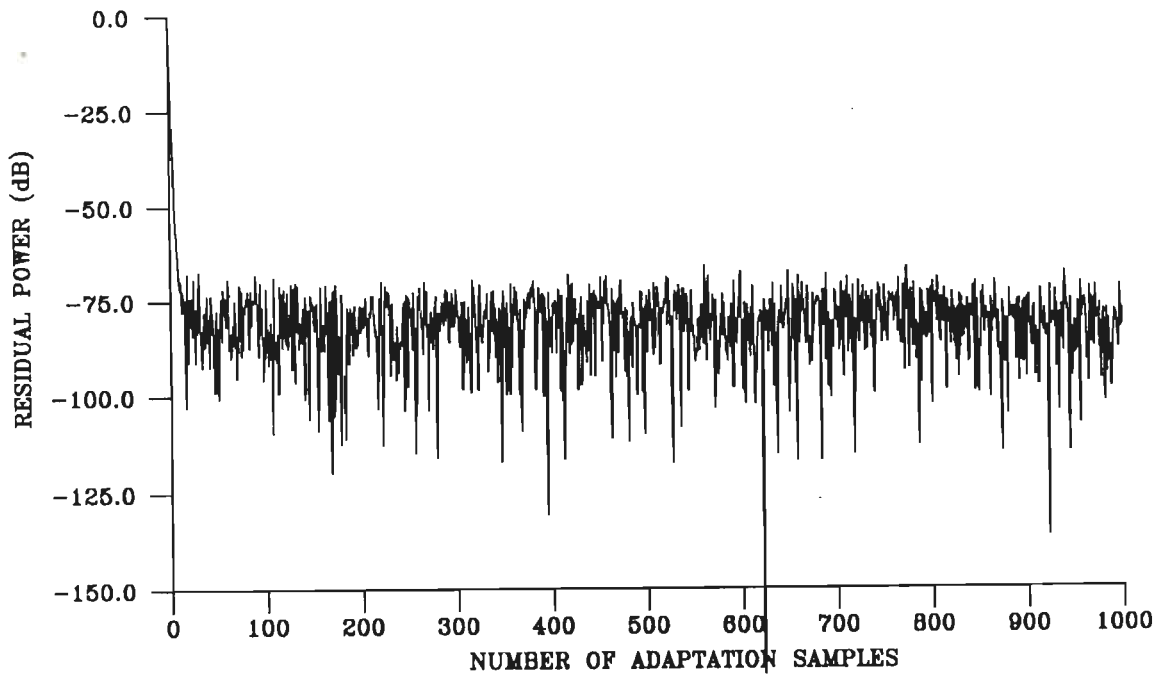
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER

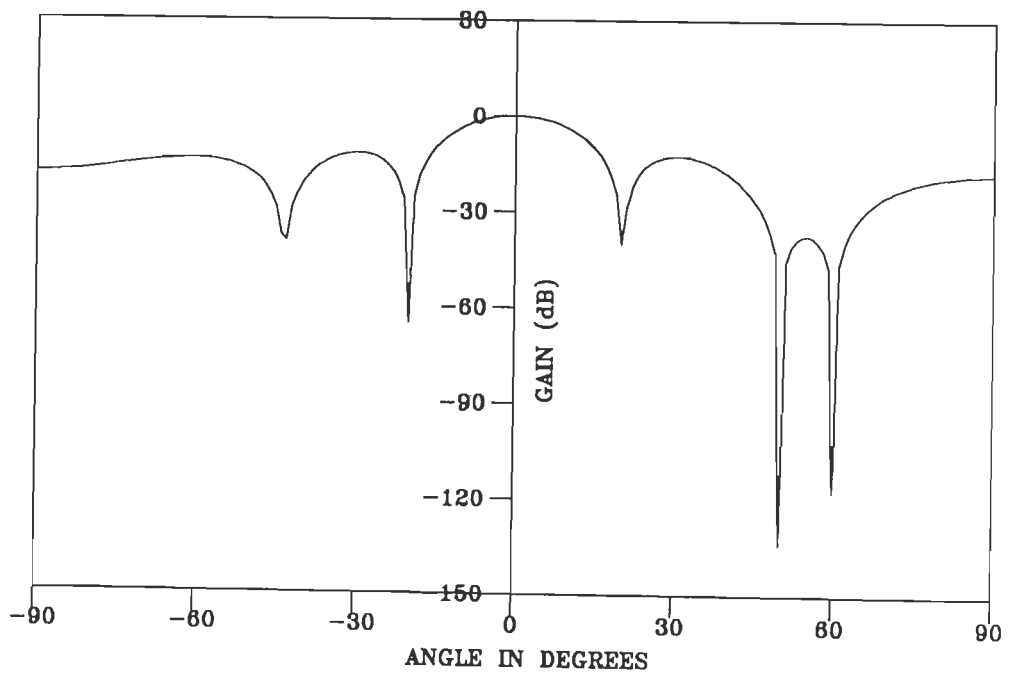


(c) RMGSEF BEAMFORMER

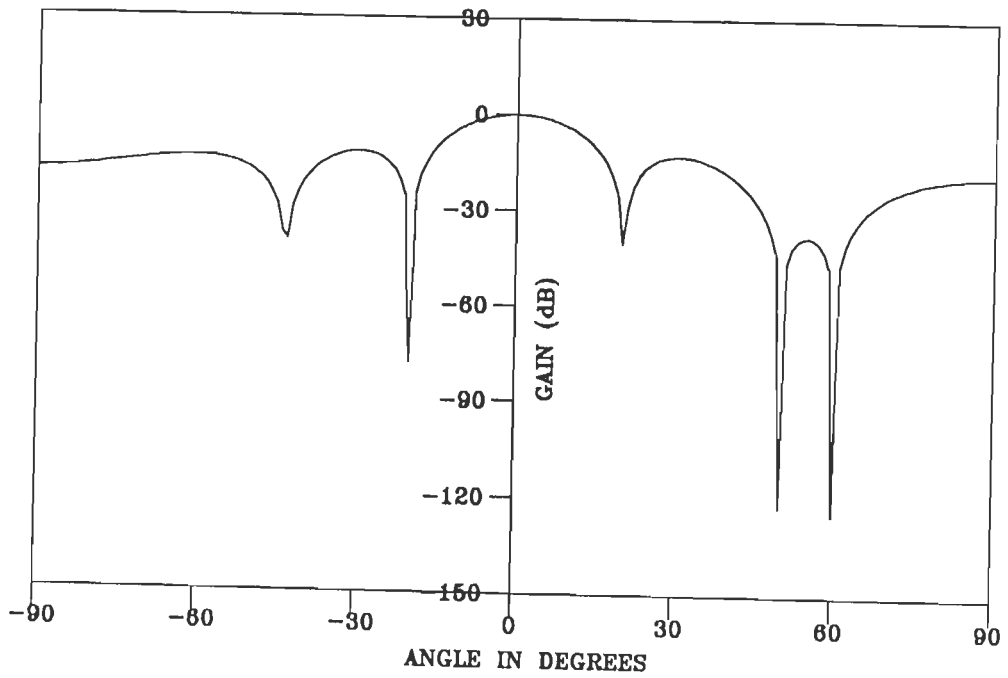


(d) QRD-LS BEAMFORMER

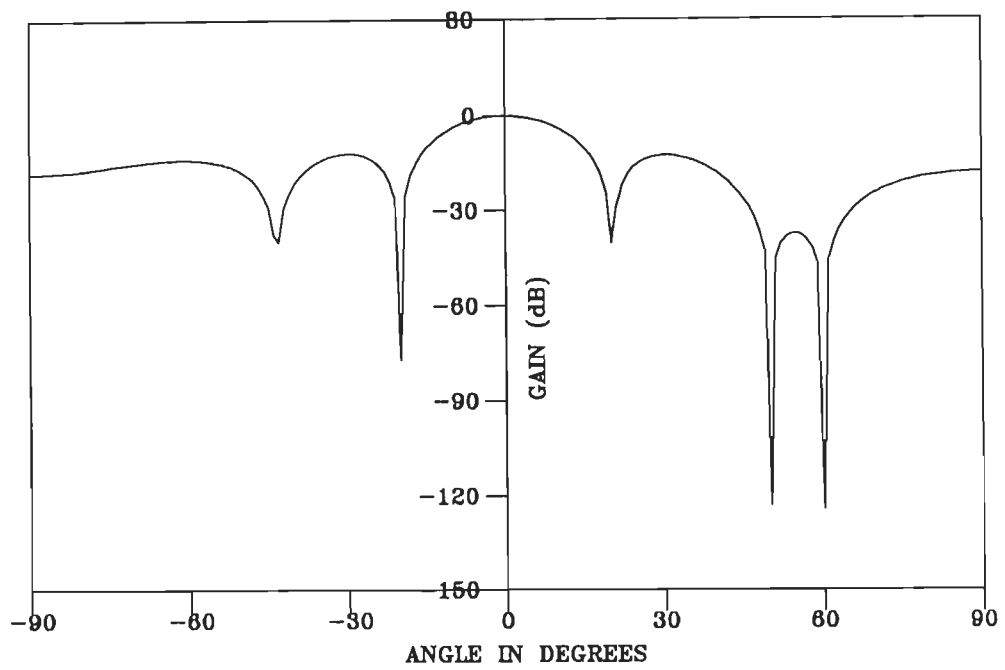
FIG.3.16 CONVERGENCE CHARACTERISTICS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $50^\circ$  AND  $60^\circ$ .



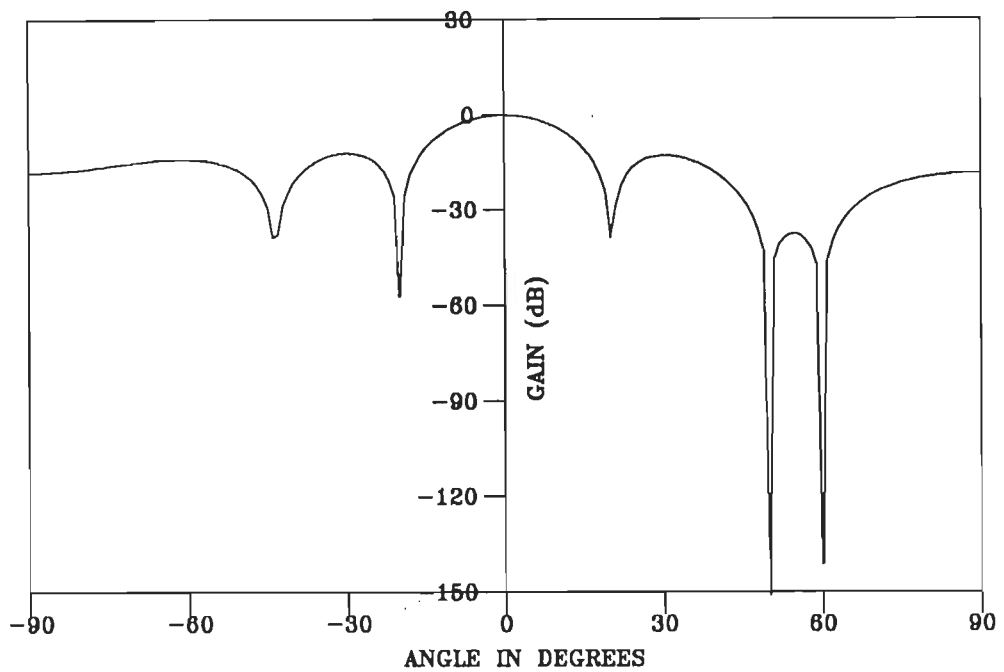
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER

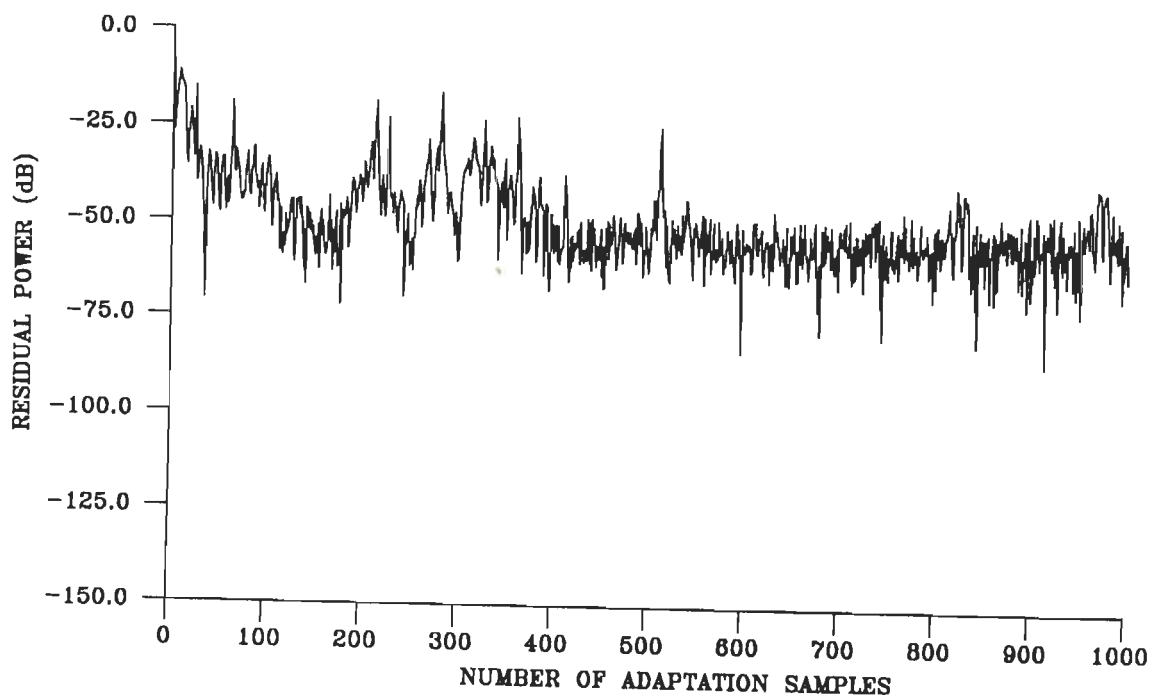


(c) RMGSEF BEAMFORMER

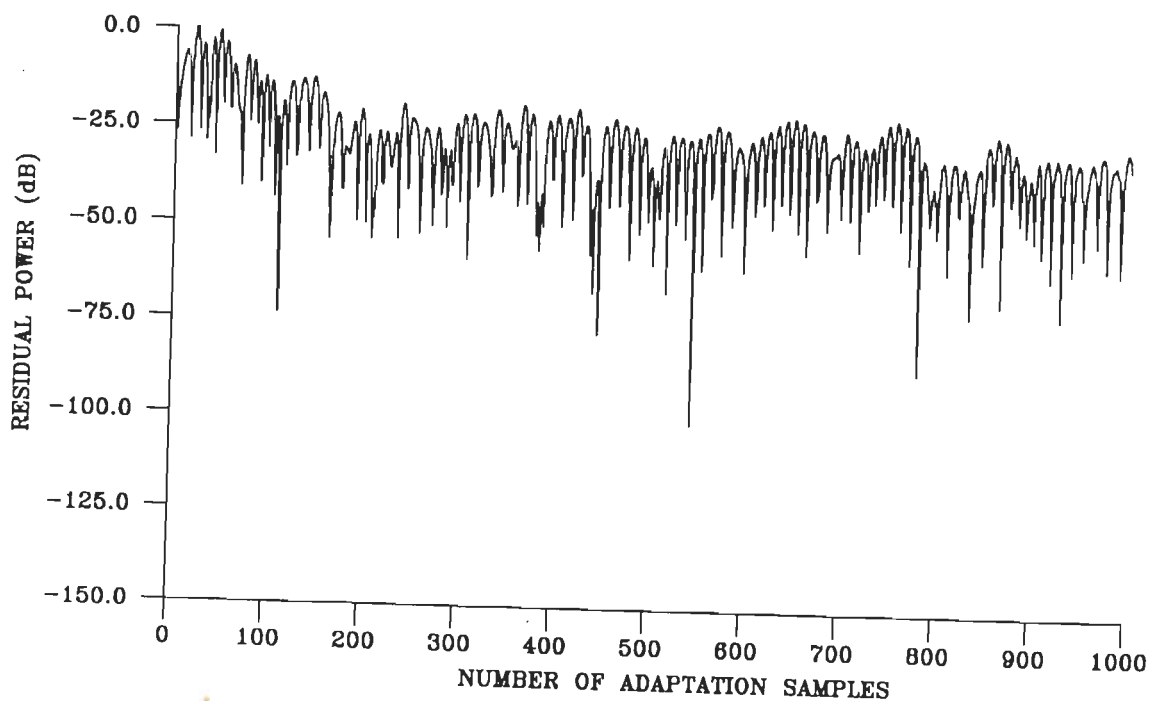


(d) QRD-LS BEAMFORMER

FIG.3.17 VOLTAGE PATTERNS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT 50° & 60°.

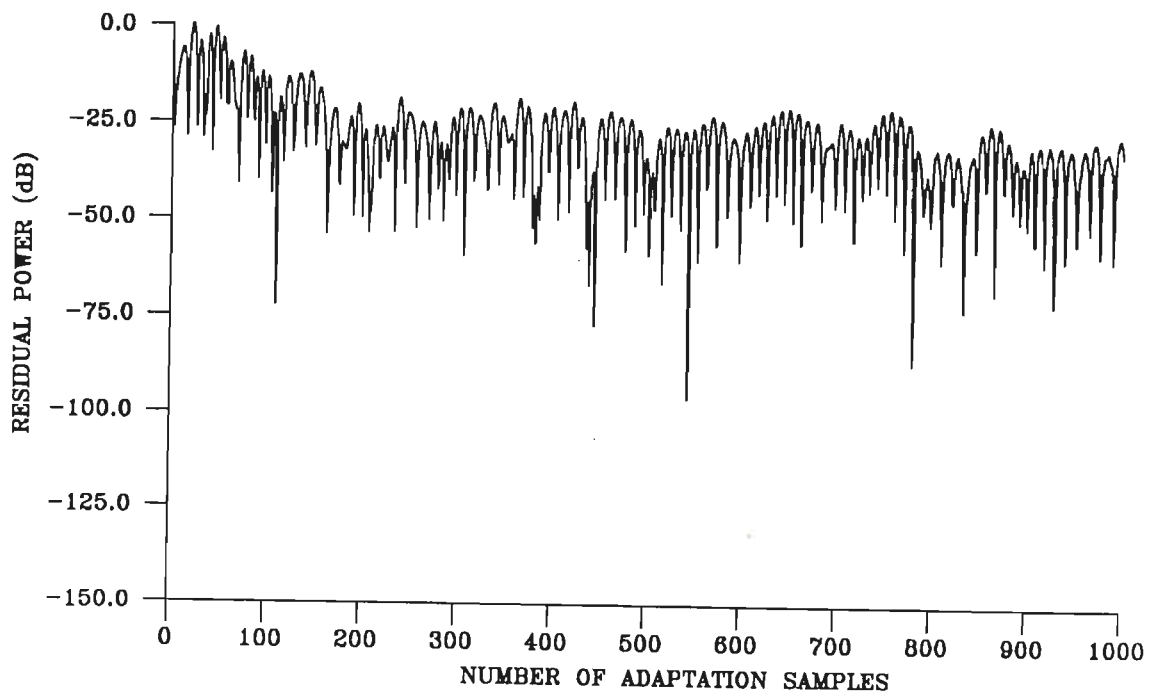


(a) RLS BEAMFORMER

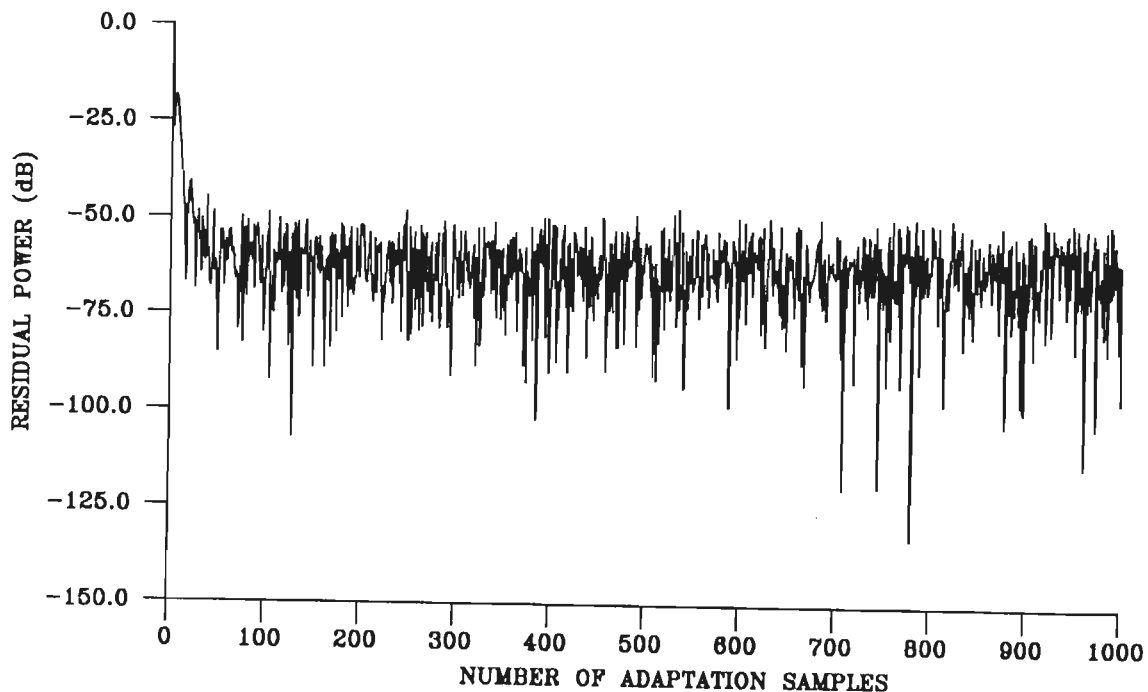


(b) RMGS BEAMFORMER





(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

FIG.3.18 CONVERGENCE CHARACTERISTICS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $5^\circ$  AND  $-5^\circ$ .

the same convergence speed as in earlier examples.

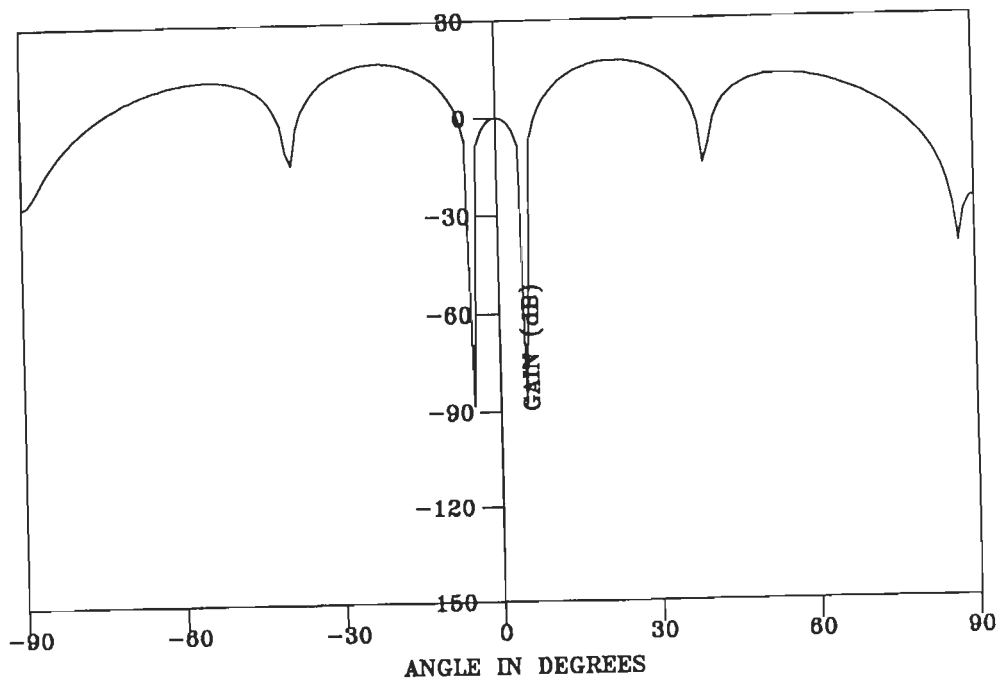
From Fig.3.19, it is seen that all the four beamformers successfully place deep nulls in the directions of interferences and the largest null depths is again obtained in the case of QRD-LS beamformer.

**Example 3.6-2.4** In this example, the interferences arrive from near endfire directions, ie,  $80^\circ$  and  $-80^\circ$ . The remaining parameters are as in Table 3.11.

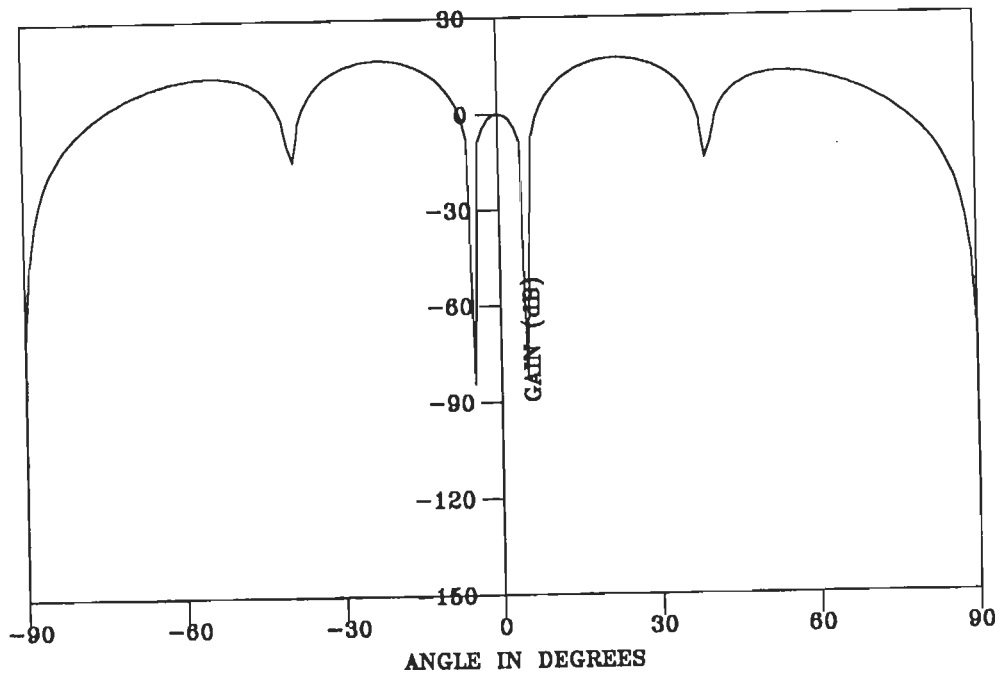
A study of Fig.3.20 indicates that for this signal environment, both the RLS and the QRD-LS beamformers exhibit fastest convergence and take only 40 samples to converge. The RMGS and RMGSEF beamformers take about 100 samples to converge which is approximately half the number required in previous examples. After convergence is achieved, all the beamformers produce, more or less, the same amount of residual power.

The voltage patterns are shown in Fig.3.21. The RLS beamformer produces deep nulls of about -115dB depth in the direction of interferences. However, it also produces a spurious null at about  $85^\circ$ . The other three beamformers produce nulls ( $>120$ dB) in the direction of the interferences.

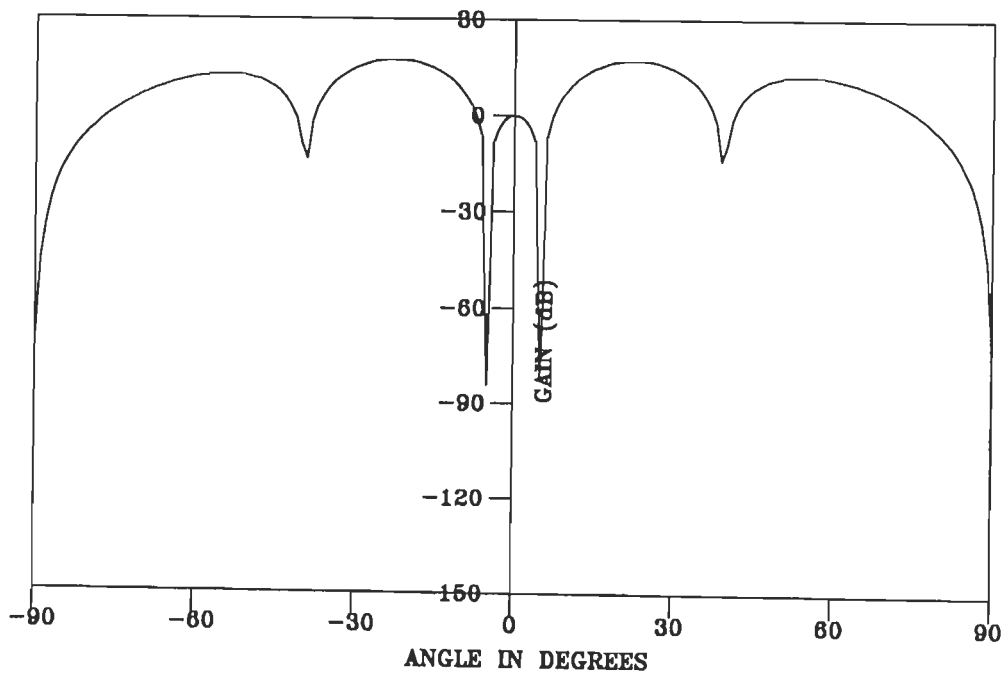
**Example 3.6-2.5** In order to compare the performance of exact least-square algorithms in the narrowband and broadband signal environments, we consider here the case of four interferences whose parameters are given in Table-3:12.



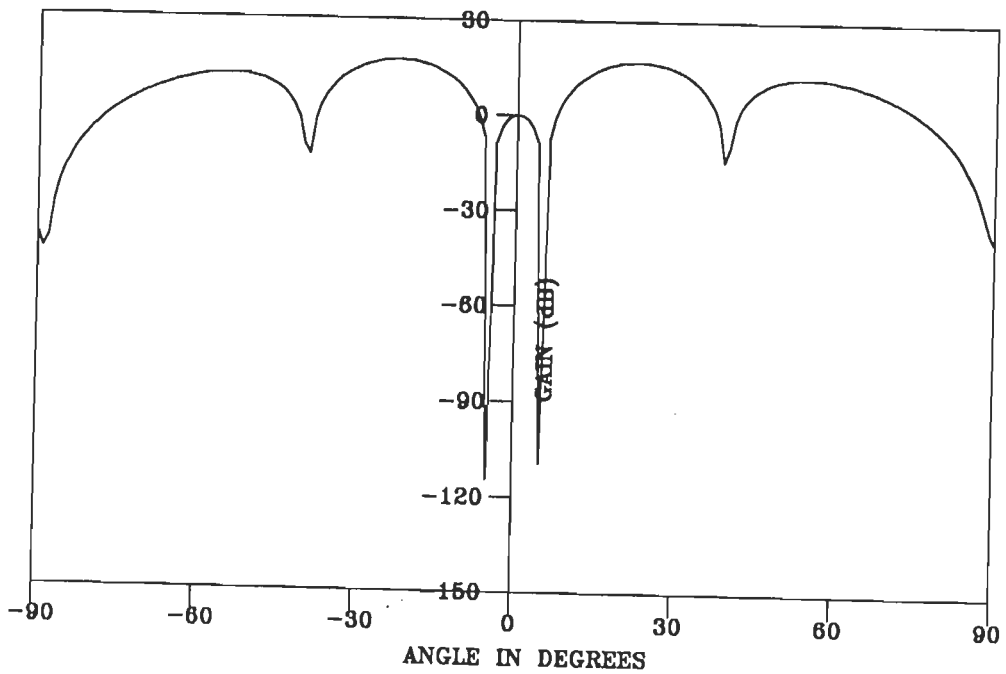
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER

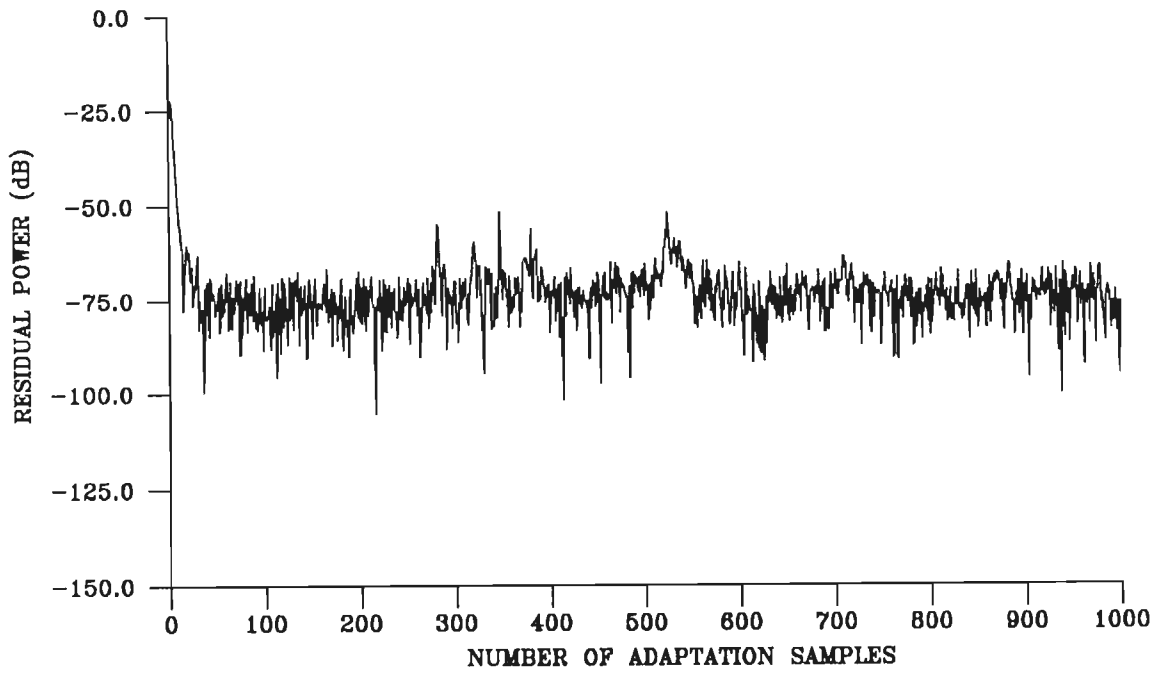


(c) RMGSEF BEAMFORMER

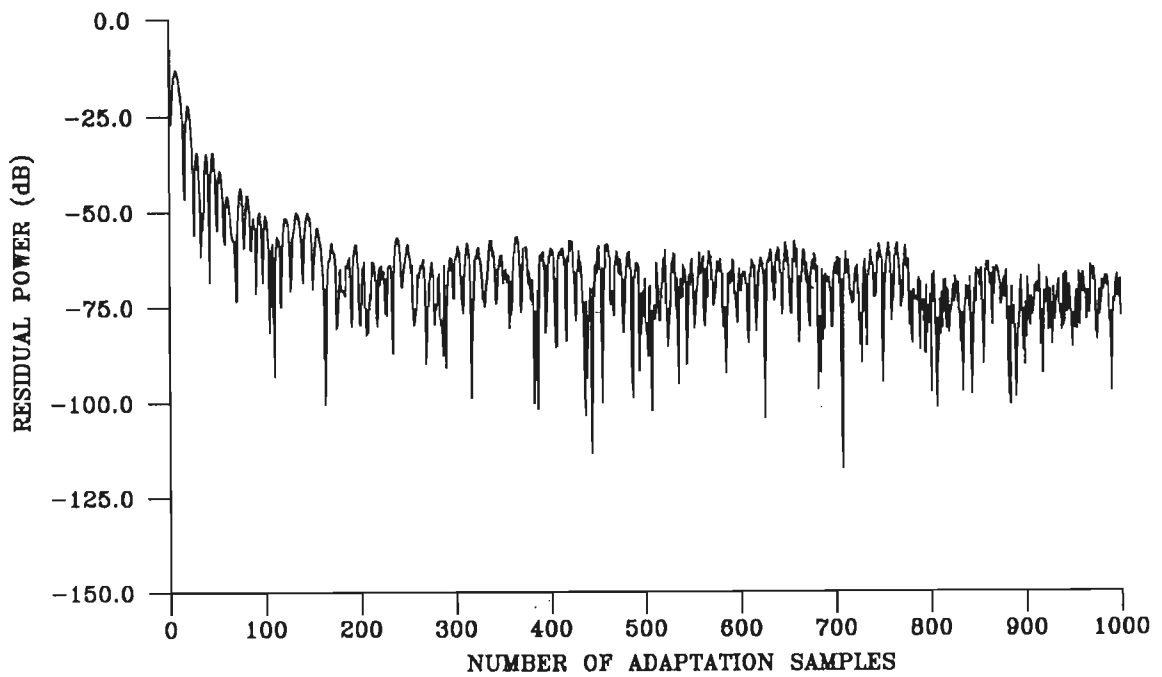


(d) QRD-LS BEAMFORMER

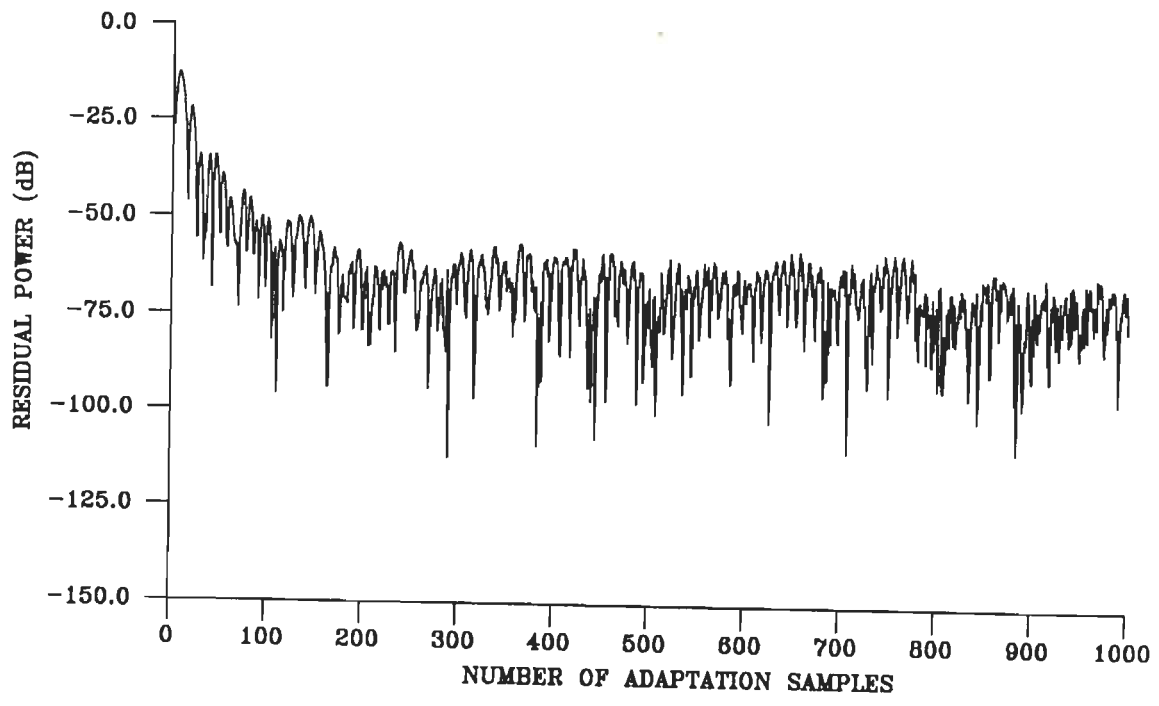
FIG.3.19 VOLTAGE PATTERNS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $5^\circ$  &  $-5^\circ$ .



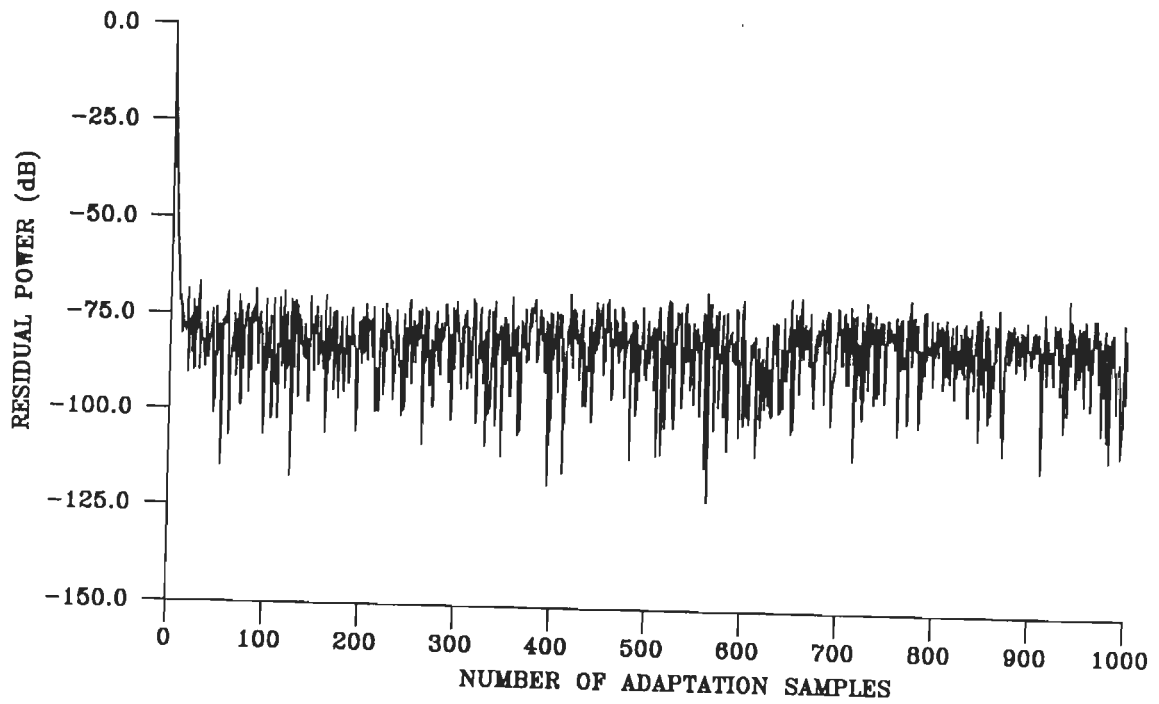
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER

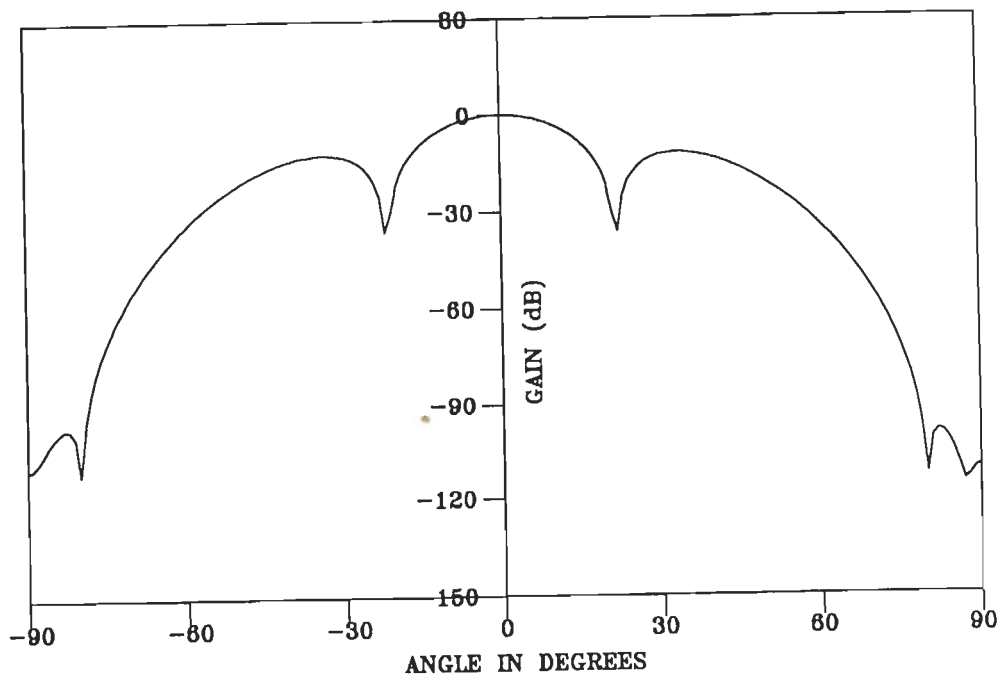


(c) RMGSEF BEAMFORMER

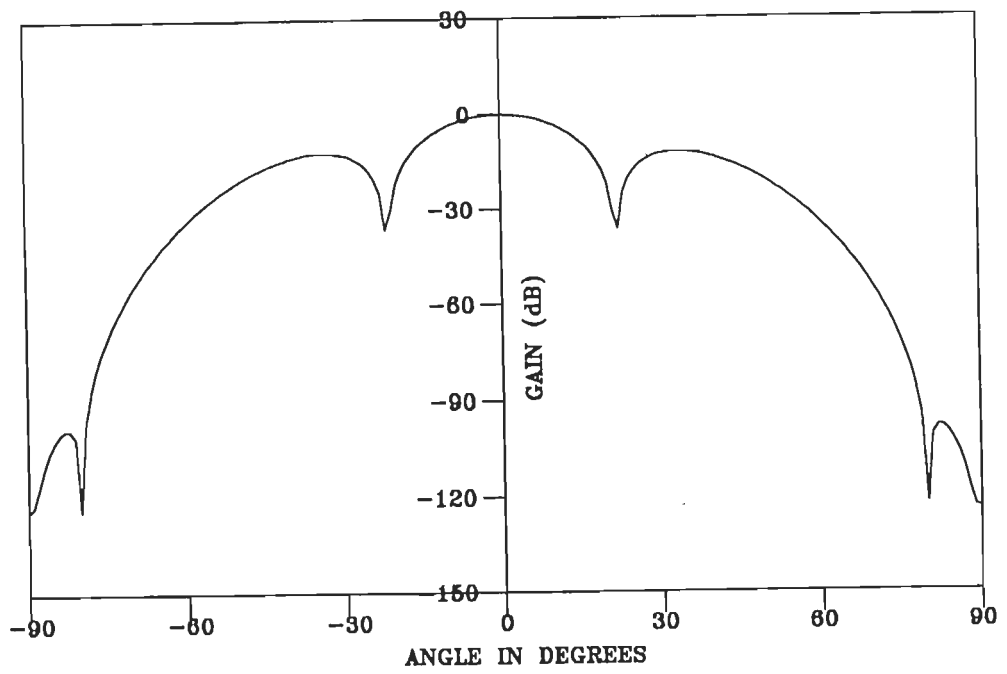


(d) QRD-LS BEAMFORMER

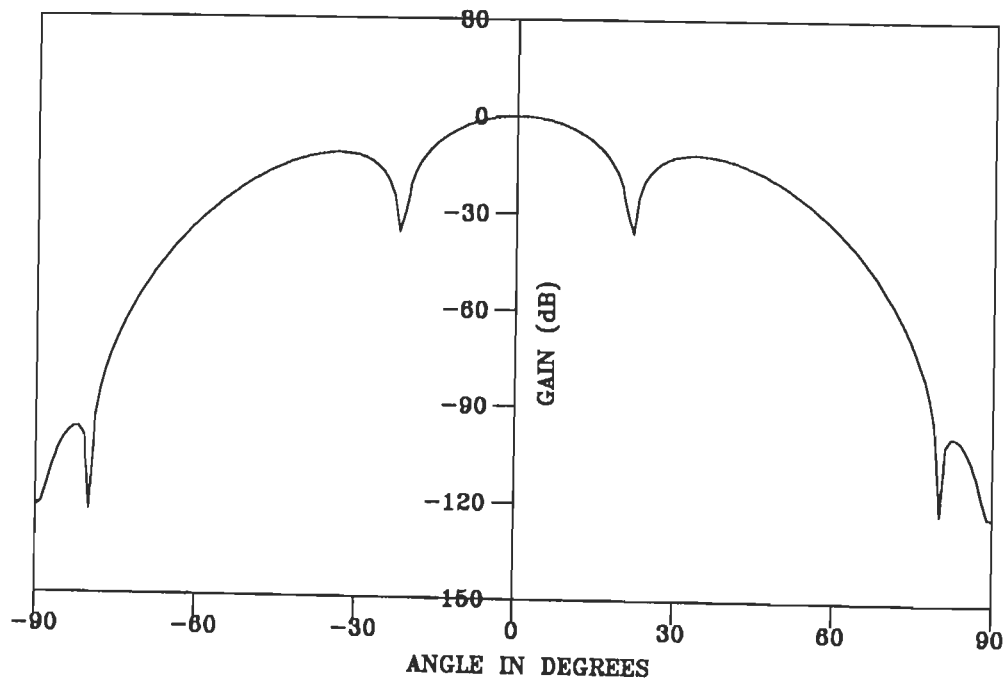
FIG.3.20 CONVERGENCE CHARACTERISTICS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT 80° AND -80°.



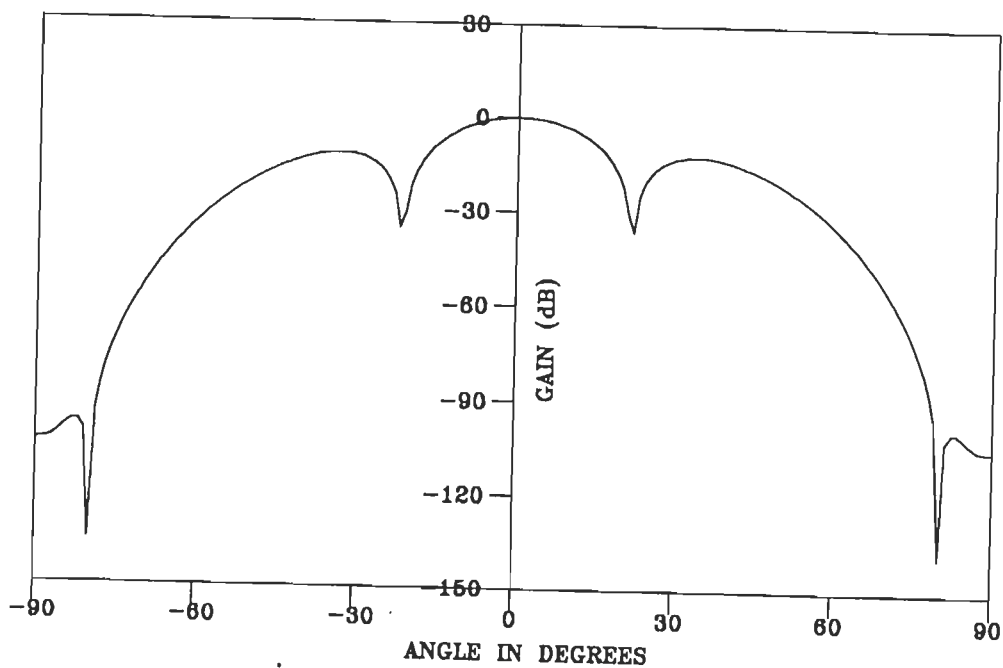
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER



(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

FIG.3.21 VOLTAGE PATTERNS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $80^\circ$  &  $-80^\circ$ .



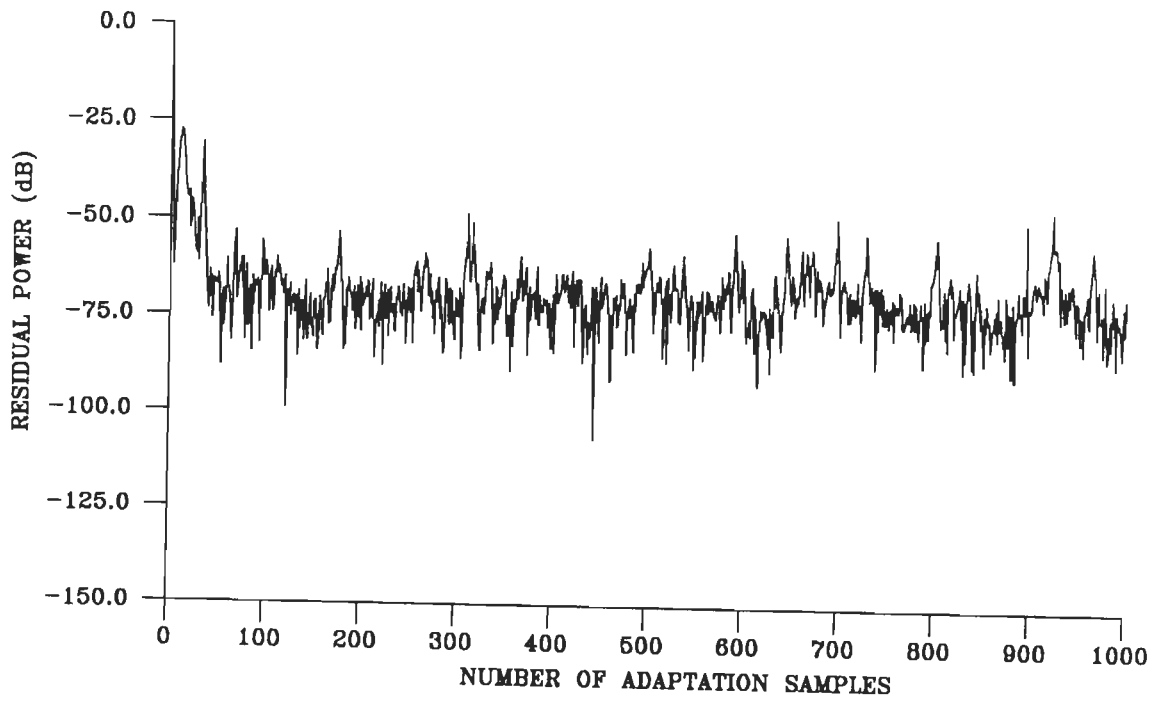
Table -3.12

Parameters of the Broadband Interferences in Example 3.6-2.5

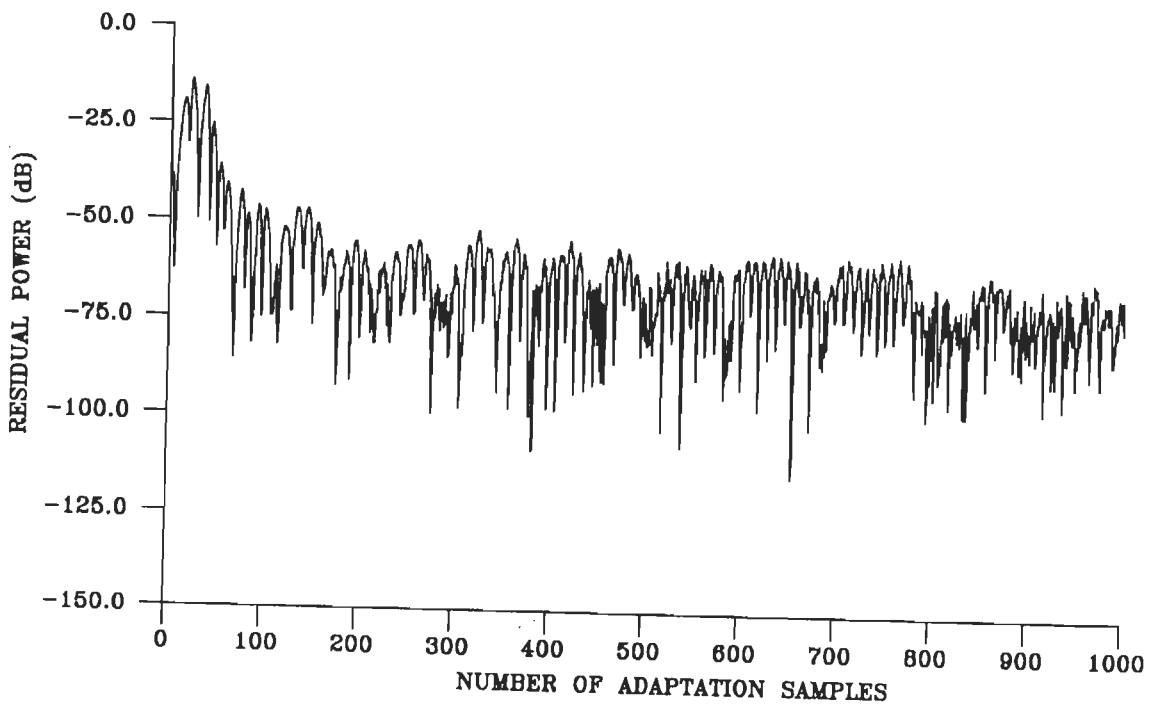
Parameter	Interference 1	Interference 2	Interference 3	Interference 4
$S_i$	10.0	10.0	10.0	10.0
$\theta_i$	$30^\circ$	$-30^\circ$	$60^\circ$	$-60^\circ$
$\omega_i$	1.0	1.2	1.1	0.8
$\Delta_i$	0.8	1.0	0.8	0.6

Fig.3.22 shows the residual power characteristics for the four beamformers. On comparison with Fig.3.5, it is evident that the RLS beamformer (Fig.3.22(a)) exhibits faster convergence in the broadband signal environment. Further the residual power does not show any increasing trend which was observed in the narrowband case. In other words, RLS beamformer exhibits better numerical stability in broadband signal environment. The other three beamformers viz, the RMGS, RMGSEF and the QRD-LS exhibit the same convergence characteristics irrespective of the signal environment (Fig.3.5(b),(c) and (d)).

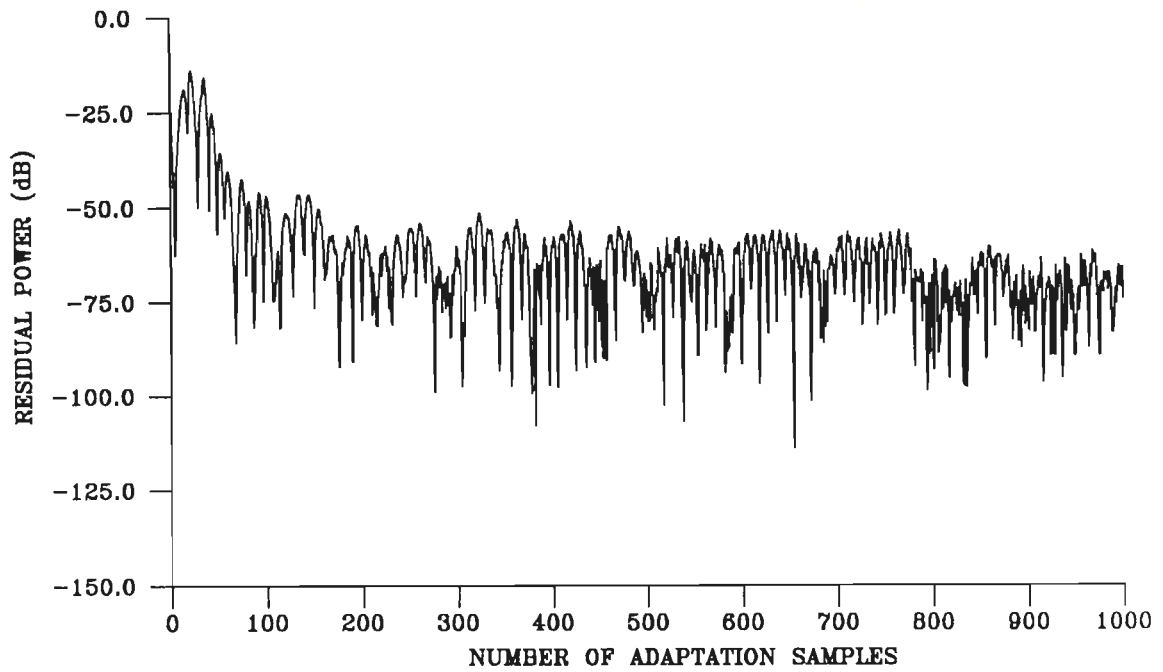
A comparison of the voltage patterns of broadband beamformers (Fig.3.23) with those of narrowband beamformers (Fig.3.6), shows that while the interference suppression is satisfactory in both the cases, the depth of nulls in the broadband case is about 10dB less than that in the narrowband case.



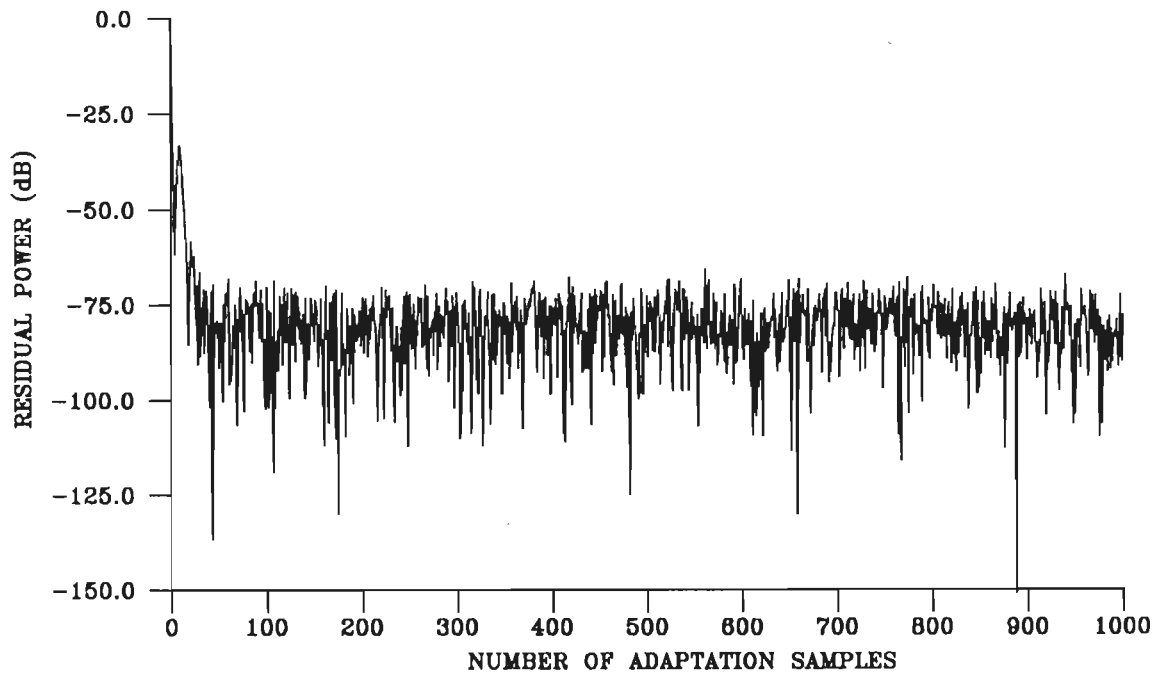
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER

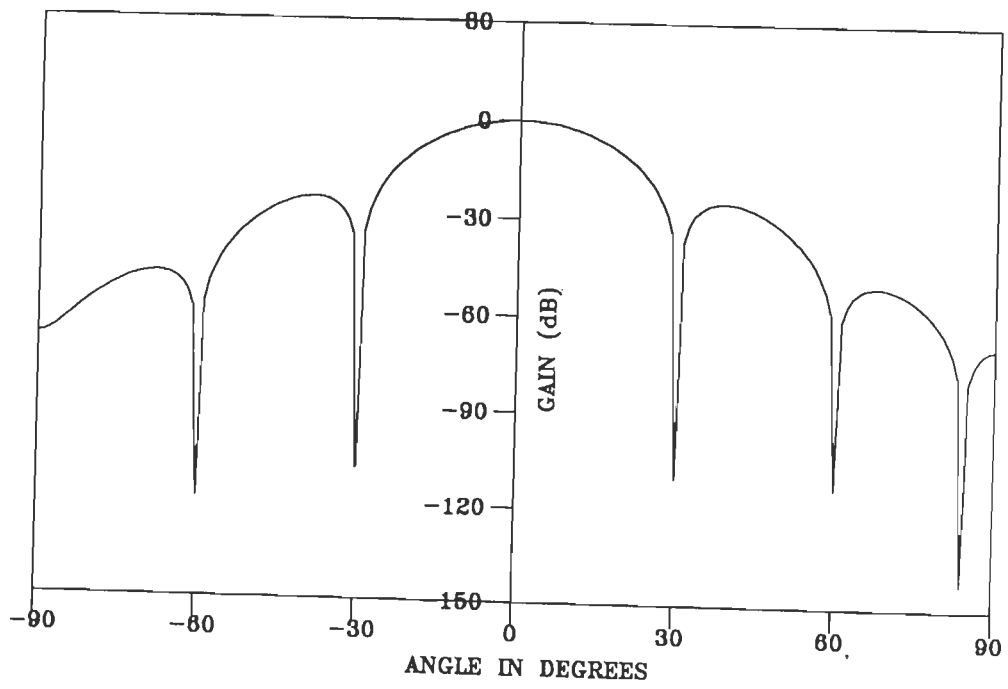


(c) RMGSEF BEAMFORMER

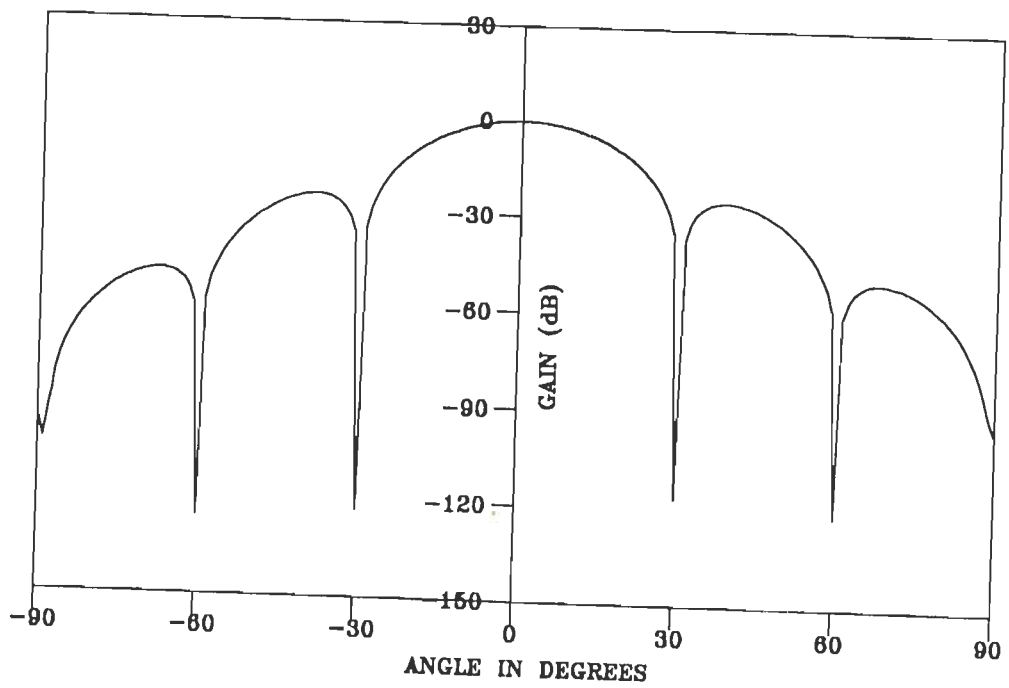


(d) QRD-LS BEAMFORMER

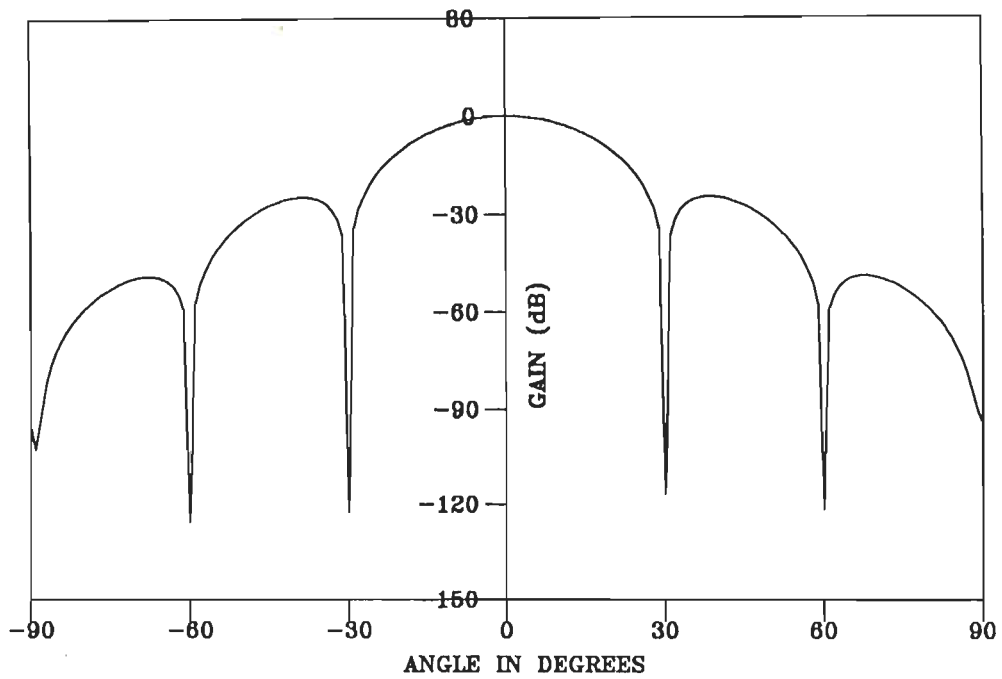
FIG.3.22 CONVERGENCE CHARACTERISTICS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ$ ,  $-30^\circ$ ,  $60^\circ$  AND  $-60^\circ$ .



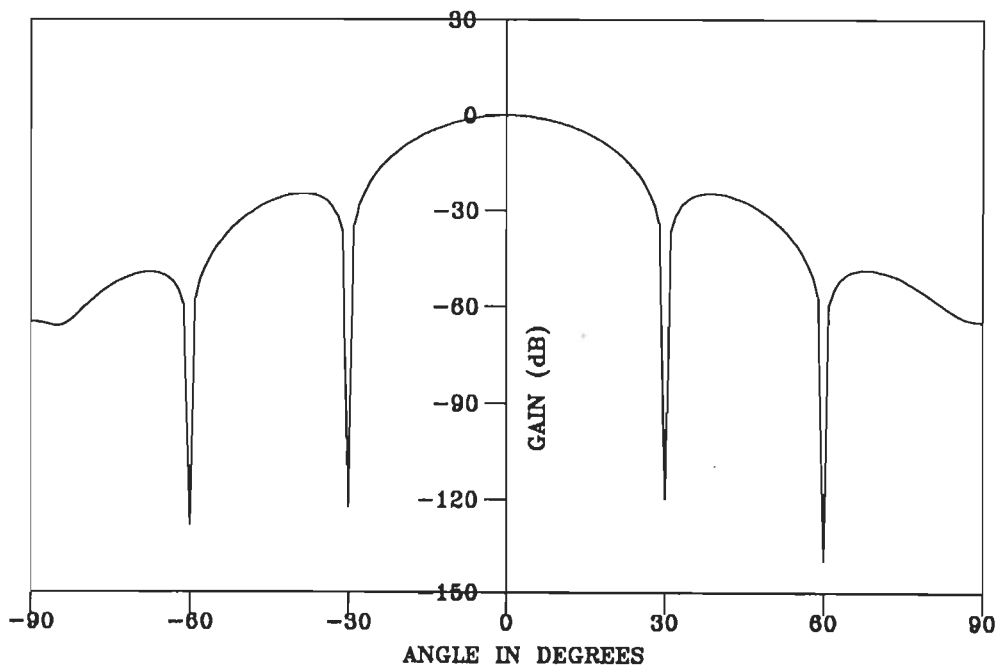
(a) RLS BEAMFORMER



(b) RMGS BEAMFORMER



(c) RMGSEF BEAMFORMER



(d) QRD-LS BEAMFORMER

FIG.3.23 VOLTAGE PATTERNS OF BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ$ ,  $-30^\circ$ ,  $60^\circ$ , AND  $-60^\circ$ .

### 3.7 CONCLUSIONS

In this chapter, a detailed study has been carried out on the adaptive beamformers based on exact least-square algorithms, viz., the RLS, RMGS, RMGSEF and the QRD-LS algorithm. Both narrowband and broadband arrays have been investigated by considering several typical signal environments in the computer simulations. The results of these investigations can be summarized as follows.

Of the four beamformers considered here, the RLS beamformer has been found to give the poorest performance. It suffers from numerical instability, which is evident from a study of residual power plots for narrowband, and in certain situations, fails to place nulls in the directions of the interferences. For example, when a large number of interferences are present, or when interferences arrive simultaneously from directions close to the desired signal as well as from near end-fire, the RLS beamformer fails to null the end-fire interferences. The performance is better in broadband signal environment, so far as numerical stability is concerned, but the convergence is slow when interferences arrive very close to the desired signal.

On the other hand, the beamformers based on QR decomposition techniques viz, the RMGS, RMGSEF and the QRD-LS algorithms, exhibit good numerical stability and place deep nulls in the direction of interferences, in both narrowband and broadband signal environments. A better numerical stability is expected here since all the calculations are carried out in data matrix domain and involve only scalar operations. Of these three beamformers, the QRD-LS beamformer exhibits the best performance in terms of convergence rate and the depth of

nulls. Although RMGS and RMGSEF beamformers have a slower convergence rate relative to QRD-LS, they exhibit a comparable interference-suppression ability in all the signal environments considered here.

All the four beamformers considered here have a computational complexity on the order of  $O(P^2)$ . Table-3.13 compares the number of operations required per time sample, in each case. In the case of QRD-LS and RMGS beamformers another  $P(P-1)/2$  operations are needed to compute the weight vector  $\underline{W}$ .

**Table-3.13**  
**Computational Complexity of Exact Least-Square Beamformers**

RLS	QRD-LS	RMGS	RMGSEF
$3P^2+3P$	$2.5P^2+6.5P$	$1.5P^2+4.5P$	$1.5P^2+7.5P$
$+(P^2+2P)$	+2P divisions	$+(P^2+3P)$	$+(0.5P^2+1.5P)$
divisions	+P square-roots	divisions	divisions

It is clear from Table-3.13 that the QRD-LS beamformer has the highest computational complexity as it involves 'P' square-root operations. In the case of broadband beamformers, where the computational complexity is  $O(P^2M^2)$ , the number of square-root operations, increases to 'PM' which makes it computationally very expensive. On the other hand, the RMGS class of beamformers have the least computational complexity of all and at the same time, exhibit an interference-suppression capability which is comparable to that of QRD-LS beamformer. Therefore, it may be concluded that the RMGS class

of beamformers, in particular the RMGSEF beamformer, offers a good compromise between good performance and computational complexity in most adaptive beamforming applications.



## CHAPTER - 4

### FAST RECURSIVE LEAST-SQUARE ALGORITHMS FOR ADAPTIVE BEAMFORMERS

In the previous chapter, adaptive arrays based on recursive least-square algorithms, viz, the RLS, the QRD-LS algorithm based on Givens rotations and the RMGS algorithms have been discussed. These algorithms have a computational complexity of the  $O(P^2)$  in narrowband beamforming problems. The complexity increases to the  $O(P^2M^2)$  in the case of broadband arrays, where 'M' is the number of taps in each tapped delay line.

Adaptive filters for broadband beamforming are two dimensional filters with one dimension being space and the other dimension being time. The filtering in time dimension is a simple convolution and, hence, fast algorithms can exploit the computational redundancy in this dimension [58]. Corresponding to the two filter structures, there are two families of such fast algorithms ; the fast lattice [12] and the fast transversal filter [FTF] algorithms [6]. The broadband beamforming application requires the multichannel formulation of the fast lattice and the FTF algorithms.

The fast lattice algorithm solves the general adaptive filtering problem using both order and time recursion. As a consequence, the number of sections can be easily increased or decreased without affecting the parameters of the remaining sections. On the other hand, the FTF algorithm is fixed in order and solves the filtering problem recursively in time.

Although different multichannel least-square lattice [MLSL] algorithms have been developed [28,39,32,35,68] and discussed in the



context of adaptive filtering, their application to the adaptive beamforming problem has received little attention in the scientific literature. Lee et al [31] have used the multichannel lattice filter to realize a generalized sidelobe cancellor. In their scheme, the P-by-1 forward and backward prediction error vectors at the  $m^{\text{th}}$  stage and the P-by-P forward and backward coefficient matrices are recursively updated, using the LMS approach to minimize the mean squared values of local errors. However, the multichannel lattice algorithm in its exact form has not been applied to the adaptive beamforming problem, so far.

Slock and Kailath [58], have proposed the use of multichannel FTF algorithm for adaptive broadband beamforming. Using the geometric approach, they have derived a modified version of the algorithm that is suitable for parallel implementation. However, they have not provided any simulation results.

The application of QRD-LS algorithm using Givens rotations to the narrowband beamforming, and its extension to broadband beamforming, has been discussed in the previous chapter. It was shown that the algorithm has excellent numerical properties but is computationally expensive. Recently, there has been an increasing interest in QRD-LS rotation-based fast-RLS algorithms, since they combine the advantages of both the algorithms, i.e., the numerical stability of the QRD-LS algorithm and reduced computational complexity of the lattice algorithms.

McWhirter and Proudler [40] have recently proposed least-square lattice algorithm for broadband beamforming which has at its root, the Givens rotation based QRD-LS algorithm. The algorithm

also contains within it, the QRD-based lattice algorithm for solving multichannel least-square linear prediction.

Yang and Bohme [67] have shown how it is possible to transform a conventional Lattice algorithm into a QRD-based one. Following Lewis [35] and McWhirter [40], they transform a multichannel lattice algorithm into one, composed only of orthogonal operations. They achieve this in two steps : (i) transformation from covariance domain to data domain by the use of Cholesky decomposition (ii) using time recursive QRD update technique [62] to incorporate new data into the square-root factor produced by Cholesky decomposition.

Since a detailed study of beamformers based on these algorithms is lacking in literature, in this chapter, we present the adaptive beamformers based on the multichannel least-square lattice, the hybrid multichannel QR-lattice [QR-MLSL] and the multichannel FTF algorithms. We first derive these algorithms using the algebraic approach, which has not been reported so far, and then compare their performance on the basis of computer simulation results.

#### 4.1 THE MULTICHANNEL LEAST-SQUARE LATTICE [MLSL] ALGORITHM

The basic least-square problem to be solved in the broadband beamformer is to find a set of PM dimensional vector  $\underline{w}_{-PM}(n)$  which minimizes the exponentially weighted sum of squared errors

$$\xi(n) = \sum_{j=1}^n \lambda^{n-j} |e_M(j)|^2 \quad \dots (4.1)$$

where  $e_M(j)$  is the error in the estimation of the desired signal  $d(j)$  and is given by



Similarly,  $\eta_M(n)$  denotes the a priori error and is defined as

$$\eta_M(n) = d(n) - \underline{W}_{PM}^H(n-1) \underline{X}_{PM}(n) \quad \dots(4.8)$$

The multichannel least-square lattice algorithm [MLSL] solves eqn.(4.1) recursively, in both order and time, in terms of  $e_m(j)$ ,  $m = 1, 2, \dots, M$ , at each instant of time. It may be noted that

$$e_m(j) = d(j) - \underline{W}_{Pm}^H(j) \underline{X}_{Pm}(j) \quad \dots(4.9)$$

Where  $\underline{W}_{Pm}(j)$  minimizes the exponentially weighted sum of squared errors defined in eqn.(4.1) for  $M = m$ , and  $\underline{X}_{Pm}(j)$  is defined as

$$\underline{X}_{Pm}(j) = \begin{bmatrix} x_1(j), x_2(j), \dots, x_p(j), \\ x_1(j-1), x_2(j-1), \dots, x_p(j-1), \\ \dots, \dots, \dots, \dots, \\ x_1(j-m+1), x_2(j-m+1), \dots, x_p(j-m+1) \end{bmatrix}^T \quad \dots(4.10)$$

The multichannel least-square lattice algorithm is an extension of the single channel least-square lattice algorithm considered in [22]. The extended data vector  $\underline{X}_{P(m+1)}(n)$  can be partitioned as

$$\underline{X}_{P(m+1)}(j) = \begin{bmatrix} \underline{X}(j) \\ \underline{X}_{Pm}(j-1) \end{bmatrix} \quad \dots(4.11a)$$

or

$$\underline{X}_{P(m+1)}(j) = \begin{bmatrix} \underline{X}_{Pm}(j) \\ \underline{X}(j-m) \end{bmatrix} \quad \dots(4.11b)$$

where,

$$\underline{X}(j) = \left[ x_1(j), x_2(j), \dots, x_p(j) \right]^T \quad \dots (4.12a)$$

$$\begin{aligned} \underline{X}_{Pm}(j-1) = & \left[ x_1(j-1), x_2(j-1), \dots, x_p(j-1), \right. \\ & x_1(j-2), x_2(j-2), \dots, x_p(j-2), \\ & \dots \\ & \left. x_1(j-m), x_2(j-m), \dots, x_p(j-m) \right]^T \quad \dots (4.12b) \end{aligned}$$

$$\begin{aligned} \underline{X}_{Pm}(j) = & \left[ x_1(j), x_2(j), \dots, x_p(j), \right. \\ & x_1(j-1), x_2(j-1), \dots, x_p(j-1), \\ & \dots \\ & \left. x_1(j-m+1), x_2(j-m+1), \dots, x_p(j-m+1) \right]^T \\ & \dots (4.12c) \end{aligned}$$

$$\underline{X}(j-m) = \left[ x_1(j-m), x_2(j-m), \dots, x_p(j-m) \right]^T \quad \dots (4.12d)$$

Using the partition properties of  $\underline{X}_{P(m+1)}(n)$ , given in eqn.(4.11), the autocorrelation matrix  $\Phi_{P(m+1)}(n)$  can be partitioned as

$$\Phi_{P(m+1)}(n) = \begin{bmatrix} q(n) & Q_m^H(n) \\ Q_m(n) & \Phi_{Pm}(n-1) \end{bmatrix} \quad \dots (4.13a)$$

$$\Phi_{P(m+1)}(n) = \begin{bmatrix} \Phi_{Pm}(n) & U_m(n) \\ U_m^H(n) & u(n) \end{bmatrix} \quad \dots (4.13b)$$

where P-by-P Hermitian matrices  $q(n)$ ,  $u(n)$  and the P(m-1)-by-P matrices  $Q_m(n)$ ,  $U_m(n)$  are given by

$$q(n) = \sum_{j=1}^n \lambda^{n-j} \underline{X}(j) \underline{X}^H(j) \quad \dots (4.14a)$$

$$u(n) = \sum_{j=1}^n \lambda^{n-j} \underline{X}(j-m) \underline{X}^H(j-m) \quad \dots (4.14b)$$

$$Q_m(n) = \sum_{j=1}^n \lambda^{n-j} \underline{X}_{Pm}(j-1) \underline{X}^H(j) \quad \dots (4.15a)$$

$$U_m(n) = \sum_{j=1}^n \lambda^{n-j} \underline{X}_{Pm}(j) \underline{X}^H(j-m). \quad \dots (4.15b)$$

We next define the  $m^{\text{th}}$  order P-by-1 forward and backward error residual vectors,  $\underline{e}_m^f(n)$  and  $\underline{e}_m^b(n)$ , as

$$\underline{e}_m^f(n) = \underline{X}(n) - A_{Pm}^H(n) \underline{X}_{Pm}(n-1) \quad \dots (4.16a)$$

$$\underline{e}_m^b(n) = \underline{X}(n-m) - B_{Pm}^H(n) \underline{X}_{Pm}(n). \quad \dots (4.16b)$$

In the above equations the components of Pm-by-P matrices  $A_{Pm}(n)$  and  $B_{Pm}(n)$  are called one-step-forward and one-step-backward multichannel predictor coefficients and are chosen so as to minimize

$$\text{Trace of } \left[ \sum_{j=1}^n \lambda^{n-j} \underline{e}_m^f(j) \left[ \underline{e}_m^f(j) \right]^H \right] \quad \dots (4.17a)$$

and

$$\text{Trace of } \left[ \sum_{j=1}^n \lambda^{n-j} \underline{e}_m^b(j) \left[ \underline{e}_m^b(j) \right]^H \right] \quad \dots (4.17b)$$

with respect to  $A_{P_m}(n)$  and  $B_{P_m}(n)$ , respectively.

Using the results given in Appendix-A, eqn.(4.17a) can be minimized to obtain

$$\Phi_{P_m}(n-1) A_{P_m}^H(n) - Q_m(n) = 0 \quad \dots(4.18)$$

and  $r_m^f(n)$ , the error variance matrix of the forward error vector  $e_m^f(n)$  is given by

$$r_m^f(n) = q(n) - Q_m^H(n) A_{P_m}(n) \quad \dots(4.19)$$

Equations (4.18) and (4.19) can be combined into a single augmented equation as,

$$\begin{bmatrix} q(n) & Q_m^H(n) \\ Q_m(n) & \Phi_{P_m}(n-1) \end{bmatrix} \begin{bmatrix} I \\ -A_{P_m}(n) \end{bmatrix} = \begin{bmatrix} r_m^f(n) \\ 0 \end{bmatrix} \quad \dots(4.20)$$

Where 0 is a null matrix of dimension  $P(m-1)$ -by- $P$ .

Equations similar to (4.18)-(4.20) are obtained by minimizing (4.17b) with respect to  $B_{P_m}(n)$ .

Using matrix inversion lemma [48], the inverse of  $\Phi_{P(m+1)}(n)$  may be written as

$$\Phi_{P(m+1)}^{-1}(n) = \begin{bmatrix} \Phi_{P_m}^{-1}(n) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -B_{P_m}(n) \\ I \end{bmatrix} \left[ r_m^b(n) \right]^{-1} \begin{bmatrix} -B_{P_m}^H(n) & I \end{bmatrix} \quad \dots(4.21a)$$

or

$$\Phi_{P(m+1)}^{-1}(n) = \begin{bmatrix} 0 & 0 \\ 0 & \Phi_{P_m}^{-1}(n-1) \end{bmatrix} + \begin{bmatrix} I \\ -A_{P_m}(n) \end{bmatrix} \left[ r_m^f(n) \right]^{-1} \begin{bmatrix} I & -A_{P_m}^H(n) \end{bmatrix} \quad \dots(4.21b)$$



Substituting (n-1) for n in eqn. (4.21a) and premultiplying by  $Q_{m+1}(n)$ , we get the following order update equation for  $A_{Pm}(n)$

$$A_{P(m+1)}(n) = \begin{bmatrix} A_{Pm}(n) \\ 0 \end{bmatrix} + \begin{bmatrix} -B_{Pm}(n-1) \\ I \end{bmatrix} \left[ r_m^b(n-1) \right]^{-1} K_m(n) \quad \dots (4.22)$$

where  $K_m(n)$  is a P-by-P matrix, defined as

$$K_m(n) = \begin{bmatrix} -B_{Pm}^H(n-1) & I \end{bmatrix} Q_{m+1}(n). \quad \dots (4.23)$$

Similarly, the order update equation for  $B_{Pm}(n)$  is obtained as

$$B_{P(m+1)}(n) = \begin{bmatrix} 0 \\ B_{Pm}(n-1) \end{bmatrix} + \begin{bmatrix} I \\ -A_{Pm}(n) \end{bmatrix} \left[ r_m^f(n) \right]^{-1} \hat{K}_m(n) \quad \dots (4.24)$$

where

$$\hat{K}_m(n) = \begin{bmatrix} I & -A_{Pm}^H(n) \end{bmatrix} U_{m+1}(n). \quad \dots (4.25)$$

It may be easily shown that

$$\hat{K}_m(n) = K_m^H(n). \quad \dots (4.26)$$

The order update recursions for the forward and backward error residual vectors  $e_{m+1}^f(n)$  and  $e_{m+1}^b(n)$  can be derived by substituting eqn. (4.22) in (4.16a) and eqn. (4.24) in (4.16b) and are given by

$$e_{m+1}^f(n) = e_m^f(n) - K_m^H(n) \left[ r_m^b(n-1) \right]^{-1} e_m^b(n-1) \quad \dots (4.27a)$$

$$e_{m+1}^b(n) = e_m^b(n-1) - K_m(n) \left[ r_m^f(n) \right]^{-1} e_m^f(n) \quad \dots (4.27b)$$

The  $(m+1)^{\text{th}}$  order forward and backward error variance matrices can be written as

$$r_{m+1}^f(n) = r_m^f(n) - K_m^H(n) \left[ r_m^b(n-1) \right]^{-1} K_m(n) \quad \dots (4.28a)$$

$$r_{m+1}^b(n) = r_m^b(n-1) - K_m(n) \left[ r_m^f(n) \right]^{-1} K_m^H(n) \quad \dots (4.28b)$$

The two recursive equations in (4.27) specify the multichannel lattice filter. The initial conditions on the order updates are

$$\underline{e}_0^f(n) = \underline{e}_0^b(n) = \underline{X}(n) \quad \dots (4.29)$$

$$\begin{aligned} r_0^f(n) = r_0^b(n) &= \sum_{j=1}^n \lambda^{n-j} \underline{X}(j) \underline{X}^H(j) \\ &= \lambda r_0^f(n-1) + \underline{X}(n) \underline{X}^H(n) \end{aligned} \quad \dots (4.30)$$

We next derive the time update equation for  $K_m(n)$ , using

$$K_m(n) = U_{m+1}^H(n) \begin{bmatrix} I \\ -A_{Pm}(n) \end{bmatrix} \quad \dots (4.31)$$

This requires a time update equation for  $A_{Pm}(n)$  which may be written as [60]

$$A_{Pm}(n) = A_{Pm}(n-1) + C_{Pm}(n-1) \left[ \underline{e}_m^f(n) \right]^H \quad \dots (4.32)$$

Similarly, we can also write

$$B_{Pm}(n) = B_{Pm}(n-1) + C_{Pm}(n) \left[ \underline{e}_m^b(n) \right]^H \quad \dots (4.33)$$

In eqns.(4.32) and (4.33),  $\underline{C}_{Pm}(n)$  is a complex vector given by

$$\underline{C}_{Pm}(n) = \Phi_{Pm}^{-1}(n) \underline{X}_{Pm}(n). \quad \dots(4.34)$$

Using equation (4.21a) in (4.34), we get

$$\underline{C}_{Pm}(n) = \begin{bmatrix} \underline{C}_{P(m-1)}(n) \\ 0 \end{bmatrix} + \begin{bmatrix} -B_{P(m-1)}(n) \\ I \end{bmatrix} \left[ r_m^b(n) \right]^{-1} \underline{e}_m^b(n). \quad \dots(4.35)$$

Using eqn.(4.32) in (4.31),  $K_m(n)$  may be rewritten as,

$$K_m(n) = \lambda K_m(n-1) + \frac{\underline{e}_m^b(n-1) \left[ \underline{e}_m^f(n) \right]^H}{\alpha_m(n-1)} \quad \dots(4.36)$$

where  $\alpha_m(n)$  is a scalar real constant, given by

$$\alpha_m(n) = \alpha_{m-1}(n) - \left[ \underline{e}_{m-1}^b(n) \right]^H \left[ r_{m-1}^b(n-1) \right]^{-1} \underline{e}_{m-1}^b(n) \quad \dots(4.37)$$

In order to complete the derivation of multichannel least-square lattice algorithm, eqn.(4.5) should be solved recursively, which may be rewritten as,

$$\underline{W}_{Pm}(n) = \Phi_{Pm}^{-1}(n) \underline{\theta}_{Pm}(n). \quad \dots(4.38)$$

Substituting for  $\Phi_{Pm}^{-1}(n)$  from eqn.(4.21a), we get

$$\underline{W}_{Pm}(n) = \begin{bmatrix} \underline{W}_{P(m-1)}(n) \\ 0 \end{bmatrix} + \begin{bmatrix} -B_{P(m-1)}(n) \\ I \end{bmatrix} \left[ r_{m-1}^b(n) \right]^{-1} \underline{d}_{m-1}(n) \quad \dots(4.39)$$

where

$$\underline{d}_{m-1}(n) = \begin{bmatrix} -B_{P(m-1)}^H & I \end{bmatrix} \underline{\theta}_{-Pm}(n) \quad \dots(4.40)$$

The P-by-1 dimensional vector  $\underline{d}_m(n)$  satisfies a time update equation which is obtained from the time update equations for  $B_{Pm}(n)$  and  $\underline{\theta}_{-Pm}(n)$ , given by (4.33) and (4.7) respectively. Thus,

$$\underline{d}_m(n) = \lambda \underline{d}_m(n-1) + \underline{e}_m^b(n) e_m^*(n) / \alpha_m(n). \quad \dots(4.41)$$

With  $e_o(n) = d(n)$ , the order update equation for the error residual of the multichannel least-square lattice filter can be written as,

$$e_m(n) = d(n) - \underline{W}_{-Pm}^H(n) \underline{X}_{-Pm}(n) \quad \dots(4.42)$$

Substituting eqn.(4.39) for  $\underline{W}_{-Pm}(n)$ , the above equation can be simplified as

$$e_m(n) = e_{m-1}(n) - \underline{d}_{m-1}^H(n) \left[ \underline{r}_{m-1}^b(n) \right]^{-1} e_{m-1}^b(n). \quad \dots(4.43)$$

Thus, we have obtained the order and time update relations for the multichannel least-square lattice algorithm based beamformer, which are summarized below.

$$K_m(n) = \lambda K_m(n-1) + \frac{\underline{e}_m^b(n-1) \left[ \underline{e}_m^f(n) \right]^H}{\alpha_m(n-1)} \quad \dots(4.36)$$

$$\underline{e}_{m+1}^f(n) = \underline{e}_m^f(n) - K_m^H(n) \left[ \underline{r}_m^b(n-1) \right]^{-1} \underline{e}_m^b(n-1) \quad \dots(4.27a)$$

$$\underline{e}_{m+1}^b(n) = \underline{e}_m^b(n-1) - K_m(n) \left[ r_m^f(n) \right]^{-1} \underline{e}_m^f(n) \quad \dots (4.27b)$$

$$r_{m+1}^f(n) = r_m^f(n) - K_m^H(n) \left[ r_m^b(n-1) \right]^{-1} K_m(n) \quad \dots (4.28a)$$

$$r_{m+1}^b(n) = r_m^b(n-1) - K_m(n) \left[ r_m^f(n) \right]^{-1} K_m^H(n) \quad \dots (4.28b)$$

$$\alpha_{m+1}(n) = \alpha_m(n) - \left[ \underline{e}_m^b(n) \right]^H \left[ r_m^b(n-1) \right]^{-1} \underline{e}_m^b(n) \quad \dots (4.37)$$

$$\underline{d}_m(n) = \lambda \underline{d}_m(n-1) + \underline{e}_m^b(n) \underline{e}_m^*(n) / \alpha_m(n) \quad \dots (4.41)$$

$$e_{m+1}(n) = e_m(n) - d_m^H(n) \left[ r_m^b(n) \right]^{-1} \underline{e}_m^b(n) \quad \dots (4.43)$$

Here, the order update equations for  $r_{m+1}^f(n)$  and  $r_{m+1}^b(n)$  are computationally expensive as they involve multiplication of three P-by-P matrices. Moreover, these equations may cause numerical instability as they use the inverses of  $r_m^b(n-1)$  and  $r_m^f(n)$  for order updating  $r_{m+1}^f(n)$  and  $r_{m+1}^b(n)$ . Therefore, a better strategy is to use their respective time update equations.

We obtain the time update equations for  $r_m^f(n)$ , by substituting for  $A_{Pm}(n)$  from (4.32) in eqn.(4.19). This gives

$$r_m^f(n) = \lambda r_m^f(n-1) + \frac{\underline{e}_m^f(n) \left[ \underline{e}_m^f(n) \right]^H}{\alpha_m(n-1)} \quad \dots (4.44)$$

Similarly we can write

$$r_m^b(n) = \lambda r_m^b(n-1) + \frac{\underline{e}_m^b(n) \left[ \underline{e}_m^b(n) \right]^H}{\alpha_m(n-1)} \quad \dots (4.45)$$

The above time update equations do not involve either matrix multiplications or the inverse of  $r_m^f(n)$  and  $r_m^b(n-1)$ . Therefore, these equations are superior with respect to both computational complexity and numerical stability [67].

Using equations (4.44) and (4.45), the multichannel least-square lattice algorithm can be rewritten in the following form.

$$r_m^f(n) = \lambda r_m^f(n-1) + \frac{e_{-m}^f(n) \left[ e_{-m}^f(n) \right]^H}{\alpha_m(n-1)} \quad \dots (4.44)$$

$$r_m^b(n-1) = \lambda r_m^b(n-2) + \frac{e_{-m}^b(n-1) \left[ e_{-m}^b(n-1) \right]^H}{\alpha_m(n-2)} \quad \dots (4.45)$$

$$K_m(n) = \lambda K_m(n-1) + \frac{e_{-m}^b(n-1) \left[ e_{-m}^f(n) \right]^H}{\alpha_m(n-1)} \quad \dots (4.36)$$

$$e_{-m+1}^f(n) = e_{-m}^f(n) - K_m^H(n) \left[ r_m^b(n-1) \right]^{-1} e_{-m}^b(n-1) \quad \dots (4.27a)$$

$$e_{-m+1}^b(n) = e_{-m}^b(n-1) - K_m(n) \left[ r_m^f(n) \right]^{-1} e_{-m}^f(n) \quad \dots (4.27b)$$

$$\alpha_{m+1}(n) = \alpha_m(n) - \left[ e_{-m}^b(n) \right]^H \left[ r_m^b(n-1) \right]^{-1} e_{-m}^b(n) \quad \dots (4.37)$$

$$d_{-m}(n) = \lambda d_{-m}(n-1) + e_{-m}^b(n) e_m^*(n) / \alpha_m(n) \quad \dots (4.41)$$

$$e_{m+1}(n) = e_m(n) - d_{-m}^H(n) \left[ r_m^b(n) \right]^{-1} e_{-m}^b(n) \quad \dots (4.43)$$

Modifications can be made to some of the above equations without affecting the optimality of the algorithm. In the conventional method, the reflection coefficients and the ladder gain are computed according to the following relations.

$$\Delta_{m+1}^f(n) = -K_m^H(n) \left[ r_m^b(n-1) \right]^{-1} \quad \dots (4.46)$$

$$\Delta_{m+1}^b(n) = -K_m(n) \left[ r_m^f(n) \right]^{-1} \quad \dots (4.47)$$

$$\underline{\varepsilon}_m(n) = -\underline{d}_m^H(n) \left[ r_m^b(n) \right]^{-1} \quad \dots (4.48)$$

In MLSL algorithm, the weight vector is not computed explicitly as it is computationally costlier. However, MLSL algorithm can be used for broadband adaptive beamforming, since the error residue  $e_{m+1}(n)$ , for  $m + 1 = M$  is available which is of primary interest. The MLSL algorithm and its initialization are given in Table 4.1 and Table 4.2, respectively.

**Table-4.1**

**The MLSL Algorithm Using A Posteriori Errors**

Algorithm	Complexity
For $n = 1, 2, \dots$	
For $m = 0, \dots, M-1$	
$r_m^f(n) = \lambda r_m^f(n-1) + \frac{e_m^f(n) \left[ e_m^f(n) \right]^H}{\alpha_m(n-1)}$	$P^2M$ (T 4.1.1)

$$r_m^b(n-1) = \lambda r_m^b(n-2) + \frac{e_m^b(n-1) \left[ e_m^b(n-1) \right]^H}{\alpha_m(n-2)} \quad P^2M \quad (T 4.1.2)$$

$$K_m(n) = \lambda K_m(n-1) + \frac{e_m^b(n-1) \left[ e_m^f(n) \right]^H}{\alpha_m(n-1)} \quad P^2M \quad (T 4.1.3)$$

$$\Delta_{m+1}^f(n) = -K_m^H(n) \left[ r_m^b(n-1) \right]^{-1} \quad P^3M \quad (T 4.1.4)$$

$$\Delta_{m+1}^b(n) = -K_m(n) \left[ r_m^f(n) \right]^{-1} \quad P^3M \quad (T 4.1.5)$$

$$e_{m+1}^f(n) = e_m^f(n) + \Delta_{m+1}^f(n) e_m^b(n-1) \quad P^2M \quad (T 4.1.6)$$

$$e_{m+1}^b(n) = e_m^b(n-1) + \Delta_{m+1}^b(n) e_m^f(n) \quad P^2M \quad (T 4.1.7)$$

$$\alpha_{m+1}(n-1) = \alpha_m(n-1) - \left[ e_m^b(n-1) \right]^H \left[ r_m^b(n-2) \right]^{-1} e_m^b(n-1) \quad (P^2+P)M \quad (T 4.1.8)$$

Joint process estimation

For  $n = 1, 2, \dots$

For  $m = 0, \dots, M$

$$d_m(n-1) = \lambda d_m(n-2) + e_m^b(n-1) e_m^*(n-1) / \alpha_m(n-1) \quad P(M+1) \quad (T 4.1.9)$$

$$e_{m+1}(n-1) = e_m(n-1) - d_m^H(n-1) \left[ r_m^b(n-1) \right]^{-1} e_m^b(n-1) \quad (P^2+P)(M+1) \quad (T 4.1.10)$$

---

Total	4PM divisions + $2P^3M + 6P^2M + P^2(M+1) + 3P(M+1)$
-------	--

---



The MLSL algorithm requires about  $\left[ 2P^3M + 6P^2M + P^2(M+1) + 3P(M+1) \right]$  operations per time sample. In addition, it requires  $4PM$  divisions. Further, for each order, inverses of two  $P$ -by- $P$  error variance matrices  $r_m^f(n)$  and  $r_m^b(n)$  are to be computed, which requires about  $2P^3(M-1)$  operations per time step.

Table-4.2

Initialization of the MLSL Algorithm with A Posteriori Errors

---

1. To initialize the algorithm, at time  $n = 0$  set

$$K_m(0) = 0 \quad \text{for } m = 0, 1, \dots, M-1$$

$$r_m^f(0) = \delta I, \quad \delta = \text{small positive constant}$$

$$r_m^b(-1) = \delta I$$

2. At each instant,  $n \geq 1$ , generate the various zeroth order variables as

$$\underline{e}_0^f(n) = \underline{e}_0^b(n) = \underline{X}(n)$$

$$r_0^f(n) = r_0^b(n) = \lambda r_0^f(n-1) + \underline{X}(n)\underline{X}^H(n)$$

3. For joint process estimation, initialize the algorithm by setting at time  $n = 0$

$$\underline{d}_m(0) = 0$$

at each instant  $n \geq 1$ , generate the zeroth order variable

$$e_0(n) = d(n).$$


---

## 4.2 THE QR-MLSL ALGORITHM

In this section, we reformulate the MLSL algorithm derived in the previous section to a Givens rotation-based MLSL algorithm by suitable transformation of filter quantities.

In order to derive the QR-MLSL algorithm, we start with multichannel lattice recursions (4.44), (4.45), (4.36), (4.27), (4.37), (4.41) and (4.43), and write the error covariance matrices  $r_m^f(n)$  and  $r_m^b(n-1)$  in terms of their upper triangular cholesky factors  $R_m^f(n)$ ,  $R_m^b(n-1)$  as [67]

$$r_m^f(n) = \left[ R_m^f(n) \right]^H R_m^f(n) \quad \dots (4.49)$$

$$r_m^b(n-1) = \left[ R_m^b(n-1) \right]^H R_m^b(n-1) \quad \dots (4.50)$$

Following [67], we next introduce a new set of variables

$$\bar{r}_m^f(n) = \left[ R_m^f(n) \right]^{-H} K_m(n) \quad \dots (4.51a)$$

$$\bar{r}_m^b(n) = \left[ R_m^b(n-1) \right]^{-H} K_m(n) \quad \dots (4.51b)$$

$$\bar{\beta}_m^f(n) = \left[ R_m^f(n) \right]^{-H} e_m^f(n) \quad \dots (4.51c)$$

$$\bar{\beta}_m^b(n-1) = \left[ R_m^b(n-1) \right]^{-H} e_m^b(n-1) \quad \dots (4.51d)$$

$$\tilde{\alpha}_m(n) = \sqrt{\alpha_m(n)} \quad \dots (4.51e)$$

$$\tilde{e}_{-m}^f(n) = e_{-m}^f(n) / \tilde{\alpha}_m(n-1) \quad \dots (4.51f)$$

$$\tilde{e}_{-m}^b(n) = e_{-m}^b(n) / \tilde{\alpha}_m(n) \quad \dots (4.51g)$$

$$\underline{e}_{-m}^e(n-1) = \left[ R_m^b(n-1) \right]^{-H} \underline{d}_{-m}(n-1) \quad \dots (4.51h)$$

$$\tilde{e}_{m+1}(n) = e_{m+1}(n) / \tilde{\alpha}_m(n) \quad \dots (4.51i)$$

In the above equations, superscript 'H' denotes the Hermitian transposed matrix. The errors  $\tilde{e}_{-m}^f(n)$ ,  $\tilde{e}_{-m}^b(n)$  and  $\tilde{e}_m(n)$  are the geometric mean of the a posteriori and a priori errors, which are also known as 'angle normalized' or 'rotated errors' in the QR decomposition literature [67]. It is also well known that the maximum likelihood factor  $\tilde{\alpha}_m(n)$  is the conversion factor between the geometric mean and the a priori errors, which are related as

$$\eta_m(n) = \tilde{e}_m(n) / \tilde{\alpha}_m(n) \quad \dots (4.52)$$

It can be seen that, the Cholesky factors, and all the variables in eqn.(4.51), can be updated using only two orthogonal transformations. Accordingly, we construct two suitable block matrices and apply two orthogonal transformations  $\hat{Q}_m^f(n)$  and  $\hat{Q}_m^b(n)$ .

$$\begin{aligned} \hat{Q}_m^f(n) & \begin{bmatrix} \sqrt{\lambda} R_m^f(n-1) & \sqrt{\lambda} \underline{e}_{-m}^f(n-1) & \underline{0} \\ \left[ \tilde{e}_{-m}^f(n) \right]^H & \left[ \tilde{e}_{-m}^b(n-1) \right]^H & \tilde{\alpha}_m(n-1) \end{bmatrix} \\ & = \begin{bmatrix} R_m^f(n) & \underline{e}_{-m}^f(n) & \underline{\beta}_m^f(n) \\ \underline{0}^H & \left[ \tilde{e}_{-m+1}^b(n) \right]^H & \tilde{\alpha}_{m+1}(n) \end{bmatrix} \end{aligned} \quad \dots (4.53)$$

$$\hat{Q}_m^b(n) \begin{bmatrix} \sqrt{\lambda} R_m^b(n-2) & \sqrt{\lambda} \Gamma_m^b(n-1) & \sqrt{\lambda} \Gamma_m^e(n-2) & \underline{0} \\ \left[ \tilde{e}_m^b(n-1) \right]^H & \left[ \tilde{e}_m^f(n) \right]^H & \tilde{e}_m^*(n-1) & \tilde{\alpha}_m(n-1) \end{bmatrix} \\ = \begin{bmatrix} R_m^b(n-1) & \Gamma_m^b(n) & \Gamma_m^e(n-1) & \beta_m^b(n-1) \\ \underline{0}^H & \left[ \tilde{e}_{m+1}^f(n) \right]^H & \tilde{e}_{m+1}^*(n-1) & \tilde{\alpha}_{m+1}(n-1) \end{bmatrix} \quad \dots(4.54)$$

All the matrix elements appearing on the left hand side of eqns. (4.53) and (4.54) are assumed to be known from the previous time and order-recursions. As explained in chapter-3, the two orthogonal matrices  $\hat{Q}_m^f(n)$  and  $\hat{Q}_m^b(n)$  are designed to eliminate the vectors  $\left[ \tilde{e}_m^f(n) \right]^H$  and  $\left[ \tilde{e}_m^b(n-1) \right]^H$ , while keeping  $R_m^f(n)$  and  $R_m^b(n)$  upper triangular. This is accomplished with the help of Givens rotations, as in chapter 3. The resulting quantities on the right hand side of eqns. (4.53) and (4.54) are the desired updates of the corresponding positions on the LHS.

The above statement can be proved by using the orthogonality of matrices  $\hat{Q}_m^f(n)$  and  $\hat{Q}_m^b(n)$ . From the orthonormality relation  $\left[ \hat{Q}_m^f(n) \right]^H \left[ \hat{Q}_m^f(n) \right] = I_p$ , we have the following identity [67].

$$B_1^H B_2 = A_1^H A_2 \quad \text{for } \hat{Q}_m^f(n) \begin{bmatrix} A_1 & A_2 \end{bmatrix} = \begin{bmatrix} B_1 & B_2 \end{bmatrix} \quad \dots(4.55)$$

The above identity is a powerful tool in designing rotation based algorithms. By associating  $A_1$  and  $A_2$  with suitable columns of the left hand sides of the eqns. (4.53) and (4.54), we can arrive at

the MLSL recursions (4.44), (4.45), (4.36), (4.27), (4.37), (4.41) and (4.43).

For illustration, we associate both  $A_1$  and  $A_2$  with

$$\begin{bmatrix} \sqrt{\lambda} R_m^f(n-1) \\ \left[ \tilde{e}_m^f(n) \right]^H \end{bmatrix}$$

and  $B_1$  and  $B_2$  with

$$\begin{bmatrix} R_m^f(n) \\ \underline{0}^T \end{bmatrix}$$

using eqn.(4.53). Applying the identity in eqn.(4.55), we get

$$\left[ R_m^f(n) \right]^H \left[ R_m^f(n) \right] = \lambda \left[ R_m^f(n-1) \right]^H R_m^f(n-1) + \tilde{e}_m^f(n) \left[ \tilde{e}_m^f(n) \right]^H \quad \dots (4.56)$$

Substituting eqns.(4.49) and (4.51f) for  $r_m^f(n)$  and  $e_m^f(n)$ , the above equation can be simplified to

$$r_m^f(n) = \lambda r_m^f(n-1) + \frac{e_m^f(n) \left[ e_m^f(n) \right]^H}{\alpha_m(n-1)} \quad \dots (4.57)$$

Similarly, with suitable definitions of  $A_1, A_2, B_1$  and  $B_2$  (See Appendix-D), we get the following equations.

$$\left[ R_m^f(n) \right]^H \left[ R_m^f(n) \right] = \lambda \left[ R_m^f(n-1) \right]^H \left[ R_m^f(n-1) \right] + \tilde{e}_m^f(n) \left[ \tilde{e}_m^b(n-1) \right]^H \quad \dots (4.58)$$

$$\left[ R_m^f(n) \right]^H \underline{\beta}_m^f(n) = \left[ \tilde{e}_m^f(n) \right] \tilde{\alpha}_m(n-1) \quad \dots (4.59)$$

$$\left[ \tilde{r}_m^f(n) \right]^H \underline{\beta}_m^f(n) + \left[ \tilde{e}_{m+1}^b(n) \right] \tilde{\alpha}_{m+1}(n) = \tilde{e}_m^b(n-1) \tilde{\alpha}_m(n-1) \quad \dots (4.60)$$

$$\left[ \tilde{\beta}_m^f(n) \right]^H \underline{\beta}_m^f(n) + \tilde{\alpha}_{m+1}^2(n) = \tilde{\alpha}_m^2(n-1) \quad \dots (4.61)$$

Using eqns. (4.51), the above equations can be simplified as

$$K_m(n) = \lambda K_m(n-1) + \frac{e_m^f(n) \left[ e_m^b(n-1) \right]^H}{\alpha_m(n-1)} \quad \dots (4.62)$$

$$\underline{e}_m^f(n) = \underline{e}_m^f(n) \quad \dots (4.63)$$

$$\underline{e}_{m+1}^b(n) = \underline{e}_m^b(n-1) - K_m(n) \left[ r_m^f(n) \right]^{-1} \underline{e}_m^f(n) \quad \dots (4.64)$$

$$\alpha_{m+1}(n) = \alpha_m(n-1) - \left[ \underline{e}_m^f(n) \right]^H \left[ r_m^f(n) \right]^{-1} \underline{e}_m^f(n) \quad \dots (4.65)$$

Eqns. (4.57), (4.62), (4.64) and (4.65) clearly correspond to a subset of MLSL recursions.

Similarly eqn. (4.54) can be shown to be equivalent to another subset of lattice recursions given by eqns. (4.45), (4.27a), (4.37), (4.41) and (4.43).

Alternatively,  $\tilde{\alpha}_m(n-1)$  and  $\tilde{\alpha}_m(n)$  can also be computed in the following way.

$$\tilde{\alpha}_{m+1}(n) = \tilde{\alpha}_m(n-1) \prod_{i=1}^P \cos \theta_{m,1}^f(n) \quad \dots (4.66)$$

$$\tilde{\alpha}_{m+1}^{(n-1)} = \tilde{\alpha}_m^{(n-1)} \prod_{i=0}^P \cos \theta_{m,i}^b(n) \quad \dots(4.67)$$

where  $\theta_{m,i}^f(n)$  and  $\theta_{m,i}^b(n)$ , ( $i = 0,1,2,\dots,P$ ) are the angles of Givens rotations corresponding to  $\hat{Q}_m^f(n)$  and  $\hat{Q}_m^b(n)$ , since the last diagonal elements of  $\hat{Q}_m^f(n)$  and  $\hat{Q}_m^b(n)$  are the products of cosine parameters.

Table 4.3 summarizes the QR-MLSL algorithm. After initialization, two orthogonal transformations per time sample and per filter section are computed. Only one of the possibilities in eqn.(4.66) and (4.67) is used to update  $\tilde{\alpha}_m(n)$ . We then compute the posteriori prediction and joint process errors using the following equations.

$$e_{-m+1}^f(n) = \tilde{e}_{-m+1}^f(n) \tilde{\alpha}_{m+1}^{(n-1)} \quad \dots(4.68)$$

$$e_{-m+1}^b(n) = \tilde{e}_{-m+1}^b(n) \tilde{\alpha}_{m+1}^{(n)} \quad \dots(4.69)$$

$$e_{m+1}^{(n-1)} = \tilde{e}_{m+1}^{(n-1)} \tilde{\alpha}_{m+1}^{(n-1)} \quad \dots(4.70)$$

The joint process error  $e_{m+1}^{(n-1)}$  for  $m = M$  is the error residue of the adaptive beamformer. As in the case of MLSL beamformer, here also, the weights are not computed explicitly as it is computationally expensive.

It may be noted that the QR-MLSL algorithm (Table 5.3) has a computational complexity of  $(9P^2 + 6.5P) M$  arithmetic operations per time sample. In addition, it needs  $2PM$  square-root operations per time sample.

**Table-4.3**  
**The QR-MLSL Algorithm**

Initialization

$$R_m^f(0) = R_m^b(-1) = \delta I_P \quad \delta \text{ a small positive constant}$$

$$\Gamma_m^f(0) = \Gamma_m^b(0) = 0, \quad \Gamma_m^e(0) = (0)$$

Algorithm	Complexity
-----------	------------

For  $n = 1, 2, \dots, M$  do

$$\tilde{e}_0^f(n) = \tilde{e}_0^b(n) = \underline{X}(n), \quad \tilde{e}_0^e(n) = d(n), \quad \tilde{\alpha}(n) = 1$$

For  $m = 0, 1, 2, \dots, M$  do

$$\hat{Q}_m^f(n) \begin{bmatrix} \sqrt{\lambda} R_m^f(n-1) & \sqrt{\lambda} \Gamma_m^f(n-1) \\ \left[ \tilde{e}_{-m}^f(n) \right]^H & \left[ \tilde{e}_{-m}^b(n-1) \right]^H \end{bmatrix} = \begin{bmatrix} R_m^f(n) & \Gamma_m^f(n) \\ \underline{0}^T & \left[ \tilde{e}_{-m+1}^b(n) \right]^H \end{bmatrix} \quad \begin{matrix} (4.5P^2 + 2.5P)M \\ + \text{PM sq.roots} \end{matrix}$$

$$\hat{Q}_m^b(n) \begin{bmatrix} \sqrt{\lambda} R_m^b(n-2) & \sqrt{\lambda} \Gamma_m^b(n-1) & \sqrt{\lambda} \Gamma_m^e(n-2) \\ \left[ \tilde{e}_{-m}^b(n-1) \right]^H & \left[ \tilde{e}_{-m}^f(n) \right]^H & \tilde{e}_m^*(n-1) \end{bmatrix} \\ = \begin{bmatrix} R_m^b(n-1) & \Gamma_m^b(n) & \Gamma_m^e(n-1) \\ \underline{0}^T & \left[ \tilde{e}_{-m+1}^f(n) \right]^H & \tilde{e}_{m+1}^*(n-1) \end{bmatrix} \quad \begin{matrix} (4.5P^2 + 2.5P)M \\ + \text{PM sq.roots} \end{matrix}$$

$$\tilde{\alpha}_{m+1}(n-1) = \tilde{\alpha}_m(n-1) \prod_{i=1}^P \cos \theta_{m,i}^b(n) \quad \text{PM}$$

Total	(2PM square roots) + (9P <sup>2</sup> M+6.5PM)
-------	--



### 4.3 THE MULTICHANNEL FAST TRANSVERSAL FILTER [MFTF] ALGORITHM

The MLSL and the QR-MLSL algorithms discussed in the previous sections are recursive both in order and time and hence, are computationally expensive. Since the number of taps behind each sensor is equal to  $M$ , the order of the adaptive beamformer is fixed. Therefore, here we, derive the multichannel fast transversal filter (MFTF) algorithm, which is fixed in order and solves the eqn.(4.1) recursively in time. We use a priori error forms of forward prediction errors, backward prediction errors and joint process estimation errors as the variables of interest.

In addition to the a priori error residual of the broadband beamformer given in eqn.(4.8), the a priori forward and backward error residual vectors are defined as

$$\underline{\eta}_M^f(n) = \underline{X}(n) - A_{PM}^H(n-1)\underline{X}_{PM}(n-1) \quad \dots(4.71)$$

$$\underline{\eta}_M^b(n) = \underline{X}(n-M) - B_{PM}^H(n-1)\underline{X}_{PM}(n) \quad \dots(4.72)$$

It can be shown that the error variance matrices  $r_M^f(n+1)$  and  $r_M^b(n+1)$  of equations (4.28a) and (4.28b) for  $m = M$ , can be recursively updated as

$$r_M^f(n+1) = \lambda r_M^f(n) + e_M^f(n+1) \left[ \underline{\eta}_M^f(n+1) \right]^H \quad \dots(4.73)$$

$$r_M^b(n+1) = \lambda r_M^b(n) + \underline{\eta}_M^b(n+1) \left[ e_M^b(n+1) \right]^H \quad \dots(4.74)$$

It may also be easily shown, as in [43], that the coefficient matrices of forward and backward filters,  $A_{PM}(n)$  and  $B_{PM}(n)$ , are recursively updated in time using the following equations.

$$A_{PM}(n+1) = A_{PM}(n) + C_{PM}(n) \left[ e_M^f(n+1) \right]^H \quad \dots (4.75)$$

$$B_{PM}(n+1) = B_{PM}(n) + C_{PM}(n+1) \left[ e_M^b(n+1) \right]^H \quad \dots (4.76)$$

Similarly, the weight vector  $\underline{W}_{PM}(n)$  of the adaptive beamformer is recursively updated using

$$\underline{W}_{PM}(n+1) = \underline{W}_{PM}(n) + C_{PM}(n+1) e_M^*(n+1) \quad \dots (4.77)$$

The complex gain vector  $C_{PM}(n+1)$  in the above equation is defined as

$$C_{PM}(n+1) = \frac{1}{\lambda} \Phi_{PM}^{-1}(n) \underline{X}_{PM}(n+1) \quad \dots (4.78)$$

By analogy with the above equation, we define an  $(M+1)P$ -by-1 extended gain vector

$$\underline{C}_{P(M+1)}(n+1) = \frac{1}{\lambda} \Phi_{P(M+1)}^{-1}(n) \underline{X}_{P(M+1)}(n+1) \quad \dots (4.79)$$

where  $\Phi_{P(M+1)}(n)$  is the correlation matrix of the extended vector  $\underline{X}_{P(M+1)}(n+1)$  and is given by

$$\Phi_{P(M+1)}(n+1) = \sum_{j=1}^n \lambda^{n-j} \left[ \underline{X}_{P(M+1)}(n+1) \right] \left[ \underline{X}_{P(M+1)}(n+1) \right]^H \quad \dots (4.80)$$

The vector  $\underline{x}_{P(M+1)}^{(n+1)}$  in eqns. (4.79) and (4.80) is defined as

$$\underline{x}_{P(M+1)}^{(n+1)} = \begin{bmatrix} x_1(n+1), x_2(n+1), \dots, x_p(n+1), \\ x_1(n), x_2(n), \dots, x_p(n), \\ \dots \\ x_1(n-M), x_2(n-M), \dots, x_p(n-M) \end{bmatrix}^T \quad \dots (4.81)$$

In Appendix-E, it is shown that the extended gain vector  $\underline{c}_{P(M+1)}^{(n+1)}$  can be recursively updated as

$$\underline{c}_{P(M+1)}^{(n+1)} = \begin{bmatrix} \frac{1}{\lambda} \left[ r_M^f(n) \right]^{-1} & \underline{\eta}_M^f(n+1) \\ \underline{c}_{PM}^{(n)} - \frac{1}{\lambda} A_{PM}^{(n)} \left[ r_M^f(n) \right]^{-1} & \underline{\eta}_M^f(n+1) \end{bmatrix} \quad \dots (4.82a)$$

or

$$\underline{c}_{P(M+1)}^{(n+1)} = \begin{bmatrix} \underline{c}_{PM}^{(n+1)} - \frac{1}{\lambda} B_{PM}^{(n)} \left[ r_M^b(n) \right]^{-1} & \underline{\eta}_M^b(n+1) \\ \frac{1}{\lambda} \left[ r_M^b(n) \right]^{-1} & \underline{\eta}_M^b(n+1) \end{bmatrix} \quad \dots (4.82b)$$

Defining

$$\underline{c}_{P(M+1)}^{(n+1)} = \begin{bmatrix} \bar{\underline{c}}_{PM}^{(n+1)} \\ \underline{\mu}_P^{(n+1)} \end{bmatrix} \quad \dots (4.83)$$

and using eqns. (4.82a) and (4.82b), we get

$$\underline{c}_{PM}^{(n+1)} = \bar{\underline{c}}_{PM}^{(n+1)} + B_{PM}^{(n)} \underline{\mu}_P^{(n+1)} \quad \dots (4.84)$$

and

$$\underline{\eta}_M^b(n+1) = \lambda \left[ r_M^b(n) \right] \underline{\mu}_P^{(n+1)} \quad \dots (4.85)$$

As in the case of MLSL algorithm, we define here a scalar constant  $\alpha_M(n)$  given by

$$\alpha_M(n) = 1 - \underline{X}_{PM}^H(n) \underline{C}_{PM}(n) \quad \dots(4.86)$$

which is recursively updated as

$$\alpha_M(n+1) = \alpha_M(n) + \frac{1}{\lambda} \left[ \underline{\eta}_M^b(n+1) \right]^H \left[ \underline{r}_M^b(n) \right]^{-1} \underline{\eta}_M^b(n+1) \quad \dots(4.87)$$

The a posteriori error residual vectors can be updated using a priori error residual vectors and scalar constant  $\alpha_M(n)$  as

$$\underline{e}_M^f(n+1) = \underline{\eta}_M^f(n+1) / \alpha_M(n) \quad \dots(4.88)$$

$$\underline{e}_M^b(n+1) = \underline{\eta}_M^b(n+1) / \alpha_M(n+1). \quad \dots(4.89)$$

The a posteriori error residual of the least-square filter is updated as

$$\underline{e}_M(n+1) = \underline{\eta}_M(n+1) / \alpha_M(n+1). \quad \dots(4.90)$$

We can also write the MFTF algorithm using scalar constant  $\gamma_M(n)$  which is defined as

$$\gamma_M(n) = 1 / \alpha_M(n). \quad \dots(4.91)$$

The extended scalar constant, using the extended gain vector, can be written as

$$\alpha_{M+1}(n+1) = 1 - \left[ \underline{C}_{P(M+1)}(n+1) \right]^H \underline{X}_{P(M+1)}(n+1) \quad \dots(4.92)$$

Using eqn. (4.82a) and (4.82b),  $\alpha_{M+1}(n+1)$  can be expressed in two different forms as

$$\alpha_{M+1}(n+1) = \alpha_M(n) + \frac{1}{\lambda} \left[ \eta_M^f(n+1) \right]^H \left[ r_M^f(n) \right]^{-1} \eta_M^f(n+1) \quad \dots (4.93)$$

$$\alpha_{M+1}(n+1) = \alpha_M(n+1) + \frac{1}{\lambda} \left[ \eta_M^b(n+1) \right]^H \left[ r_M^b(n) \right]^{-1} \eta_M^b(n+1) \quad \dots (4.94)$$

Eqns. (4.93) and (4.94) can now be rewritten in terms of  $\gamma_{M+1}(n+1)$  as

$$\gamma_{M+1}(n+1) = \frac{\gamma_M(n)}{1 + \frac{1}{\lambda} \left[ \eta_M^f(n+1) \right]^H \left[ r_M^f(n) \right]^{-1} \eta_M^f(n+1)} \quad \dots (4.95)$$

and

$$\gamma_M(n+1) = \frac{\gamma_{M+1}(n+1)}{1 + \left[ \eta_M^b(n+1) \right]^H \mu_P(n+1) \gamma_{M+1}(n+1)} \quad \dots (4.96)$$

This completes the derivation of the MFTF algorithm. A summary of the MFTF algorithm is presented in Table 4.4. It is found that it has a computational complexity of about  $(P^3M + 4P^2M + 3P^2 + 2PM + 4P + 1)$ , with a forgetting factor  $\lambda$  equal to unity. In addition, it requires about 2 divisions per time sample. Further, inverse of the error variance matrix  $r_M^f(n)$  has to be computed once per time sample.

**Table-4.4**  
**The Multichannel Fast Transversal Filter [MFTF] Algorithm**

Initialization		
$\gamma_M(0) = \lambda$		
$r_M^f(0) = r_M^b(0) = \delta I, \delta \text{ a small +ve constant}$		(T.4.4.1)
$\underline{W}_{PM}(0) = \underline{C}_{PM}(0) = \underline{0}_{PM}, \quad A_{PM}(0) = B_{PM}(0) = \underline{0}_{PM \times P}$		(T.4.4.2)
Algorithm	Complexity	
$\eta_M^f(n+1) = \underline{X}(n+1) - A_{PM}^H(n) \underline{X}_{PM}(n)$	$P^2M$	(T.4.4.3)
$\underline{e}_M^f(n+1) = \eta_M^f(n+1) \gamma_M(n)$	$P$	(T.4.4.4)
$\underline{C}_{P(M+1)}(n+1) = \begin{bmatrix} \frac{1}{\lambda} [r_M^f(n)]^{-1} & \eta_M^f(n+1) \\ \underline{C}_{PM}(n) - \frac{1}{\lambda} A_{PM}(n) [r_M^f(n)]^{-1} & \eta_M^f(n+1) \end{bmatrix}$	$P^3M$	(T.4.4.5)
$A_{PM}(n+1) = A_{PM}(n) + \underline{C}_{PM}(n) [\underline{e}_M^f(n+1)]^H$	$P^2M$	(T.4.4.6)
$r_M^f(n+1) = \lambda r_M^f(n) + \underline{e}_M^f(n+1) [\eta_M^f(n+1)]^H$	$P^2$	(T.4.4.7)
$\underline{C}_{P(M+1)}(n+1) = \begin{bmatrix} \bar{\underline{C}}_{PM}(n+1) \\ \underline{\mu}_P(n+1) \end{bmatrix}$		(T.4.4.8)
$\underline{C}_{PM}(n+1) = \bar{\underline{C}}_{PM}(n+1) + B_{PM}(n) \underline{\mu}_P(n+1)$	$P^2M$	(T.4.4.9)
$\eta_M^b(n+1) = \lambda [r_M^b(n)] \underline{\mu}_P(n+1)$	$P^2$	(T.4.4.10)
$\gamma_{M+1}(n+1) = \frac{\gamma_M(n)}{1 + \frac{1}{\lambda} [\eta_M^f(n+1)]^H [r_M^f(n)]^{-1} \underline{e}_M^f(n+1)}$	$P^2 + P$	(T.4.4.11)

$\gamma_M(n+1) = \frac{\gamma_{M+1}(n)}{1 + \left[ \underline{\eta}_M^b(n+1) \right]^H \underline{\mu}_P(n+1) \gamma_{M+1}(n+1)}$	2P	(T. 4. 4. 12)
$\underline{e}_M^b(n+1) = \underline{\eta}_M^b(n+1) \gamma_M(n+1)$	P	(T. 4. 4. 13)
$r_M^b(n+1) = \lambda r_M^b(n) + \underline{\eta}_M^b(n+1) \left[ \underline{e}_M^b(n+1) \right]^H$	P <sup>2</sup>	(T. 4. 4. 14)
$B_{PM}(n+1) = B_{PM}(n) + \underline{C}_{PM}(n) \left[ \underline{e}_M^b(n+1) \right]^H$	P <sup>2</sup> M	(T. 4. 4. 15)
$\eta_M(n+1) = d(n+1) - \underline{W}_{PM}^H(n) \underline{X}_{PM}(n+1)$	PM	(T. 4. 4. 16)
$e_M(n+1) = \eta_M(n+1) \gamma_M(n+1)$	1	(T. 4. 4. 17)
$\underline{W}_{PM}(n+1) = \underline{W}_{PM}(n) + \underline{C}_{PM}(n+1) e_M^*(n+1)$	PM	(T. 4. 4. 18)
Total	(divisions 2) + (P <sup>3</sup> M + 4P <sup>2</sup> M + 3P <sup>2</sup> + 2PM + 4P + 1)	

#### 4.4 SIMULATION RESULTS

Extensive computer simulations were carried out to study fast RLS algorithms, viz, MLSL, MFTF and the QR-MLSL for adaptive beamforming in a broadband signal environment. To facilitate a comparison with exact least-square algorithms discussed in chapter 3, signal environments corresponding to four examples presented in sec.3.6 (Examples 3.6-2.1 to 3.6-2.4) have been considered. The performance evaluation has been based upon the convergence characteristics and the ability of these beamformers to reproduce the desired signal. The voltage patterns have been presented only for MFTF

beamformer, which appears to be the best among the three beamformers considered in this chapter.

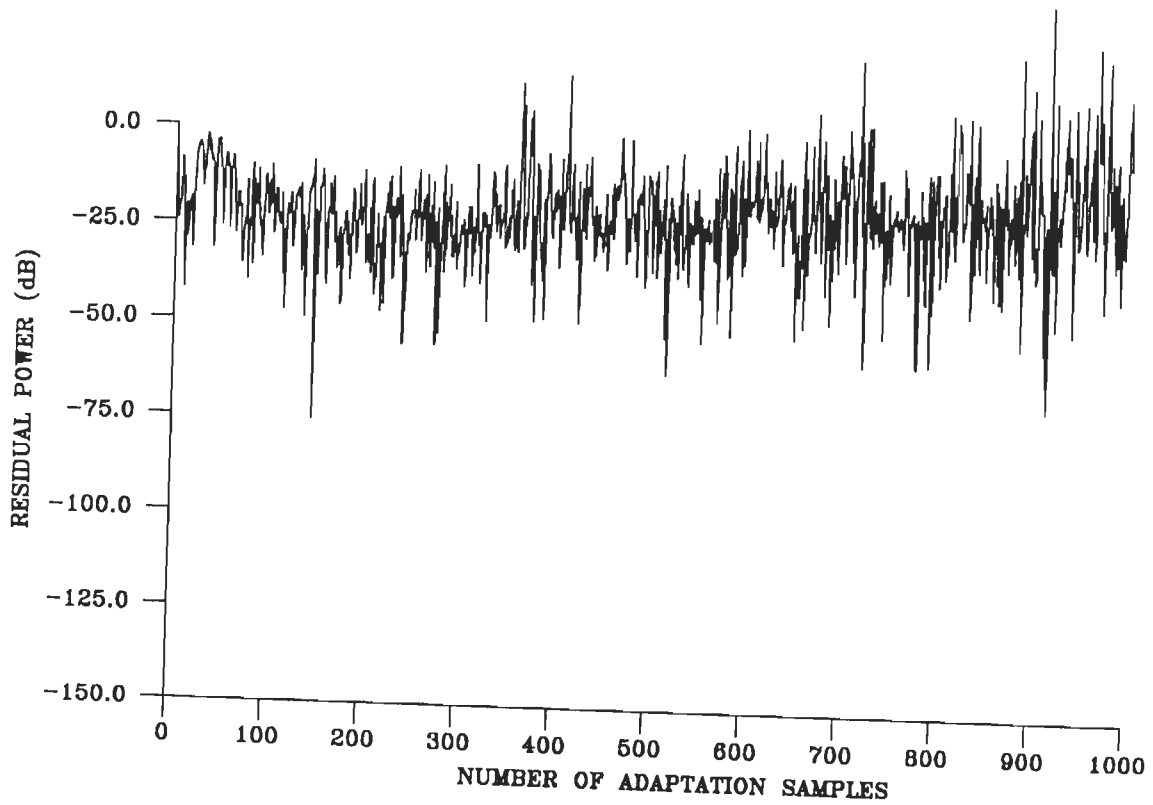
#### Example 4.4-1

In this example, two broadband interferences are assumed to arrive from  $+60^\circ$  and  $-60^\circ$ . The remaining parameters are as given in Table 3.11.

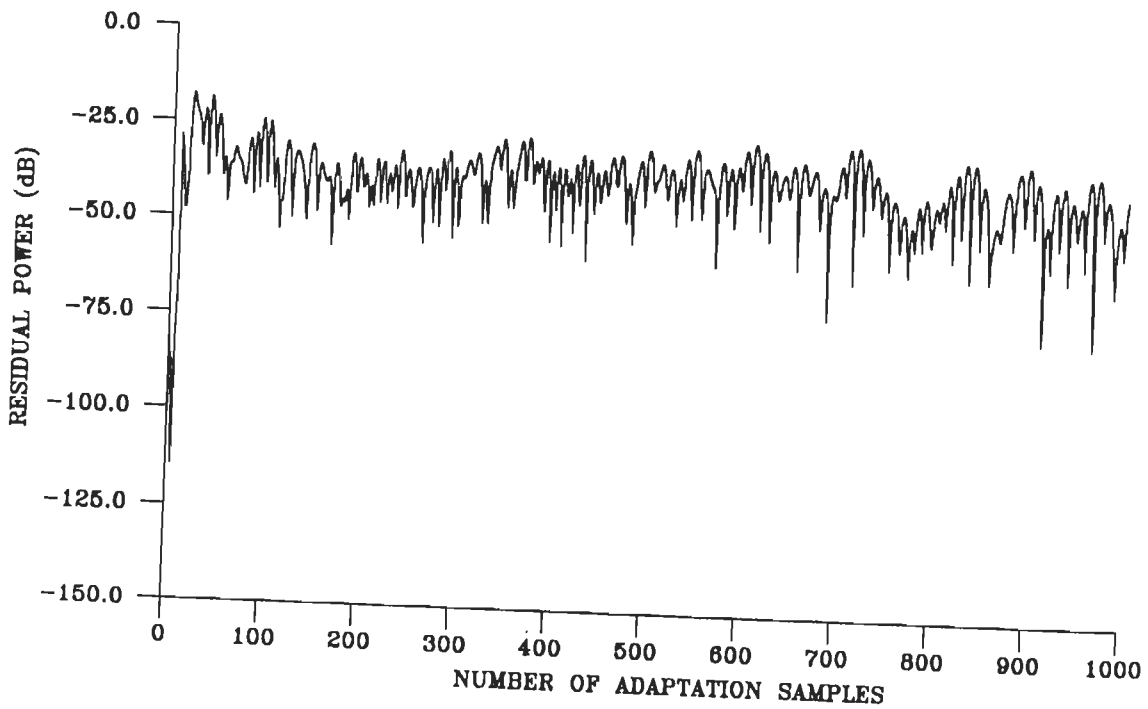
The output residual power characteristics of the three beamformers are shown in Fig.4.1. It is evident from Fig.4.1(a) that the MLSL beamformer does not converge even after 1000 samples. The beamformer exhibits an initial convergence in the first four hundred samples. However, during this interval, it exhibits a large residual power of about  $-20\text{dB}$ . After 400 samples, the characteristics exhibit continuous fluctuations, the severity of which increases with the increase in the number of samples. These fluctuations deviate from the mean by a large amplitude. On the other hand, the MFTF and the QR-MSL beamformer converge in about 50 samples and the residual powers are of the order of  $-40\text{dB}$  and  $-50\text{dB}$ , respectively. It may be recalled that, for the same problem, all the least-squares beamformers exhibit numerical stability and fast convergence (Fig.3.13), with RLS and QRD-LS beamformers producing residual powers of  $-75\text{dB}$  and  $-80\text{dB}$ , respectively.

Fig.4.2 shows the output signal waveforms from these beamformers. A comparison with the desired signal waveform (Fig.2.7) shows that the output waveform of MLSL beamformer is distorted at several places. Though the QR-MSL beamformer tracks the desired

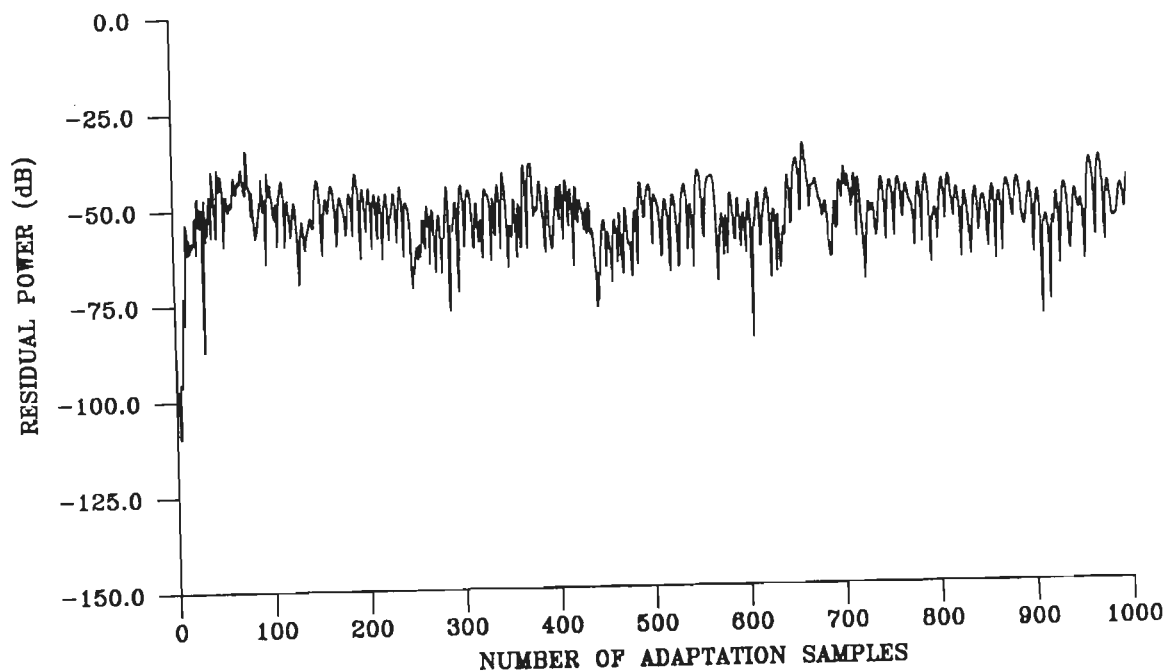




(a) MLSL BEAMFORMER

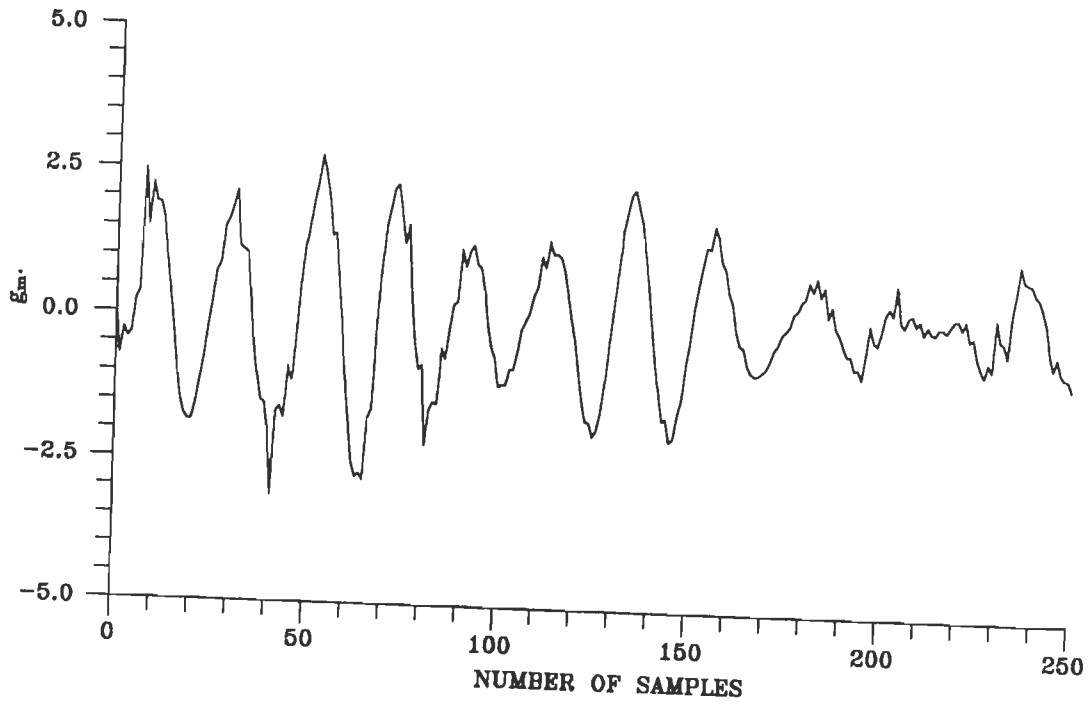


(b) MFTF BEAMFORMER

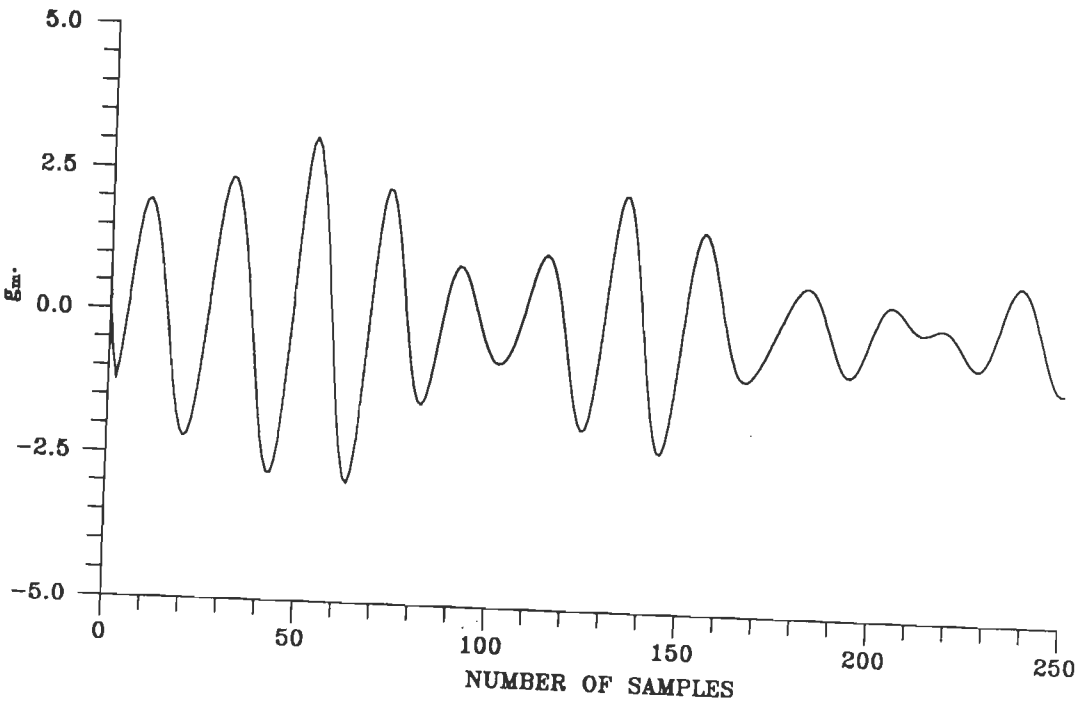


(c) QR-MLSL BEAMFORMER

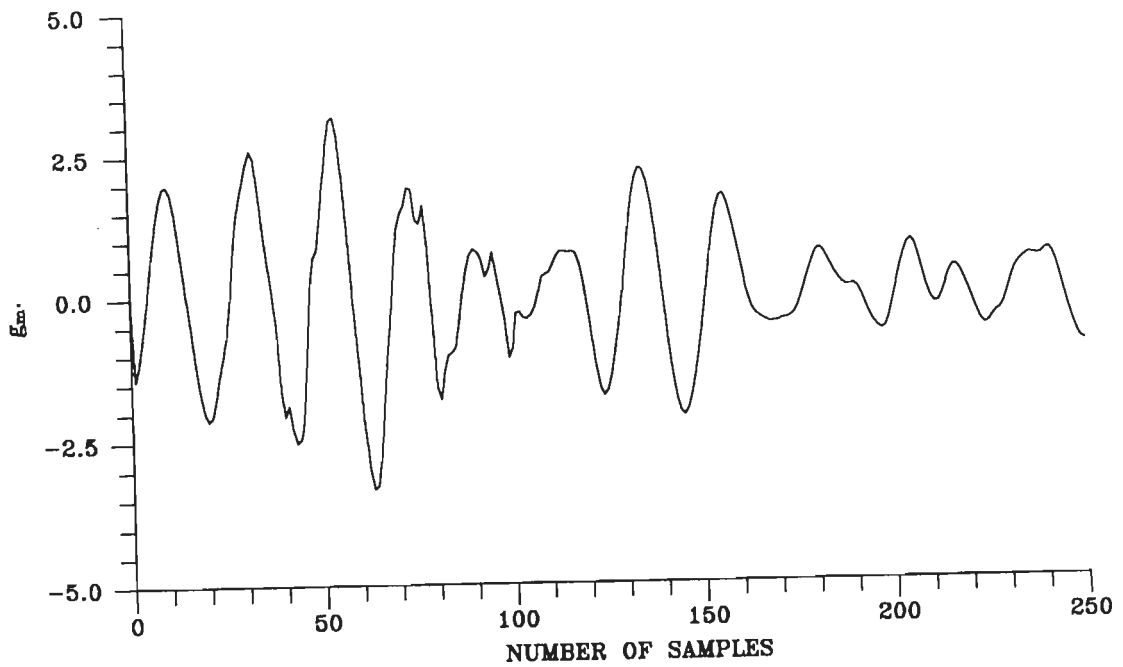
FIG.4.1 CONVERGENCE CHARACTERISTICS OF FAST RLS BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $60^\circ$  AND  $-60^\circ$ .



(a) MLSL BEAMFORMER



(b) MFTF BEAMFORMER



(c) QR-MLSL BEAMFORMER

FIG.4.2 OUTPUT SIGNAL WAVEFORMS OF THE THREE  
FAST RLS BROADBAND BEAMFORMERS WITH  
INTERFERENCES ARRIVING AT  $60^\circ$  AND  $-60^\circ$ .

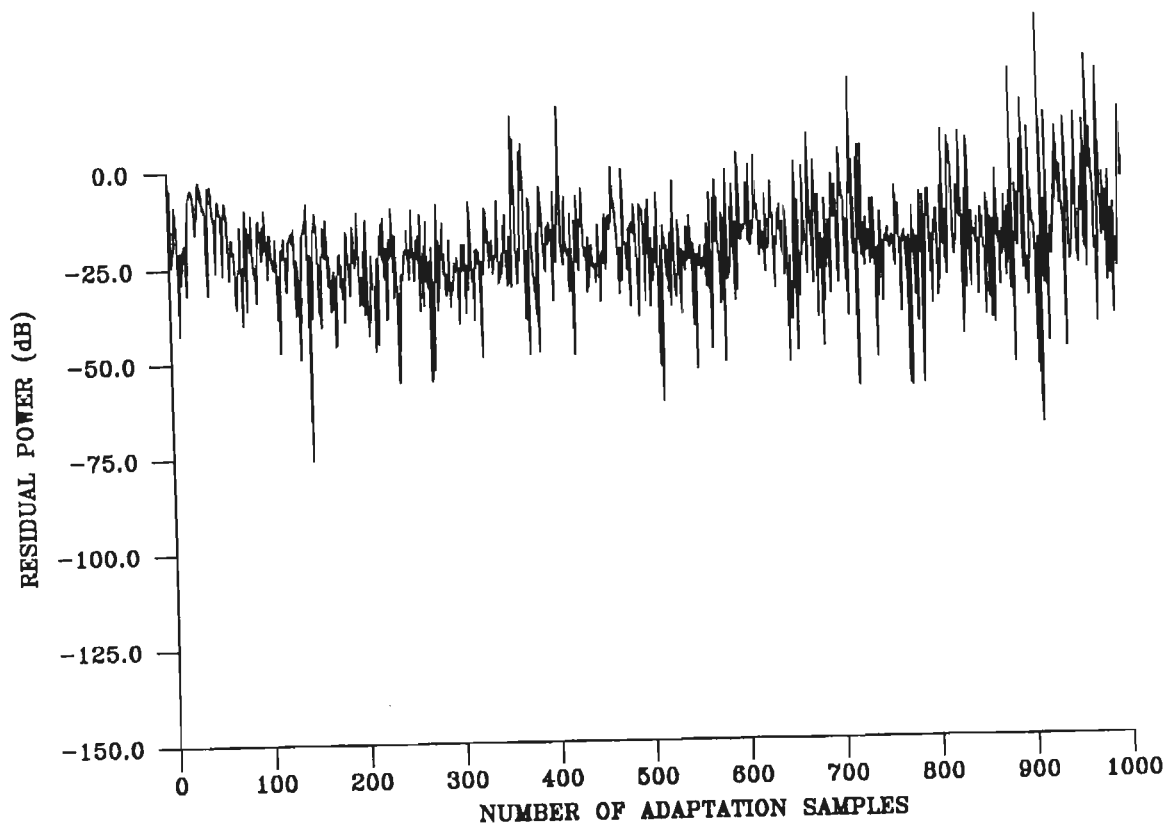
signal from the beginning, its output waveform is also distorted at many places. On the other hand, the MFTF beamformer (Fig.4.2(b)) tracks the desired signal satisfactorily just like the exact RLS beamformers (Fig.3.15).

#### Example 4.4-2

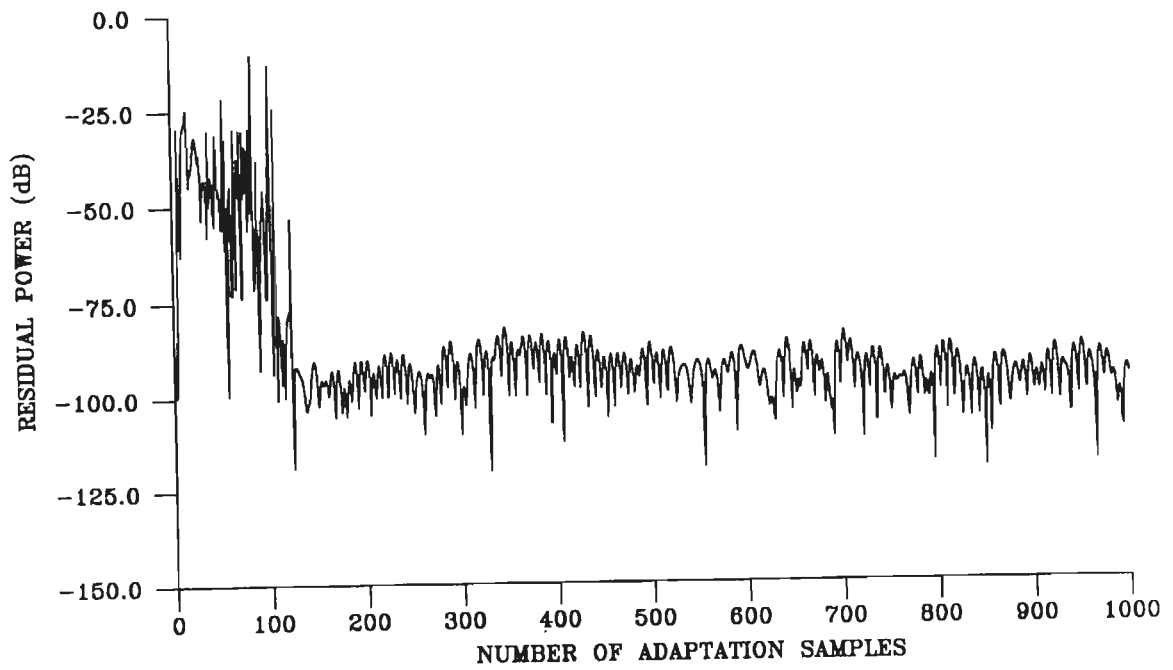
In this example, the interferences arrive very close to each other ( $50^\circ$  and  $60^\circ$ ) but far away from the desired signal.

From the residual power characteristics shown in Fig.4.3, it is clear that, as in the previous example, the MLSL beamformer fails to converge and the residual power is large (-20dB). The MFTF array converges in about 100 samples and has the least residual power (-100dB). QR-MLSL array exhibits faster convergence but has a larger residual power of the order -60dB. On comparison with the residual power characteristics of exact RLS beamformers (Fig.3.16), it is clear while the RLS beamformer exhibits faster convergence as compared to the MFTF beamformer, the RMGS beamformers (Fig.3.16(b) and (c)) and the MFTF beamformer have more or less the same convergence speed. However, the exact RLS beamformers have a lower residual power of -75dB as compared to the MFTF beamformer. The performance of QRD-LS beamformer (Fig.3.16(d)) is found to be superior to that of QR-MLSL beamformer.

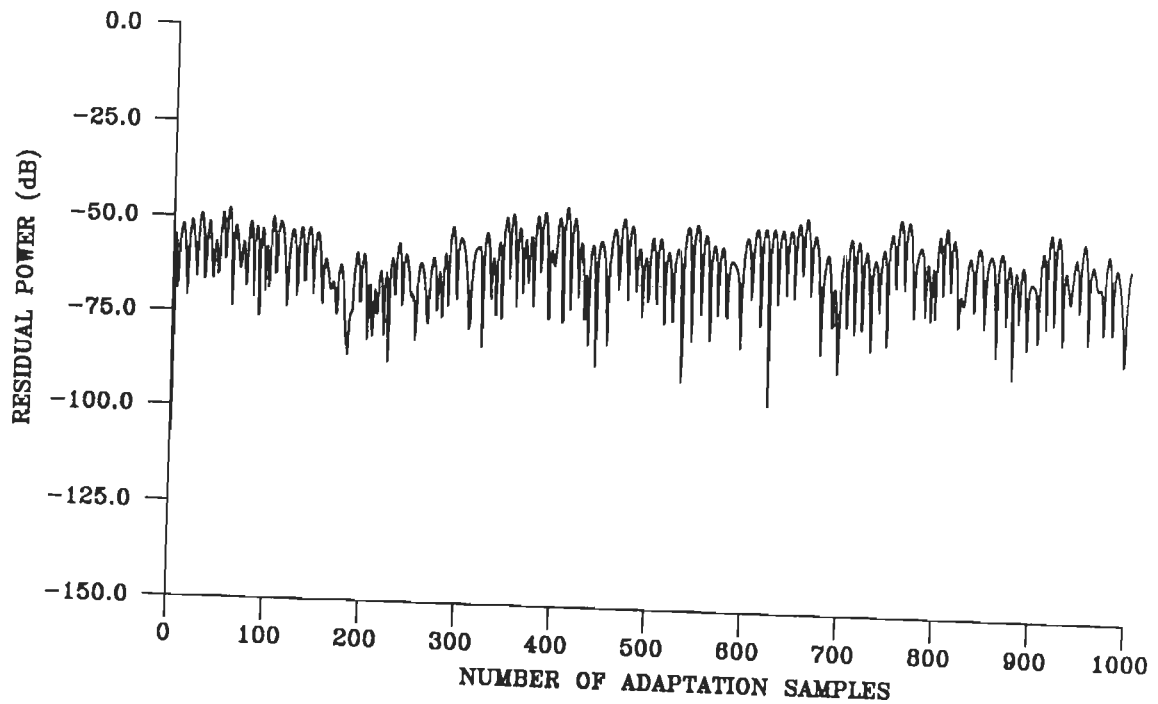
Fig.4.4 shows the output signal waveforms of these three beamformers. In this case, the output signal waveforms of all the three beamformers track the desired signal satisfactorily.



(a) MLSL BEAMFORMER

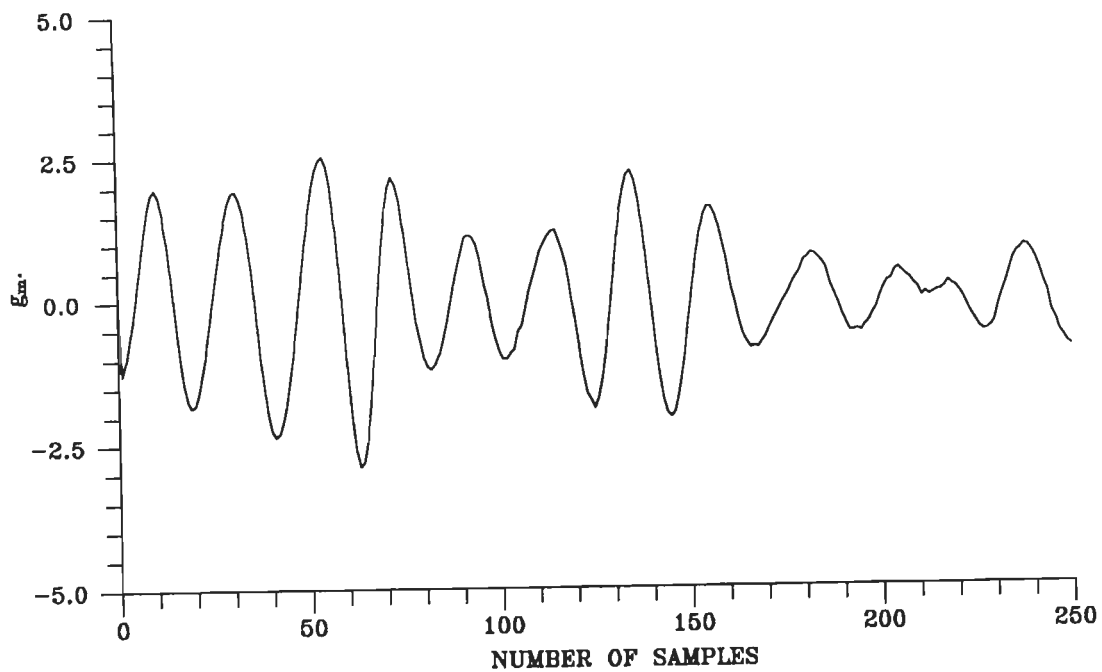


(b) MFTF BEAMFORMER

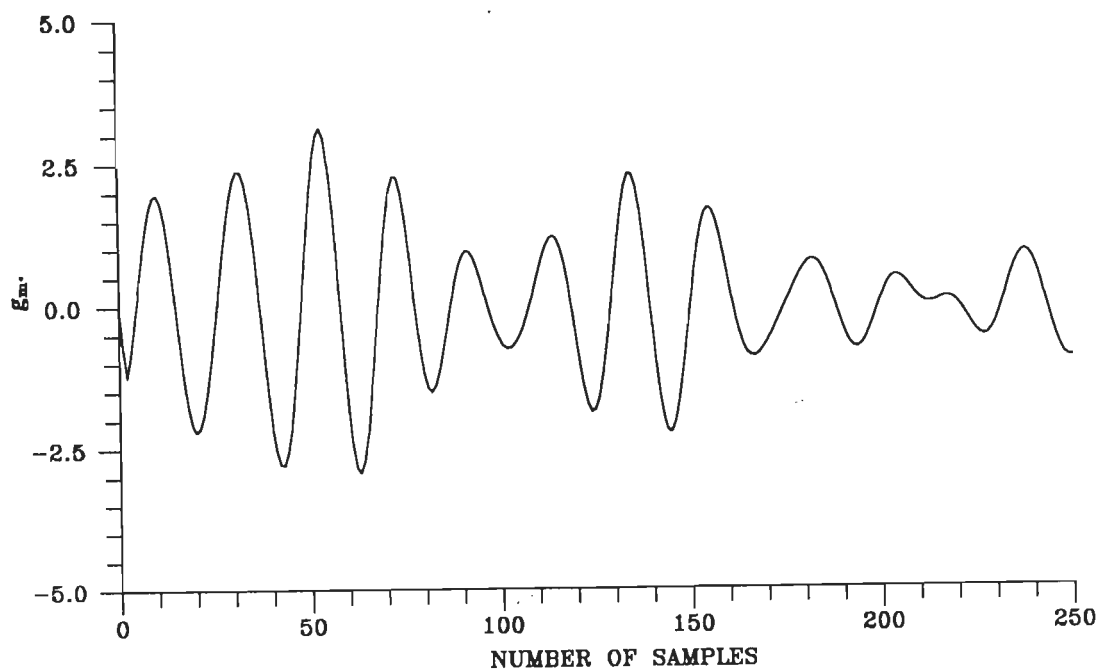


(c) QR-MLSL BEAMFORMER

FIG.4.3 CONVERGENCE CHARACTERISTICS OF FAST RLS BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT 50° AND 60°.

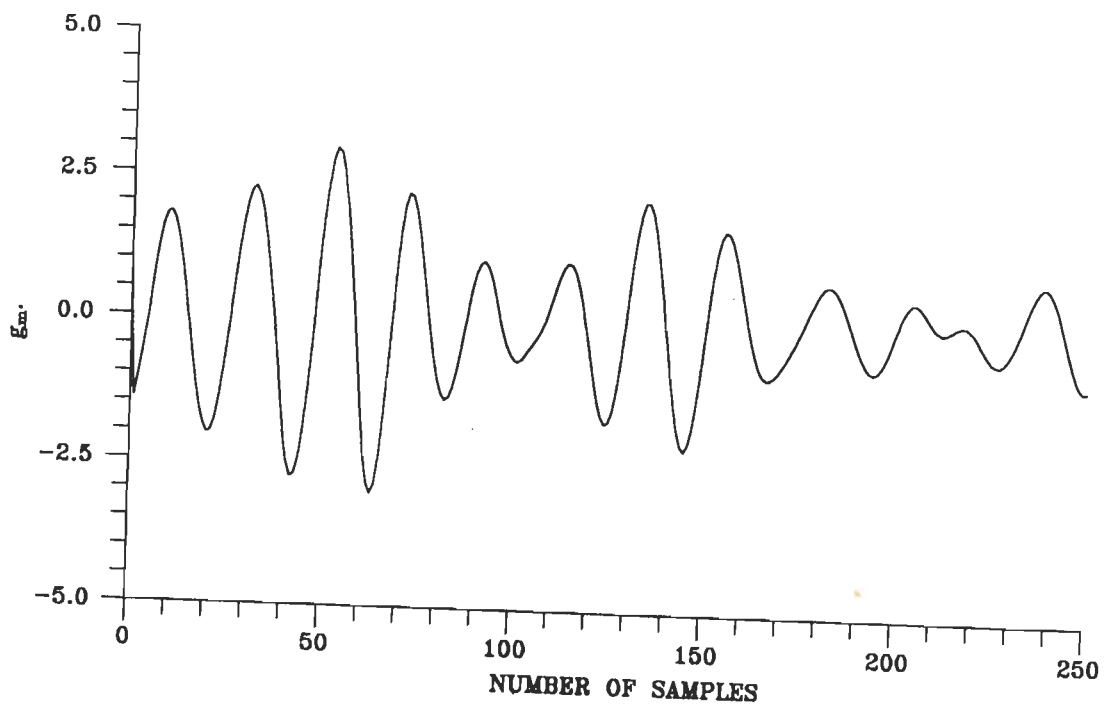


(a) MLSL BEAMFORMER



(b) MFTF BEAMFORMER





(c) QR-MLSL BEAMFORMER

FIG.4.4 OUTPUT SIGNAL WAVEFORMS OF THE THREE FAST-RLS BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT 50° AND 60°.

#### Example 4.4-3

In this example, the two interferences have been assumed to arrive very close to the desired signal, i.e., from  $+5^\circ$  and  $-5^\circ$ .

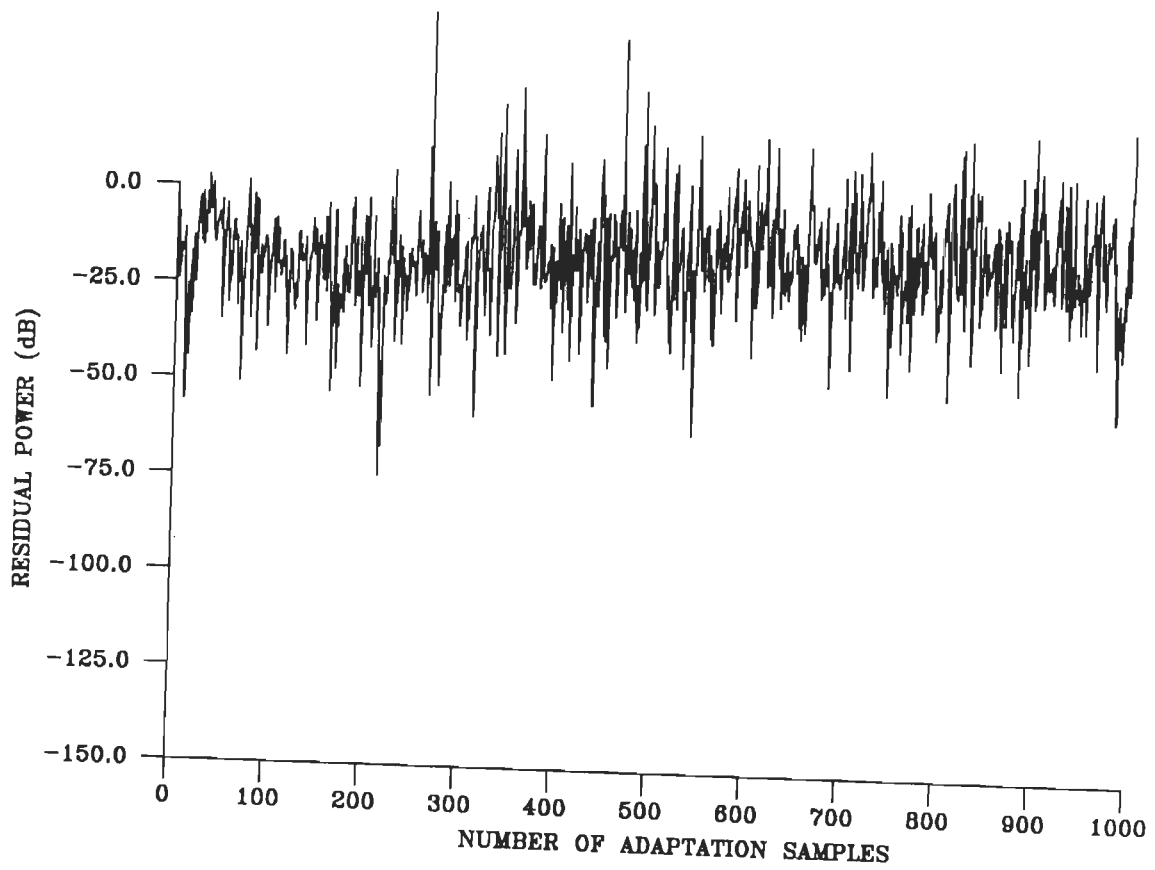
From the residual power characteristics in Fig.4.5, it can be seen that the MLSL beamformer again exhibits poor convergence with large amplitude fluctuations. On the other hand, the MFTF and the QR-MLSL beamformers converge quickly to a mean residual power of about -60dB. On comparing with Fig.3.18, it is found that the RLS beamformer (Fig.3.18(a)) takes a large number of samples (400) to converge. Also, the exact RLS beamformers exhibit a larger residual power of the order -30dB compared to that of MFTF and QR-MLSL beamformers. The performance of QR-MLSL beamformer, in the present scenario, is comparable to that of QRD-LS beamformer.

The output signal waveforms of the three beamformers are shown in Fig.4.6. It is found that the MLSL output waveform is fully distorted and does not resemble the desired signal waveform. The MFTF and QR-MLSL beamformers, on the other hand, reproduce the desired signal waveform faithfully except for some minor distortions.

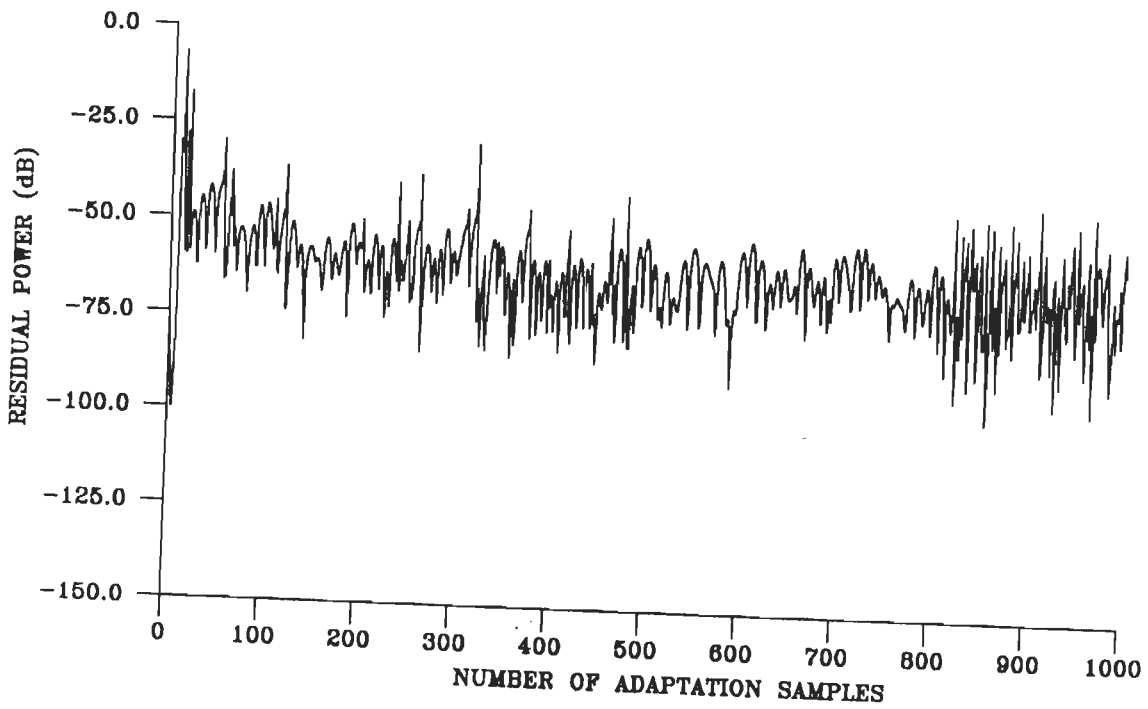
#### Example 4.4-4

In this example, the two interferences arrive from near endfire directions at  $+80^\circ$  and  $-80^\circ$ .

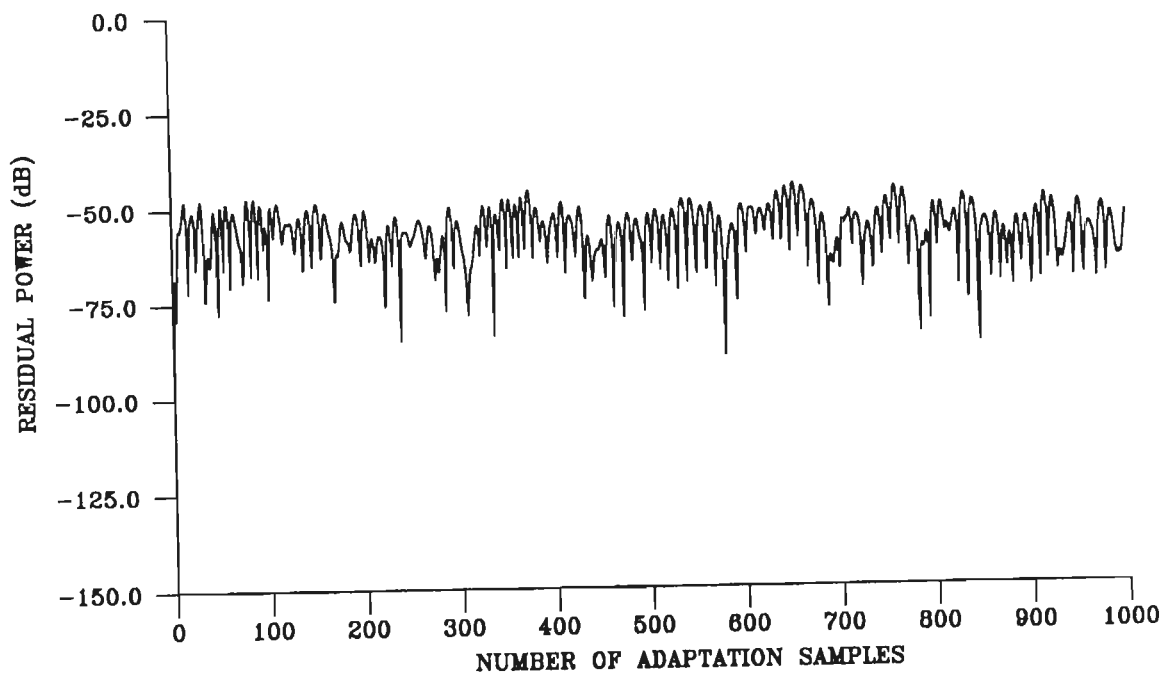
Figs. 4.7 and 4.8, respectively show the residual power and the output signal waveforms of the three beamformers. In this situation also, the MLSL beamformer exhibits divergence and produces an output waveform which is highly distorted. On the other hand, the



(a) MLSL BEAMFORMER

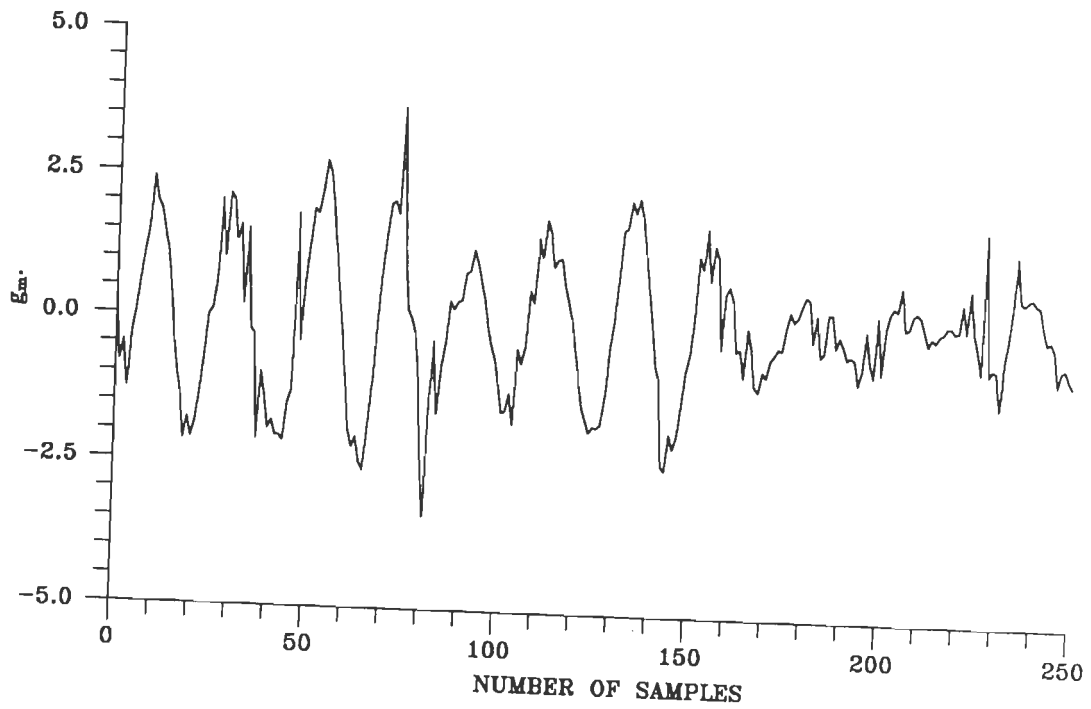


(b) MFTF BEAMFORMER

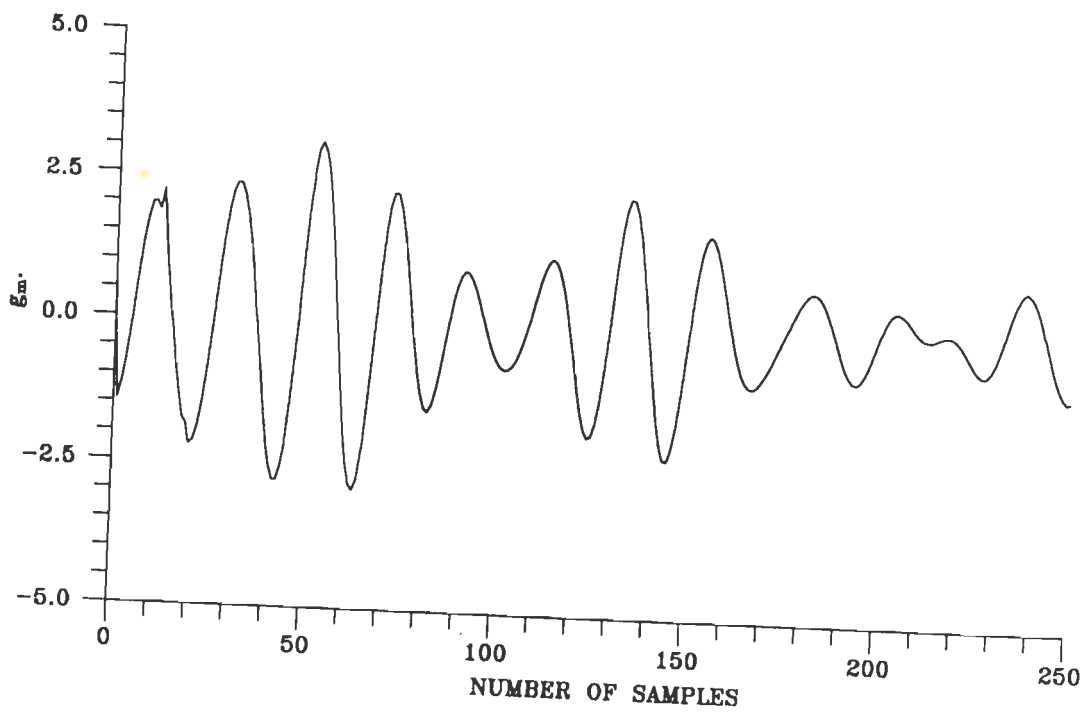


(c) QR-MSL BEAMFORMER

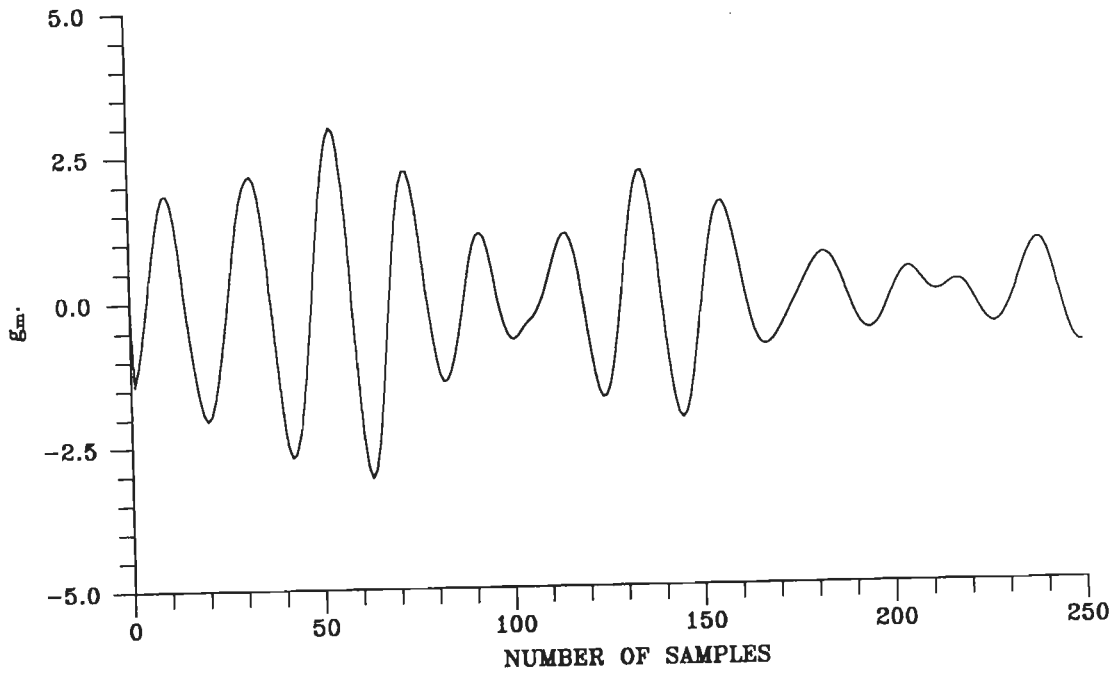
FIG.4.5 CONVERGENCE CHARACTERISTICS OF FAST RLS BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $5^\circ$  AND  $-5^\circ$ .



(a) MLSL BEAMFORMER

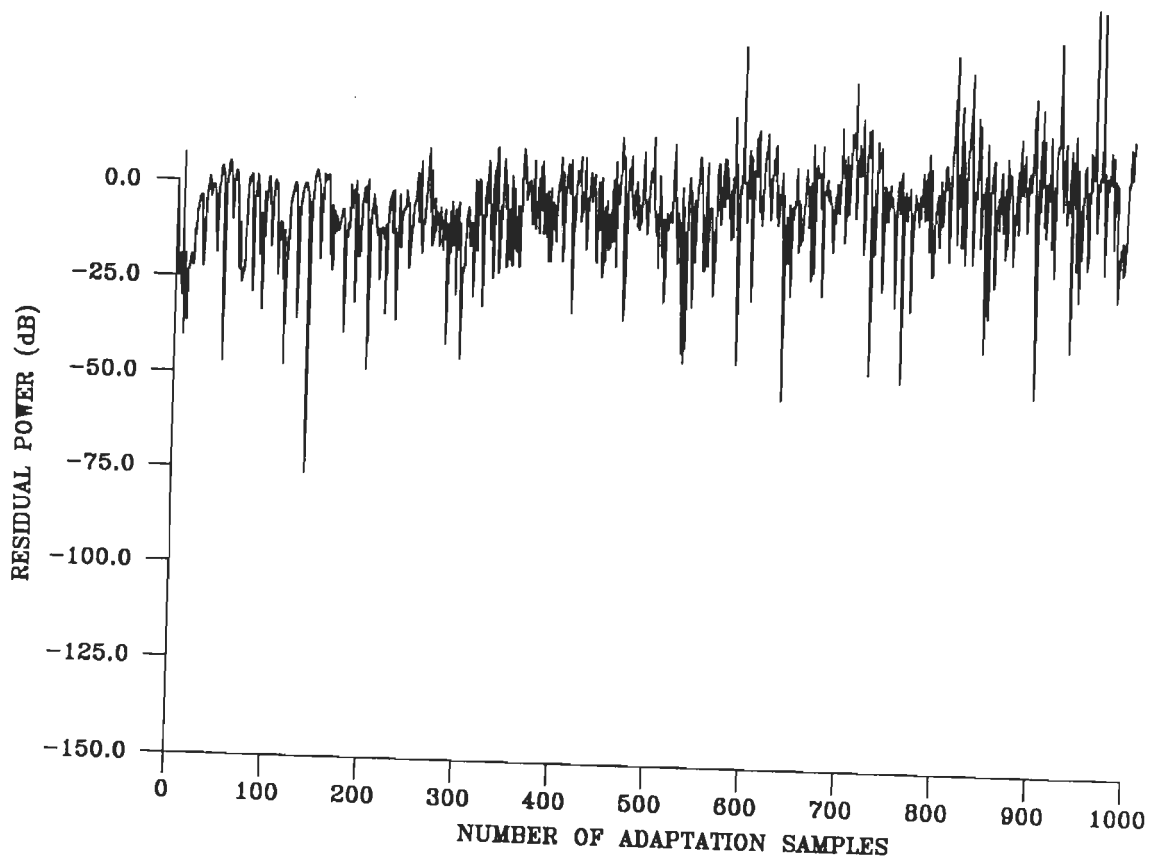


(b) MFTF BEAMFORMER

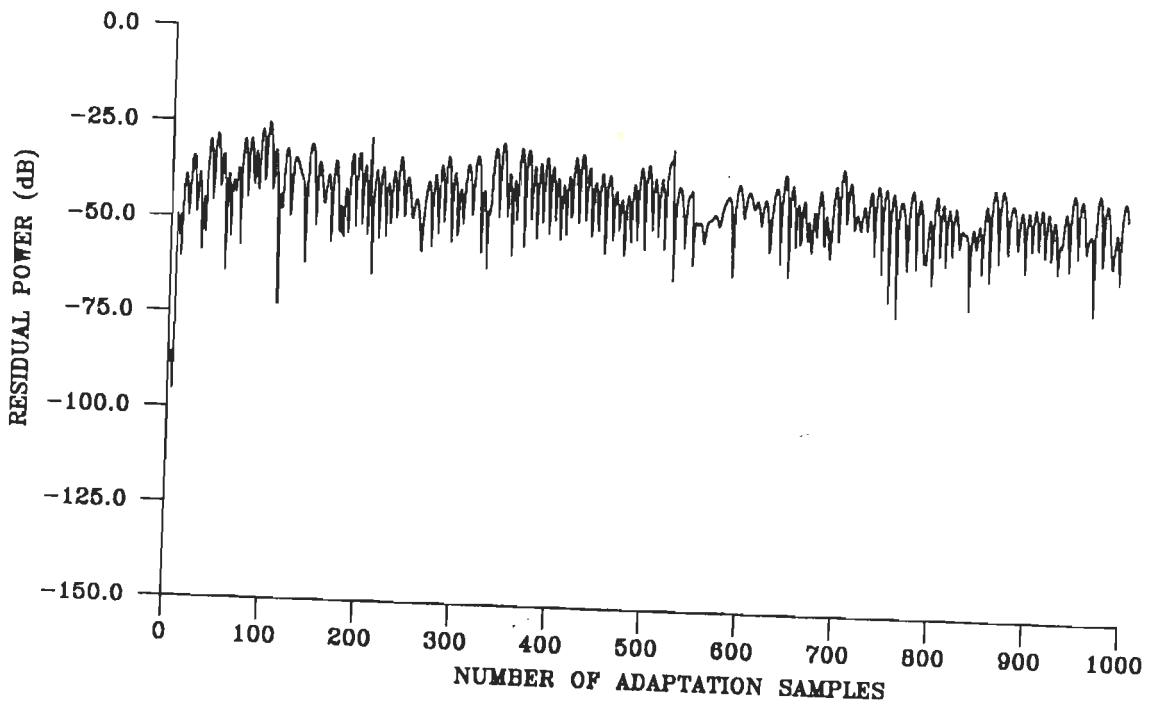


(c) QR-MLSL BEAMFORMER

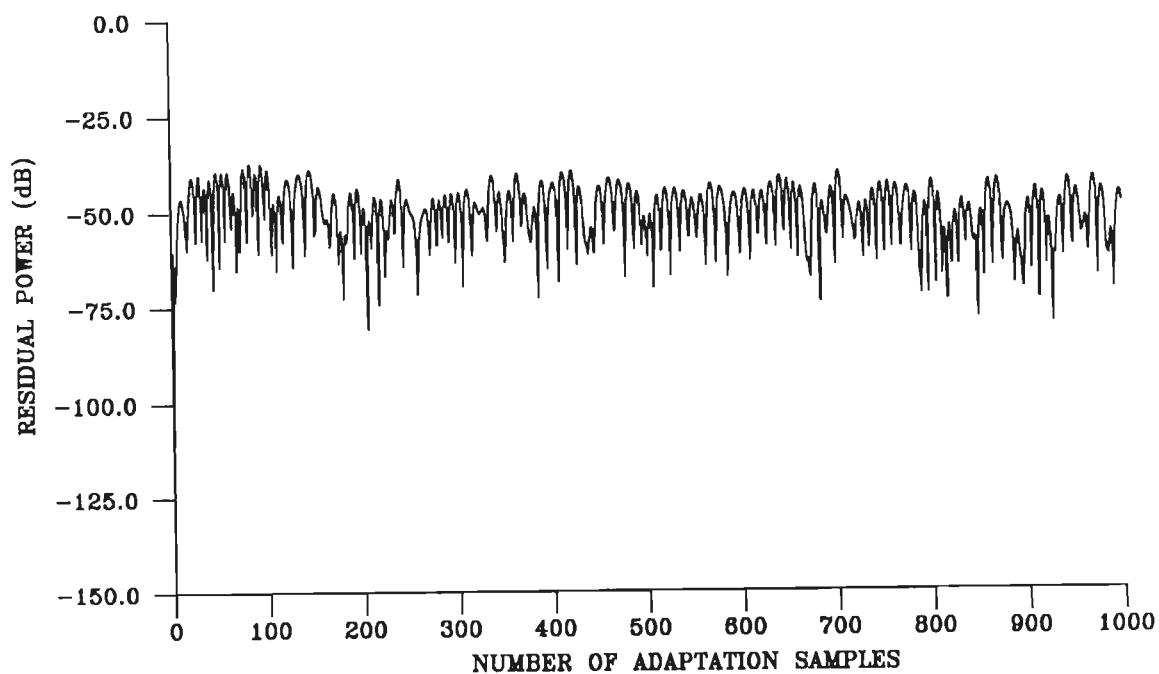
FIG.4.6 OUTPUT SIGNAL WAVEFORMS OF THE THREE  
FAST RLS BROADBAND BEAMFORMERS WITH  
INTERFERENCES ARRIVING AT  $5^\circ$  AND  $-5^\circ$ .



(a) MSL BEAMFORMER



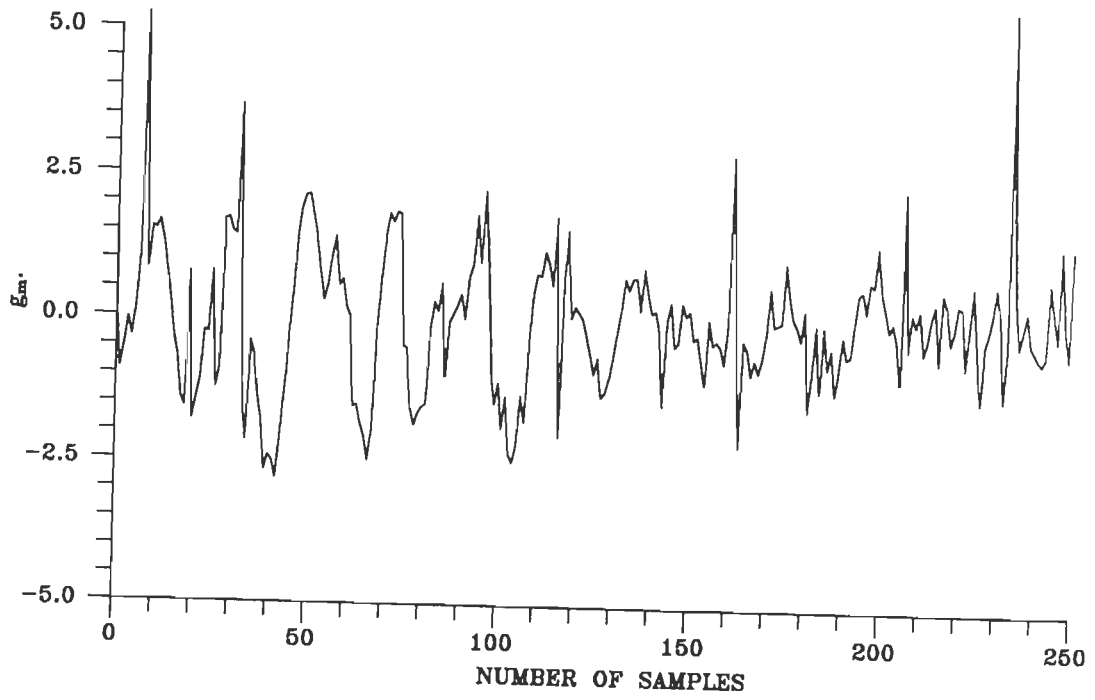
(b) MFTF BEAMFORMER



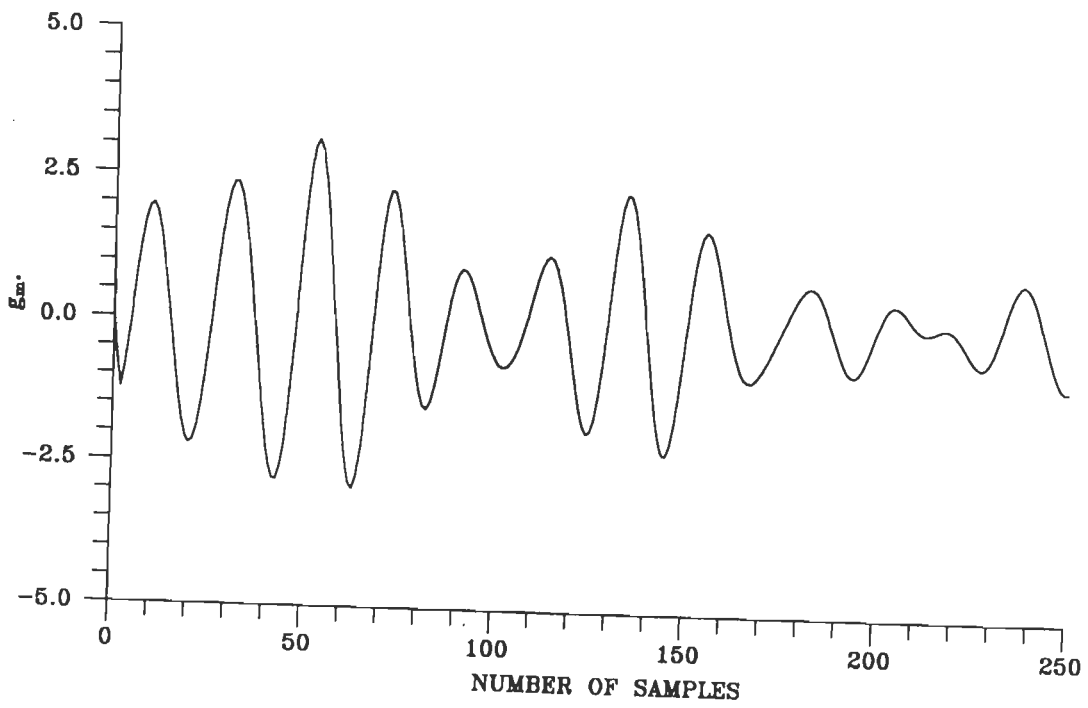
(c) QR-MLSL BEAMFORMER

FIG.4.7 CONVERGENCE CHARACTERISTICS OF FAST RLS BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $80^\circ$  AND  $-80^\circ$ .

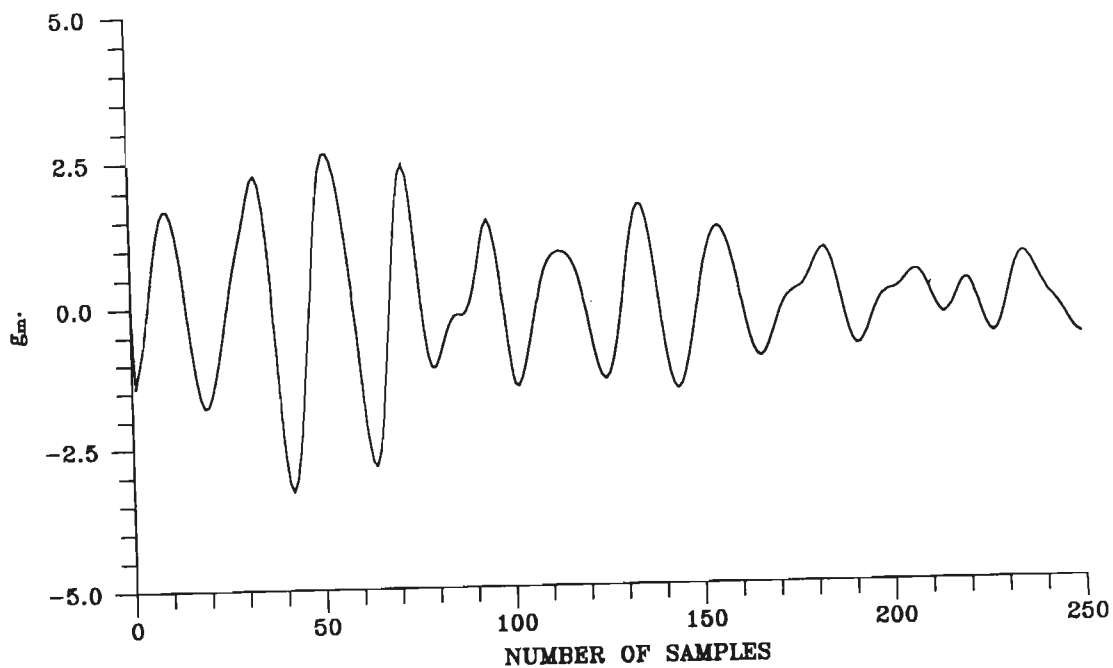




(a) MLSL BEAMFORMER



(b) MFTF BEAMFORMER



(c) QR-MLSL BEAMFORMER

FIG.4.8 OUTPUT SIGNAL WAVEFORMS OF THE THREE FAST RLS BROADBAND BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $80^\circ$  AND  $-80^\circ$ .

MFTF and QR-MLSL waveforms exhibit better characteristics, as in previous examples. A comparison of Fig.4.7 with Fig.3.20 reveals that the exact RLS beamformers exhibit superior performance compared to these fast-RLS beamformers.

From the above examples, it is clear that though the MLSL beamformer exhibits initial convergence, it produces a large residual power and tends to diverge with the increasing number of samples. In most of the situations it fails to faithfully reproduce the desired signal. Thus, it may be concluded that the MLSL beamformer suffers from numerical problems which can be attributed to the repeated computation of the inverses of the error variance matrices. Although MFTF and QR-MLSL beamformers exhibit better performance characteristics, the residual power is large compared with that obtained in the case of exact RLS beamformers.

Since weight computation is an integral part of the MFTF algorithm, the voltage patterns of the MFTF beamformer have been computed for the four examples and are shown in Fig.4.9-4.12.

It is found that in the first example, the MFTF beamformer places deep nulls of depth 110dB and 129dB (Fig.4.9) in the direction of interferences ( $\pm 60^\circ$ ), as does the exact RLS beamformers. However, it produces shallow nulls at  $45^\circ$  and  $60^\circ$ , thereby introducing a bias of  $-5^\circ$  for the interference arriving at  $50^\circ$  (Fig.4.10). When the interferences arrive very close to the desired signal ( $\pm 5^\circ$ ), the null depths are much smaller (-50dB) (Fig.4.11) than those produced by the exact RLS beamformers (-80dB). Finally, Fig.4.12 shows that the MFTF beamformer fails to place nulls in the direction of endfire interferences arriving at  $+80^\circ$  and  $-80^\circ$ .

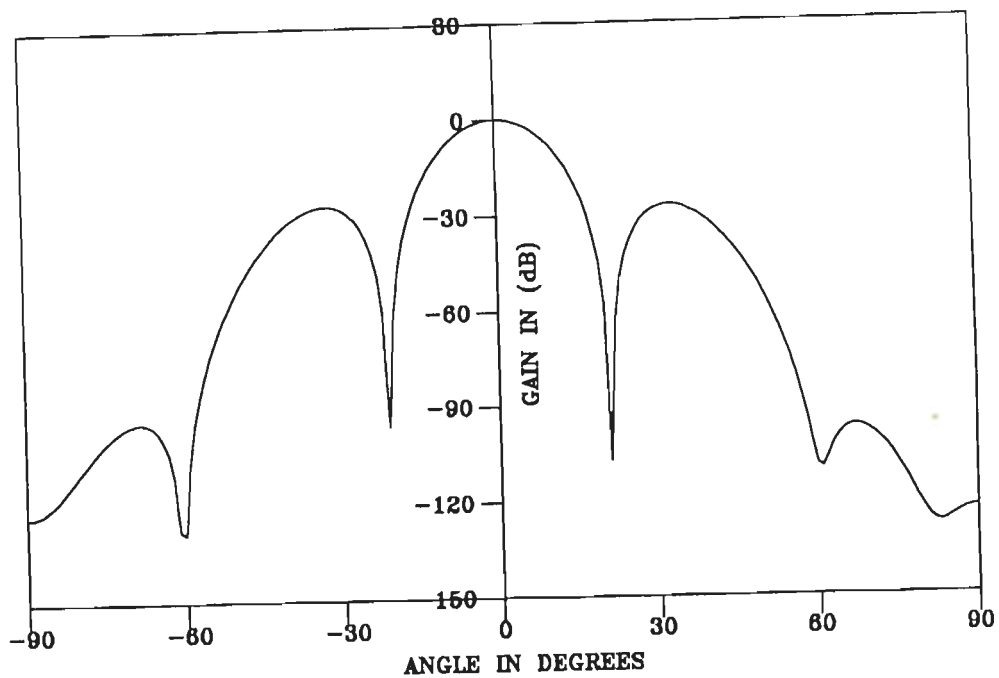


FIG.4.9 VOLTAGE PATTERN OF MFTF BEAMFORMER WITH INTERFERENCES ARRIVING AT  $60^\circ$  AND  $-60^\circ$ .

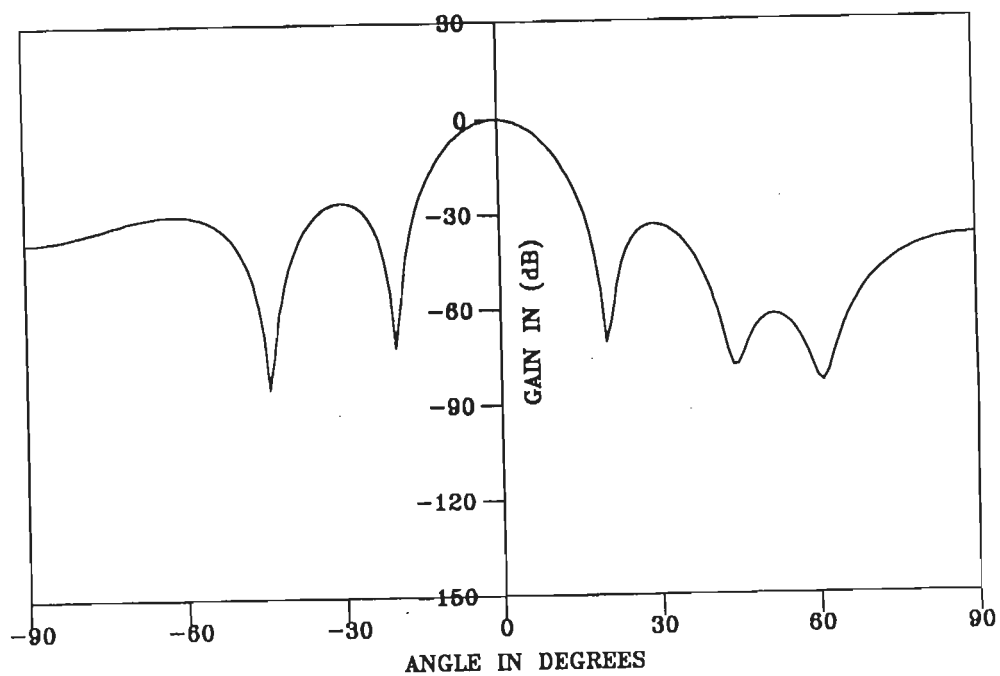


FIG.4.10 VOLTAGE PATTERN OF MFTF BEAMFORMER WITH INTERFERENCES ARRIVING AT  $50^\circ$  AND  $60^\circ$ .

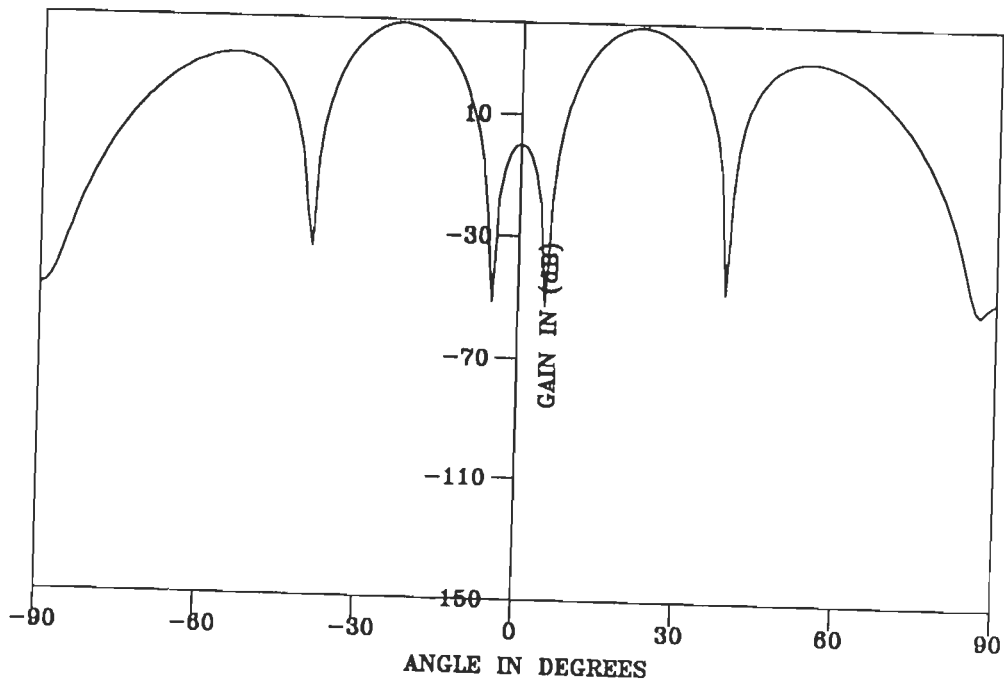


FIG.4.11 VOLTAGE PATTERN OF MFTF BEAMFORMER WITH INTERFERENCES ARRIVING AT  $5^\circ$  AND  $-5^\circ$ .

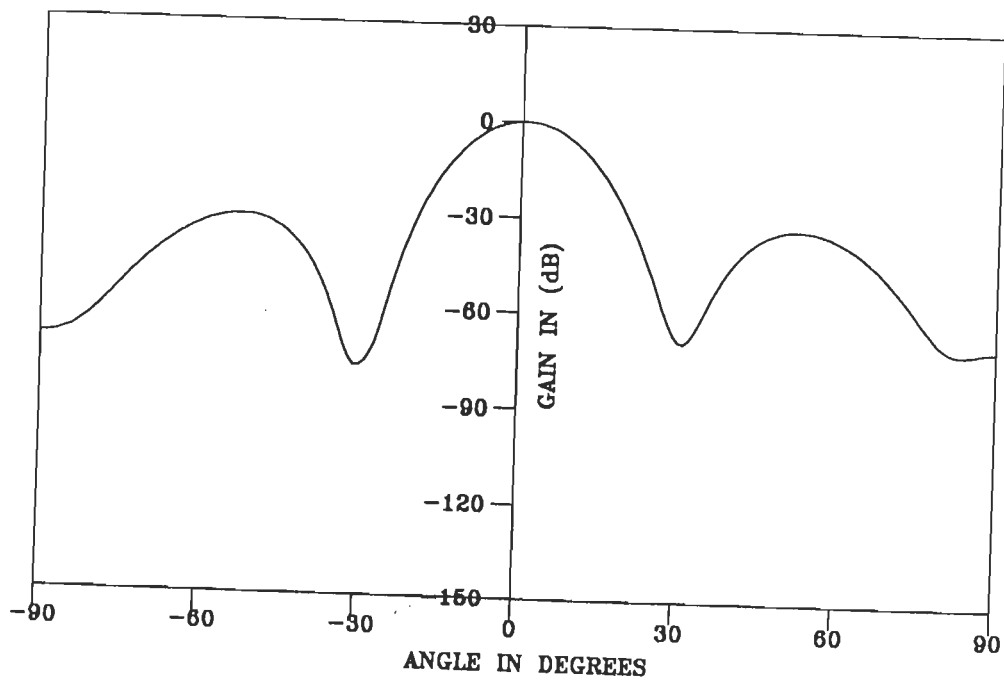


FIG.4.12 VOLTAGE PATTERN OF MFTF BEAMFORMER WITH INTERFERENCES ARRIVING AT  $80^\circ$  AND  $-80^\circ$ .

#### 4.5 CONCLUSIONS

In this chapter, the application of fast RLS algorithms, viz., the MLSL, MFTF and the QR-MLSL algorithms to the adaptive broadband beamforming problem has been investigated. These algorithms exploit the inherent delays available in the tapped delay line to arrive at beamforming techniques whose computational complexity is on the order of  $P^3M$ . The exact number of arithmetic operations required by these beamformers per time sample are given in Table 4.5.

**Table-4.5**  
**Computational Complexity of the Fast RLS Algorithm**  
**based broadband beamformers**

MLSL	MFTF	QR-MLSL
$(2P^3M + 7P^2M + P^2$ $+ 3P(M+1))$ $+ 4PM$ divisions $+ 2P^3(M-1)$ for computing the inverses	$(P^3M + 4P^2M + 3P^2$ $+ 2PM + 4P + 1)$ $+ 2P$ divisions $+ P^3$ for computing to inverses	$(9P^2M + 6.5PM)$ $+ 2PM$ Square roots

It is found that, of the three beamformers, MFTF beamformer has the least computational complexity. If the number of taps in the delay line are greater than five, the computational complexity of the MFTF beamformer is less than that of the RMGSEF beamformer. However, it is well known that by increasing the number of taps beyond four or five per channel, no significant gain is achieved in the performance of the beamformer [21].

The computer simulation results have shown that the MLSL beamformer fails to converge in different signal environments, even

after a large number of adaptation samples. Though the MFTF and the QR-MLSL beamformers exhibit better characteristics than MLSL, their performance is much inferior to that of the exact-RLS beamformers, so far as residual power, signal tracking and interference suppression are concerned.

It may, therefore, be concluded that for adaptive beamforming, the fast RLS beamformers have neither the advantages of computational complexity nor superior performance as compared to the exact RLS beamformers discussed in chapter 3.

## CHAPTER - 5

### ADAPTIVE BEAMFORMERS FOR COHERENT INTERFERENCE SUPPRESSION

In the preceding chapters, beamformers based on different adaptive algorithms have been studied in a noncoherent signal environment. Additional preprocessing is necessary to decorrelate the coherence among signal and interferences and to make these beamformers effective in a coherent signal environment. In this context, two techniques, namely, the spatial smoothing preprocessing scheme (SSPS) and the structured correlation matrix method (SCMM) were discussed in chapter 2. Through computer simulations it was shown that, of the two techniques SSPS has a superior ability for combating signal cancellation in coherent signal environment. Detailed analysis of SSPS by several authors [57,52,69] has shown that it suffers from reduced effective aperture area. A promising modification of SSPS, which results in an increased effective area, is the forward/backward spatial smoothing scheme (FBSS) which has been proposed and studied in the context of direction-of-arrival (DOA) estimation [50,66]. However, detailed investigations into its performance in beamforming applications have not been carried out, so far.

Suitable schemes are necessary to incorporate the above mentioned spatial averaging schemes in the adaptive beamformers discussed in chapters 3 and 4. This aspect has not received much attention in scientific literature. Only Shan and Kailath [57] discuss briefly a method of implementing SSPS in the LMS array. Similarly, Park and Un [45] have presented a parallel modified spatial smoothing (PMSS) technique which is a parallel implementation of the



forward/backward scheme. In this method, the array data are processed in two steps. First, a data-domain spatial preprocessing (DDSP) algorithm is applied, which expands the effective aperture area without forming covariance matrices. Then, a parallel implementation of the spatial smoothing technique is carried out to decorrelate the coherence of signal sources. They have realized this using the QRD-LS array.

The adaptive algorithms can be broadly divided into two groups, viz, the weight oriented algorithms and the residue oriented algorithms [70]. Weight oriented algorithms are those in which weight vector is updated explicitly at each time instant. The LMS, RLS and the MFTF algorithms are all weight oriented. On the other hand, orthogonalization-based algorithms such as, the RMGS and the Givens rotation based QRD-LS algorithms, do not involve any explicit computation of weights. These algorithms are known as residue or estimation error oriented algorithms. Different schemes are required to incorporate the spatial averaging techniques in these two categories of algorithms.

This chapter has been organized in the following manner. The FBSS has been discussed in the context of optimum beamformers in sec.5.1. Adaptive implementation of the SSPS and FBSS using various time recursive algorithms has been described in sec.5.2 and 5.3, respectively. Typical numerical examples are presented in sec.5.4 which demonstrate the performance of SSPS and FBSS with different adaptive algorithms. Finally, conclusions are drawn in sec.5.5 based on computer simulation studies.

## 5.1 THE FORWARD/BACKWARD SPATIAL SMOOTHING SCHEME

In addition to forward subarrays of the spatial smoothing scheme described in sec.2.3, the FBSS scheme makes use of complex conjugated backward subarrays to realize a larger effective aperture [50].

In the basic spatial smoothing preprocessing scheme (SSPS), a uniform linear array of 'P' sensors is extended by augmenting it with 'L' additional sensors. The extended array is then divided into (L+1) overlapping subarrays, each of size 'P', with first subarray formed from sensors  $\{1, \dots, P\}$ , the second from  $\{2, \dots, P+1\}$  and so on, as shown in Fig.5.1.

Using the notation of chapter 2, Let  $\underline{z}_1^f(n)$  denote the signal vector of the  $l^{\text{th}}$  forward subarray.

Then

$$\underline{z}_1^f(n) = \left[ x_1(n), x_{1+1}(n), \dots, x_{1+P-1}(n) \right]^T$$

$$l = 1, 2, \dots, L+1$$

... (5.1)

Then the covariance matrix of the  $l^{\text{th}}$  forward subarray is given by

$$\Phi_1^f = E \left[ \underline{z}_1^f(n) \left[ \underline{z}_1^f(n) \right]^H \right]$$

... (5.2)

The forward spatially smoothed correlation matrix,  $\bar{\Phi}^f$ , is defined as the mean of the forward subarray correlation matrices which can be written as

$$\bar{\Phi}^f = \frac{1}{L+1} \sum_{l=1}^{L+1} \Phi_l^f$$

... (5.3)

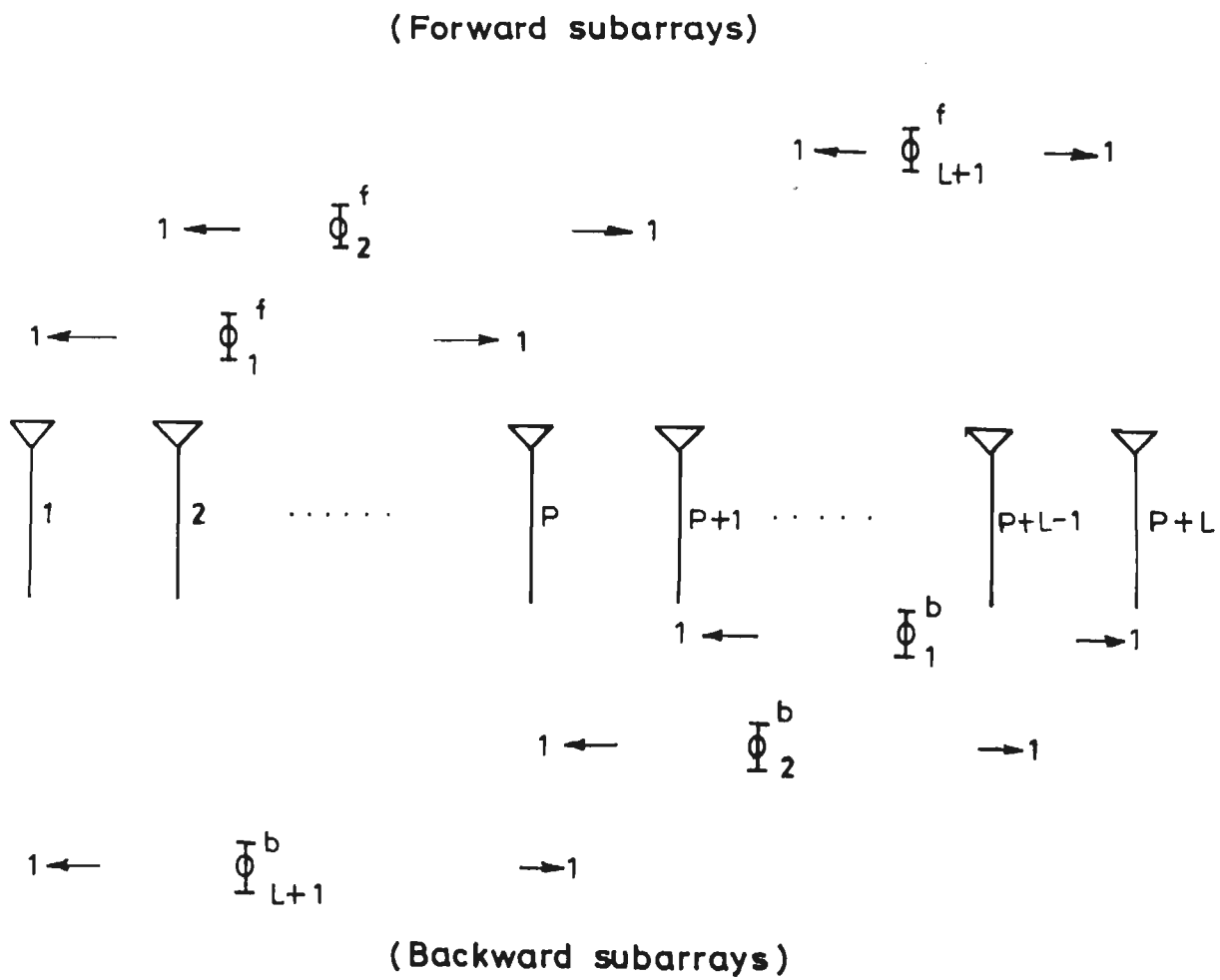


Fig. 5.1 The forward/backward spatial smoothing scheme

It was shown in sec. 2.3 that for this scheme to work, the minimum number of sensors needed is  $2K$  as compared to ' $K+1$ ' for the conventional array.

The forward/backward spatial smoothing scheme (FBSS) makes use of appropriate backward subarrays to reduce the required number of elements to  $(3K/2)$ .

Towards this purpose, additional ' $L+1$ ' backward subarrays are generated from the same set of sensors by grouping elements at  $\{P+L, P+L-1, \dots, L+1\}$  to form the first backward subarray, elements of  $\{P+L-1, P+L-2, \dots, L\}$  to form the second and so on, as shown in Fig.5.1. Let  $\underline{z}_l^b(n)$  denote the  $l^{\text{th}}$  backward subarray signal vector defined as

$$\underline{z}_l^b(n) = \left[ x_{P+L-l+1}^*(n), x_{P+L-l}^*(n), \dots, x_{L+1-l+1}^*(n) \right]^T$$

$$l = 1, 2, \dots, L+1$$

... (5.4)

Then the correlation matrix of the  $l^{\text{th}}$  backward subarray is given by

$$\Phi_l^b = E \left[ \underline{z}_l^b(n) \left[ \underline{z}_l^b(n) \right]^H \right]$$

... (5.5)

We define the spatially smoothed backward subarray covariance matrix,  $\bar{\Phi}^b$ , as the mean of the subarray correlation matrices, ie,

$$\bar{\Phi}^b = \frac{1}{L+1} \sum_{l=1}^{L+1} \Phi_l^b$$

... (5.6)

Recalling the discussion in chapter 2, it is evident that the backward spatially smoothed covariance matrix will be of full rank when  $L+1 \geq K$ . It, therefore, follows that the backward spatial smoothing scheme also requires at least  $2K$  sensors to restore the rank of the correlation matrix to  $K$ .

In the forward/backward spatial smoothing scheme, simultaneous use of forward and backward averaging schemes is made. The forward/backward smoothed correlation matrix,  $\bar{\Phi}$ , is defined as the mean of  $\bar{\Phi}^b$  and  $\bar{\Phi}^f$  [50]. That is,

$$\bar{\Phi} = \left[ \bar{\Phi}^b + \bar{\Phi}^f \right] / 2.0 \quad \dots (5.7)$$

This matrix  $\bar{\Phi}$  is then used in an optimum beamformer to compute the weight coefficients of the array.

In general, the modified correlation matrix will be non-singular, regardless of the coherence of 'K' signal sources, so long as  $2(L+1) \geq K$  [50]. Recalling that in the presence of 'K' signals, the size 'P' of each subarray must be atleast 'K+1' and since, 'L+1' overlapping subarrays are formed in one direction, it follows that if  $P_1$  is the total number of subarrays needed, then

$$P_1 \geq (L+1) + P-1 \geq K/2 + K$$

or

$$P_1 \geq \frac{3K}{2} \quad \dots (5.8)$$

## 5.2 ADAPTIVE IMPLEMENTATION OF THE SPATIAL SMOOTHING SCHEME

Although optimum beamformer discussed in sec.2.4 can be utilized to combat coherent interferences, it is computationally

expensive to estimate the correlation matrix and to compute its inverse. Further, the correlation matrix becomes illconditioned when strong jammers coexist with the desired signal [70] and the method often required double precision, which significantly decreases the beamformer throughput. Therefore, in real time applications, adaptive beamformers based on time recursive algorithms are used. Although several authors have analysed SSPS and suggested modifications into the basic scheme, its adaptive implementation has received little attention. In this section, we present two separate schemes to incorporate SSPS in adaptive beamformers : one for beamformers based on weight oriented algorithms and the other for QR-decomposition based arrays.

### 5.2-1 Adaptive Processing

In SSPS, the spatially smoothed correlation matrix is defined as in eqn.(2.67)as

$$\bar{\Phi} = \frac{1}{L+1} \sum_{l=1}^{L+1} \Phi_{zz}^{(l)} \quad \dots (5.9)$$

where

$$\hat{\Phi}_{zz}^{(l)} = E \left[ Z_l(n) \left[ Z_l(n) \right]^H \right] \quad \dots (5.10)$$

$$Z_l(n) \text{ being } \left[ x_l(n), x_{l+1}(n), \dots, x_{l+P-1}(n) \right]^T$$

We can write the estimate of the spatially smoothed correlation matrix as

$$\begin{aligned}\hat{\Phi}_N &= \frac{1}{L+1} \sum_{l=1}^{L+1} \left[ \frac{1}{N} \sum_{n=1}^N \underline{z}_l(n) \underline{z}_l^H(n) \right] \\ &= \frac{1}{N(L+1)} \sum_{n=1}^N \sum_{l=1}^{L+1} \left[ \underline{z}_l(n) \underline{z}_l^H(n) \right] \quad \dots (5.11)\end{aligned}$$

Then, by analogy

$$\begin{aligned}\hat{\Phi}_{N+1} &= \frac{1}{(N+1)(L+1)} \sum_{n=1}^{N+1} \sum_{l=1}^{L+1} \underline{z}_l(n) \underline{z}_l^H(n) \\ &= \frac{1}{(N+1)(L+1)} \sum_{n=1}^N \sum_{l=1}^{L+1} \underline{z}_l(n) \underline{z}_l^H(n) \\ &\quad + \frac{1}{(N+1)(L+1)} \sum_{l=1}^{L+1} \underline{z}_l(N+1) \underline{z}_l^H(N+1) \\ &= \frac{N}{(N+1)} \hat{\Phi}_N + \frac{1}{(N+1)(L+1)} \sum_{l=1}^{L+1} \underline{z}_l(N+1) \underline{z}_l^H(N+1) \quad \dots (5.12)\end{aligned}$$

The above equation suggests that the inverse of the spatially smoothed matrix,  $\hat{\Phi}$ , can be recursively updated by using the matrix inversion lemma iteratively (L+1) times, once for each  $\underline{z}_l$ . Similarly, SSPS can easily be implemented in the LMS, RLS and MFTF algorithms.

### 5.2-1.1 Implementation using weight oriented algorithms

In this subsection, we illustrate the adaptive implementation of the spatial smoothing scheme in weight oriented

algorithms by incorporating the scheme in the conventional RLS algorithm. Fig.5.1 shows how to form the spatial data subset from one "snapshot" and a flow-chart of the procedure is shown in Fig.5.2. At each time instant, the snapshot of 'P+L' data samples is divided into 'L+1' overlapping subgroups of 'P' data samples each. These subgroups are then fed one by one into the RLS algorithm based adaptive processor which updates the P-dimensional weight vector each time. After all the subgroups have been processed, the same procedure is repeated with the next data snapshot. Once the adaptation process is over, the weight vector, so obtained, is used to compute the array output. The algorithm is summarized in Table 5.1.

#### 5.2-1.2 Implementation using QR decomposition algorithms

To illustrate the adaptive implementation of QR-decomposition based algorithms, we consider the RMGSEF algorithm for which the flow-chart is shown in Fig.5.3. As in the previous case, at each instant of time, the snapshot of 'P+L' data samples is divided into (L+1) overlapping subgroups of 'P' data samples each (Fig.5.1). The subgroups are then fed, one by one, in succession into the RMGSEF algorithm based processor, which updates the elements of the upper triangular matrix 'R' and the associated 1-by-P vector  $\underline{K}^d(n)$  (chapter3) each time. After all the subgroups have been processed, for a fixed number of snapshots, say n, where  $n(L+1) > 2P$  for obtaining convergence, the elements of the upper triangular matrix 'R' and the



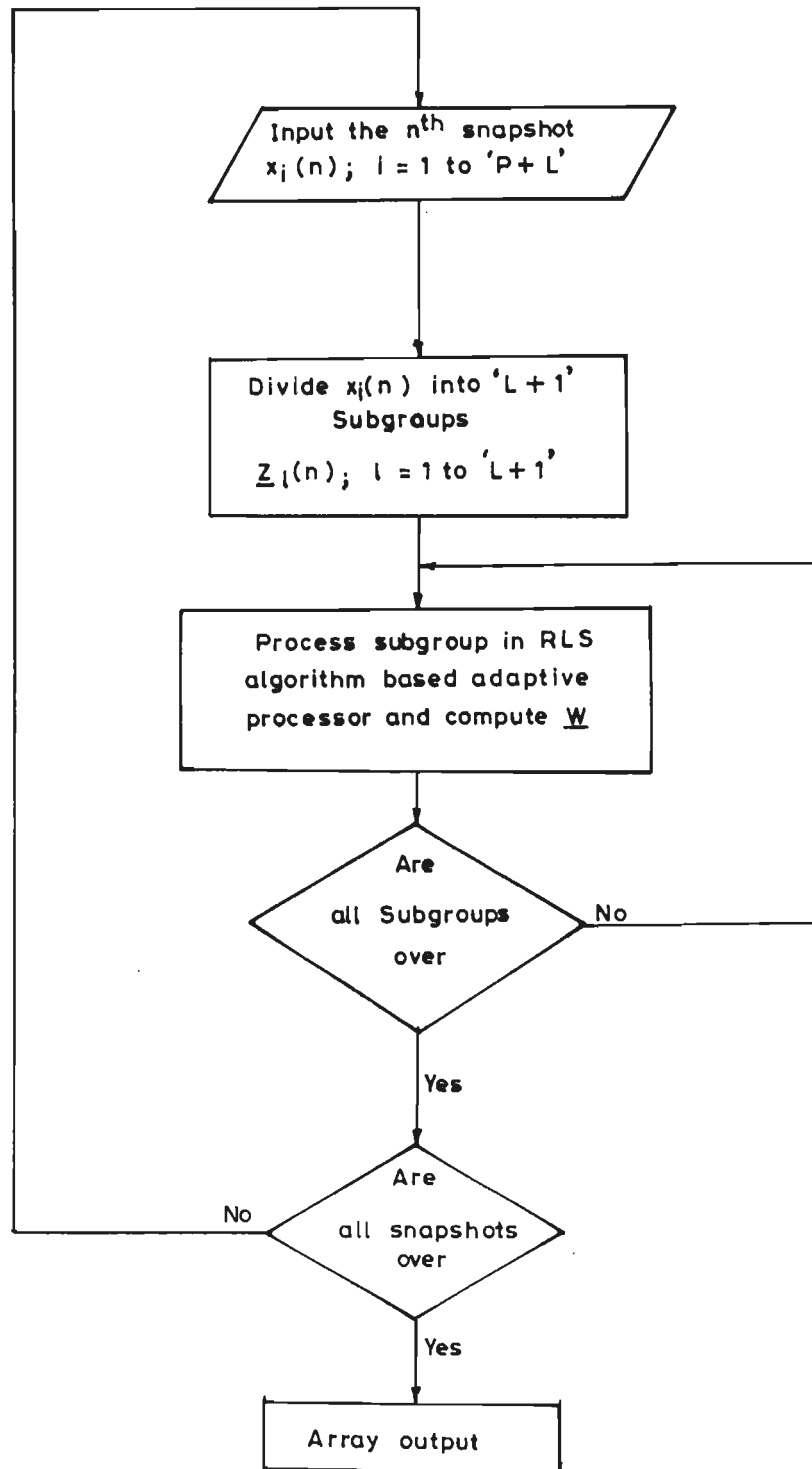


Fig. 5.2 Flow-chart for adaptive implementation of spatial smoothing scheme using the RLS adaptive processor

**Table 5.1**  
**RLS Algorithm for Adaptive Implementation**  
**of Spatial Smoothing Scheme**

Algorithm	Complexity
Initialization	
$\Phi^{-1}(0) = \delta^{-1} I,$	$\delta = \text{small + ve constant}$
$\underline{W}(0) = 0$	
for $n = 1, 2, \dots$ compute	
For $l = 1$ to $L+1$	
For $i = 1$ to $P$	
$Z_l^1(n) = x_{i+l-1}(n)$	(T 5.1.1)
$\underline{u}(n) = \lambda^{-1} \Phi^{-1}(n-1) \underline{Z}_l(n)$	$P^2(L+1)$ (T 5.1.2)
$\underline{C}(n) = \underline{u}(n) / [1 + \underline{Z}_l^H(n) \underline{u}(n)]$	$P(L+1)$ (T 5.1.3)
$\eta(n) = d(n) - \underline{W}^H(n-1) \underline{Z}_l(n)$	$P(L+1)$ (T 5.1.4)
$\underline{W}(n) = \underline{W}(n-1) + \underline{C}(n) \eta^*(n)$	$P(L+1)$ (T 5.1.5)
$\Phi^{-1}(n) = \lambda^{-1} \Phi^{-1}(n-1) - \underline{C}(n) \underline{u}^H(n)$	$2P^2(L+1)$ (T 5.1.6)
Total	$(P^2+2P)(L+1)$ divisions + $(3P^2+3P)(L+1)$

associated 1-by-P vector  $\underline{K}^d(n)$  are clocked into the linear systolic array to generate the optimum weight vector. The weight vector,  $\underline{W}$ , so obtained is then used to generate the array output.

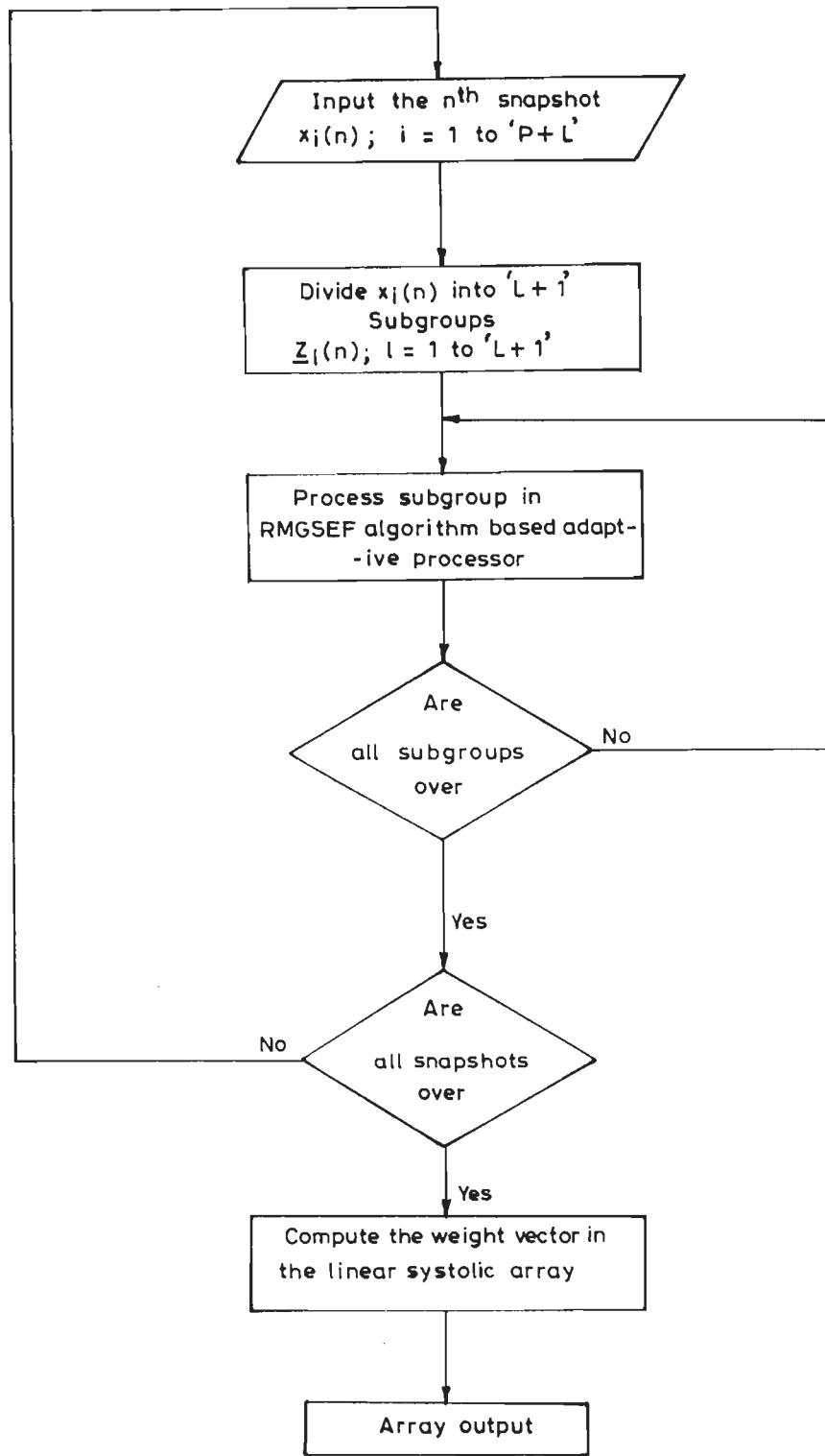


Fig. 5.3 Flow-chart for adaptive implementation of spatial smoothing scheme using RMGSEF adaptive processor

Here, spatial smoothing is effected on the elements of the upper triangular matrix 'R' and the associated 1-by-P vector  $\underline{K}^d(n)$ , because updating is carried out using the overlapping subgroups of each snapshot in succession. This is similar to adaptive implementation of the spatial smoothing scheme using LMS or RLS algorithms, where P-dimensional weight vector is updated in the same fashion. The RMGSEF algorithm for adaptive implementation of the spatial smoothing scheme is summarized in Table 5.2.

The implementation of the spatial smoothing scheme using the rotation based QRD-LS algorithm slightly differs in detail from that of the RMGS class of algorithms. In the case of RMGS algorithms, the reference signal  $d(n)$  is a part of the processing itself, whereas in the QRD-LS algorithm, the reference signal  $d(n)$  is treated as an appended element of the signal vector. Therefore, while implementing the spatial smoothing scheme, the reference signal should be appended to the subarray signal vector. The QRD-LS algorithm for implementing the spatial smoothing scheme is given in Table 5.3.

**Table - 5.2**

**RMGSEF Algorithm for Adaptive Implementation  
of the Spatial Smoothing Scheme**

---

Input definitions

$$Q_i^{(1)}(n,n) = x_i(n), \quad i = 1, 2, \dots, P+L \quad (T.5.2.1)$$

$$e_i^{(1)}(n,n) = d(n), \quad r_{i1}^{(1)}(0) = \delta, \quad i = 1, 2, \dots, P \quad (T.5.2.2)$$

Algorithm	Complexity
For $n = 1, 2, \dots$ compute	
For $l = 1$ to $L+1$	
$\alpha_1(n) = 1.0$	(T.5.2.3)
For $m = 1$ to $P$ do	
$q_m^{(1)}(n) = Q_{m+l-1}^{(1)}(n, n)$	(T.5.2.4)
For $i = 1$ to $P$ do	
$q_i(n, n) = q_i^{(i)}(n, n)$	
$r_{ii}(n) = \lambda r_{ii}(n-1) + \alpha_i(n)  q_i(n, n) ^2$	3P(L+1) (T.5.2.5)
$\alpha_{i+1}(n) = \alpha_i(n) - \alpha_i^2(n)  q_i(n, n) ^2 / r_{ii}(n)$	3P(L+1) (T.5.2.6)
For $j = i + 1$ to $P$ do	
$q_j^{(i+1)}(n, n) = q_j^{(i)}(n, n) - K_{ij}(n-1) q_i(n, n)$	P(P-1)(L+1)/2
	(T.5.2.7)
$K_{ij}(n) = K_{ij}(n-1) + \alpha_i(n) q_j^{(i+1)}(n) q_i^*(n, n) / r_{ii}(n)$	P(P-1)(L+1)
	(T.5.2.8)
$e^{(i+1)}(n, n) = e^{(i)}(n, n) - K_i^d(n-1) q_i(n, n)$	P(L+1)
	(T.5.2.9)
$K_i^d(n) = K_i^d(n-1) + \alpha_i(n) e^{(i+1)}(n, n) q_i^*(n, n) / r_{ii}(n)$	2P(L+1)
	(T.5.2.10)
$e(n) = e^{(i+1)}(n, n)$	(T.5.2.11)
Total	(0.5P <sup>2</sup> + 1.5P)(L+1) divisions + (1.5P <sup>2</sup> + 7.5P)(L+1)

**Table - 5.3**  
**The QRD-LS Algorithm for Implementation of**  
**Spatial Smoothing Scheme**

Input definitions	
$Q_i^{(1)}(n) = x_i(n),$	$i = 1, 2, \dots, P+L$ (T 5.3.1)
$e(n) = d(n)$	(T 5.3.2)
Algorithm	Complexity
For $n = 1, 2, \dots$ do	
For $l = 1$ to $L+1$	
$\alpha_1(n) = 1.0$	(T 5.3.3)
For $m = 1$ to $P$ do	
$q_m^{(1)}(n) = Q_{m+l-1}^{(1)}(n)$	(T 5.3.4)
$q_{P+l}^{(1)}(n) = e(n)$	(T 5.3.5)
For $i = 1, 2, \dots, P+1$	
$r_{ii}(n) = \sqrt{ \lambda^{1/2} r_{ii}(n-1) ^2 +  q_i^{(1)}(n) ^2}$	3(P+1)(L+1) (T 5.3.6)
If $ r_{ii}(n)  > 0, C_i =  r_{ii}(n-1) /r_{ii}(n)$	(T 5.3.7)
$S_i = q_i^{(1)}(n)/r_{ii}(n)$	(T 5.3.8)
else $C_i = 1.0, S_i = 0.0$	
$\alpha_{i+1}(n) = \alpha_i C_i$	(P+1) (T 5.3.9)
For $j = i+1, \dots, P+1$	
$r_{ij}(n) = C_i \lambda^{1/2} r_{ij}(n-1) + S_i^* q_j^{(1)}(n)$	3P(P+1)(L+1)/2 (T 5.3.10)
$q_j^{(i+1)}(n) = -S_i r_{ij}(n-1) \lambda^{1/2} + C_i q_j^{(i)}(n)$	2P(P+1)(L+1)/2 (T 5.3.11)
Total 2(P+1)(L+1) divisions + (P+1)(L+1) Sq. roots + (2.5P <sup>2</sup> +4.5P)(L+1)	

### 5.3 ADAPTIVE IMPLEMENTATION OF THE FORWARD/BACKWARD SPATIAL SMOOTHING SCHEME

In the forward/backward spatial smoothing scheme, an array of 'P+L' sensors is divided into 'L+1' forward and 'L+1' backward subarrays as shown in Fig.5.1. Park and Un [45] have proposed an alternative method of forming the backward subarrays which in combination with the forward subarrays leads to the same correlation matrix as that for the forward/backward scheme described in sec.5.1, except for the scaling factor 1/2 whose value has no effect on the beamformer function. In their approach  $l^{\text{th}}$  backward subarray signal vector  $\underline{z}_l^b(n)$  is defined as

$$\begin{aligned} \underline{z}_l^b(n) &= J \left[ \underline{z}_l^f(n) \right]^* \\ &= \left[ x_{l+P-1}^*(n), x_{l+P-2}^*(n), \dots, x_{l+1}^*(n), x_l^*(n) \right]^T \end{aligned} \quad \dots (5.13)$$

where J is the P-by-P exchange matrix defined by

$$J = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \dots (5.14)$$

We next introduce a set of vectors  $\{\underline{r}_l(n)\}$ ,  $l = 1, 2, \dots, L+1$

$$\underline{r}_l(n) = \frac{1}{2} \left[ \underline{z}_l^f(n) + \underline{z}_l^b(n) \right] \quad \dots (5.15)$$

Then, it can be shown that the spatially smoothed correlation matrix of  $\underline{r}_l(n)$  is given by

$$\begin{aligned} \Phi &= \frac{1}{L+1} \sum_{l=1}^{L+1} E \left[ \underline{r}_l(n) \underline{r}_l^H(n) \right] \\ &= \frac{1}{4(L+1)} \sum_{l=1}^{L+1} \left\{ E \left[ \underline{z}_l^f(n) \left[ \underline{z}_l^f(n) \right]^H \right] + J E \left[ \left[ \underline{z}_l^f(n) \right]^* \left[ \underline{z}_l^f(n) \right]^H \right] \right. \\ &\quad \left. + E \left[ \underline{z}_l^f(n) \left[ \underline{z}_l^f(n) \right]^H J \right] + J E \left[ \left[ \underline{z}_l^f(n) \right]^* \left[ \underline{z}_l^f(n) \right]^T \right] J \right\} \quad \dots (5.16) \end{aligned}$$

If the signals and noise are assumed to be band limited with zero-mean Gaussian characteristics, then using the identity  $E [\underline{u} \underline{u}^T] = E[\underline{u}^* \underline{u}^H] = 0$ , the second and third terms of the above equation vanish. We may, therefore, rewrite eqn. (5.16) as

$$\Phi = \frac{1}{(L+1)} \sum_{l=1}^{L+1} \left[ \Phi_l + J \Phi_l^* J \right] \quad \dots (5.17)$$

where

$$\Phi_l = E \left[ \underline{z}_l^f(n) \left[ \underline{z}_l^f(n) \right]^H \right] \quad \dots (5.18)$$

Therefore,  $\underline{r}_l(n)$  constitutes a new input vector whose spatially smoothed correlation matrix is equivalent to that obtained using eqn. 5.7 [45].





### 5.3-1 Implementation Using Weight Oriented Algorithms

The implementation of the forward/backward smoothing scheme using different weight oriented algorithms is similar to the adaptive implementation of spatial smoothing scheme employing these algorithms. The flow-chart for implementing the forward/backward scheme in RLS algorithm based processor is shown in Fig.5.4. From each snapshot, the data subsets for the forward subarrays are formed as shown in Fig.5.1. Next, a new set of data vectors  $\underline{r}_l(n)$ , for  $l = 1, 2, \dots, L+1$ , are formed in accordance with eqn. (5.15). These data vectors are then fed, one by one, into the RLS algorithm based adaptive processor which updates the P-dimensional weight vector. After all the data vectors  $\underline{r}_l(n)$ ,  $l=1, 2, \dots, L+1$ , are processed, the same procedure is repeated for the next snapshot of data samples. When the adaptation process is over, the updated weight vector is used to compute the array output. The RLS algorithm implementing the forward/backward smoothing scheme is summarized in Table 5.4.

### 5.3-2 Implementation Using QR-Decomposition Algorithms

The implementation of forward/backward scheme using QR-decomposition algorithms is similar to the spatial smoothing schemes implementation using these algorithms. After forming the data subsets for the forward subarrays from each snapshot, a new set of data vectors  $\underline{r}_l(n)$ , for  $l=1, 2, \dots, L+1$  are formed in accordance with eqns (5.13) and (5.15). Next, the data vectors are processed in the RMGSEF algorithm based processor. The flow-chart of the implementation scheme and the RMGSEF algorithm which implements the scheme are, respectively, given in Fig.5.5 and Table 5.5.

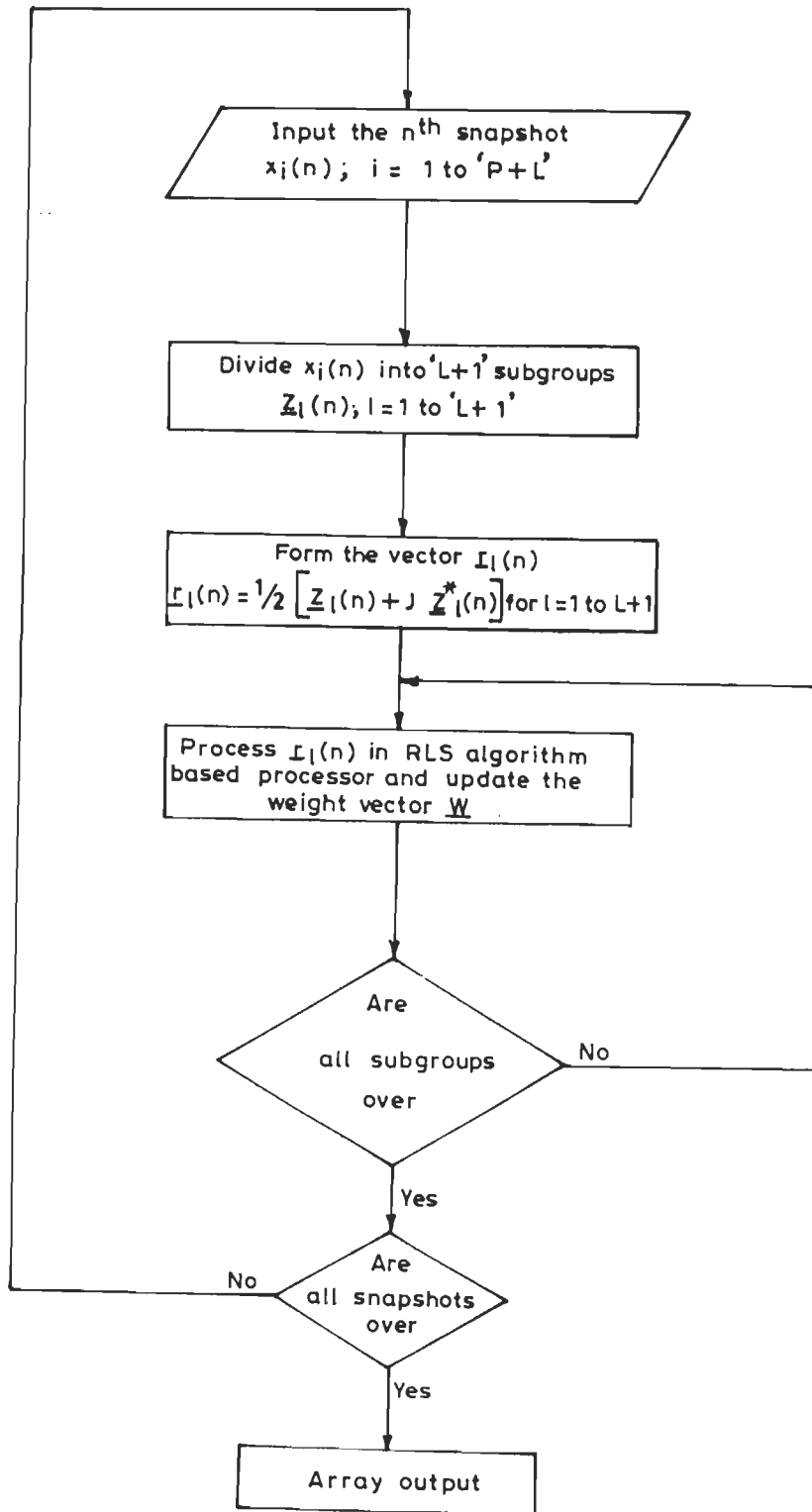


Fig.5.4 Flow-chart for adaptive implementation of forward/backward smoothing scheme using RLS adaptive processor

Table - 5.4

RLS Algorithm for Adaptive Implementation of Forward/Backward Scheme

Initialization

$$\Phi^{-1}(0) = \delta^{-1}I, \quad \delta = \text{Small + ve constant}$$

$$\underline{W}(0) = 0$$

Algorithm

Complexity

for  $n = 1, 2, \dots$  compute

For  $l = 1$  to  $L+1$

$$\left[ \begin{array}{l} \text{For } i = 1 \text{ to } P \\ z_l^i(n) = x_{i+l-1}(n) \end{array} \right. \quad (T \ 5.4.1)$$

$$\underline{r}_l(n) = \frac{1}{2} \left[ \underline{z}_l(n) + J \underline{z}_l^*(n) \right] \quad P^2(L+1) \quad (T \ 5.4.2)$$

$$\underline{u}(n) = \lambda^{-1} \Phi^{-1}(n-1) \underline{r}_l(n) \quad P^2(L+1) \quad (T \ 5.4.3)$$

$$\underline{c}(n) = \frac{\underline{u}(n)}{\left[ 1 + \underline{r}_l^H(n) \underline{u}(n) \right]} \quad P(L+1) \quad (T \ 5.4.4)$$

$$\eta(n) = d(n) - \underline{w}^H(n-1) \underline{r}_l(n) \quad P(L+1) \quad (T \ 5.4.5)$$

$$\underline{W}(n) = \underline{W}(n-1) + \underline{c}(n) \eta^*(n) \quad P(L+1) \quad (T \ 5.4.6)$$

$$\Phi^{-1}(n) = \lambda^{-1} \Phi^{-1}(n-1) - \underline{c}(n) \underline{u}^H(n) \quad 2P^2(L+1) \quad (T \ 5.11.7)$$

Total

$$4P^2(L+1)$$

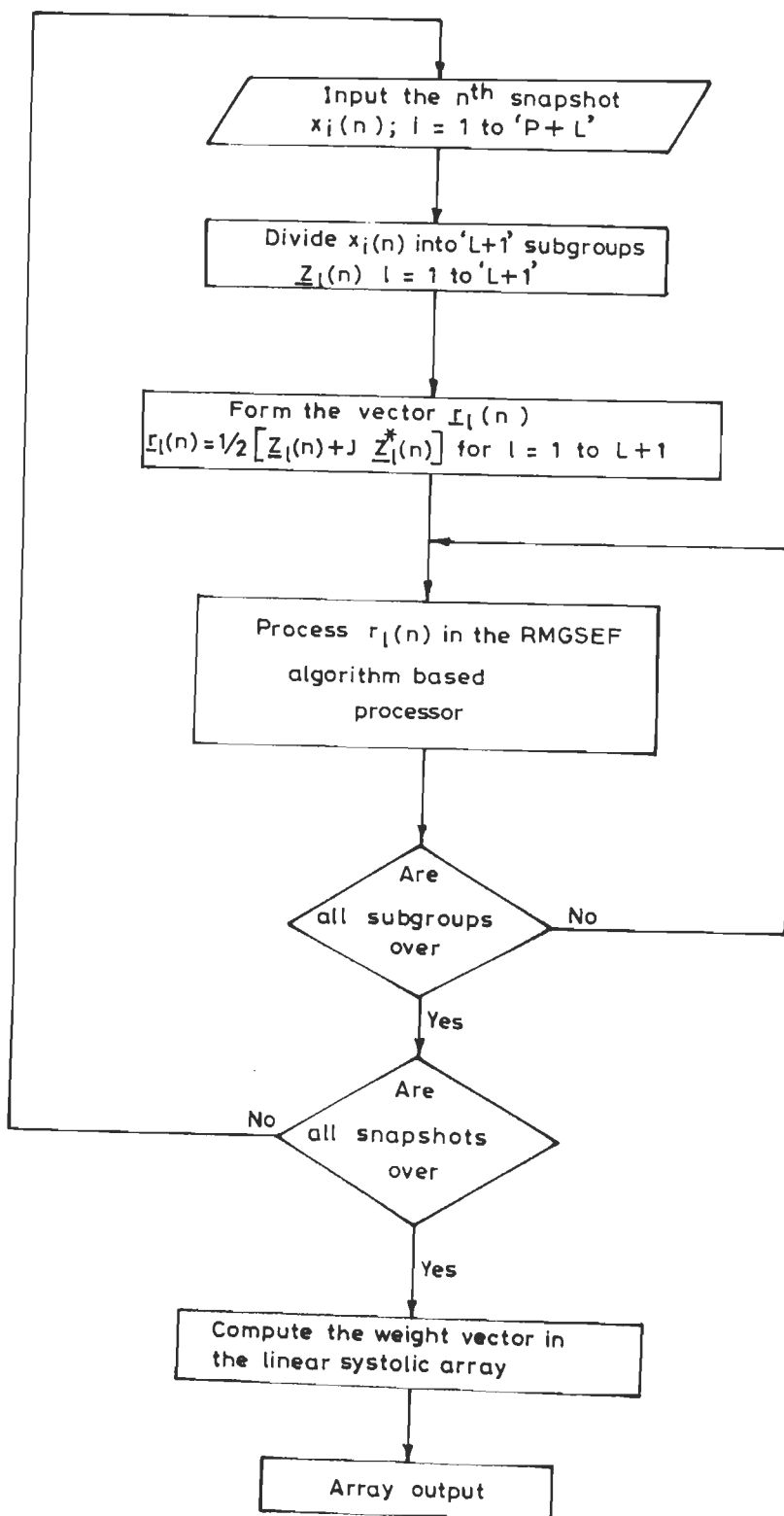


Fig. 5.5 Flow-chart for adaptive implementation of forward/backward smoothing scheme using RMGSEF adaptive processor

Table - 5.5

**RMGSEF Algorithm for Adaptive Implementation of  
Forward/Backward Smoothing Scheme**

Input definitions

$$Z_i^{(1)}(n) = x_i(n), \quad i = 1, 2, \dots, P+L \quad (T 5.5.1)$$

$$e^{(1)}(n) = d(n), \quad r_{11}(0) = \delta, \quad i = 1, 2, \dots, P \quad (T 5.5.2)$$

Algorithm	Complexity
For $n = 1, 2, \dots$ , Compute	
For $l = 1$ to $L+1$	
$\alpha_1(n) = 1.0$	(T 5.5.3)
[ For $m = 1$ to $P$	
$Z_l^1(n) = x_{m+l-1}(n)$	(T 5.5.4)
$r_l(n) = \frac{1}{2} [Z_l(n) + J Z_l^*(n)]$	$P^2+1$ (T 5.5.5)
[ For $i = 1$ to $P$	
$q_i^{(1)}(n,n) = r_i^1(n)$	(T 5.5.6)
For $i = 1$ to $P$	
$r_{11}(n) = \lambda r_{11}(n-1) + \alpha_i(n)  q_i(n,n) ^2$	$3P(L+1)$ (T 5.5.7)
$\alpha_{i+1}(n) = \alpha_i(n) - \frac{\alpha_i^2(n)  q_i(n,n) ^2}{r_{11}(n)}$	$3P(L+1)$ (T 5.5.8)
For $j = i + 1$ to $P$	

$$q_j^{(i+1)}(n, n) = q_j^{(i)}(n, n) - K_{ij}^{(n-1)} q_i(n, n) \quad P(P-1)(L+1)/2$$

(T 5.5.9)

$$K_{ij}(n) = K_{ij}^{(n-1)} + \alpha_i(n) q_j^{(i+1)}(n, n) q_i^*(n, n)/r_{ii}(n) \quad P(P-1)(L+1)$$

(T 5.5.10)

$$e^{(i+1)}(n, n) = e^{(i)}(n, n) - K_{i1}^d(n-1) q_1(n, n) \quad P(L+1)$$

(T 5.5.11)

$$K_{i1}^d(n) = K_{i1}^d(n-1) + \alpha_i(n) e^{(i+1)}(n, n) q_1^*(n, n)/r_{11}(n) \quad 2P(L+1)$$

(T 5.5.12)

$$e(n) = e^{(i+1)}(n, n) \quad (T 5.5.13)$$

Total	$\left[ (0.5P^2 + 1.5P) \text{ divisions} + (2.5P^2 + 7.5P) \right] (L+1)$
-------	--

#### 5.4 COMPUTER SIMULATIONS

In this section, the two spatial averaging preprocessing schemes, namely, SSPS and FBSS, have been compared with regard to their ability to overcome the coherent interference problem in adaptive beamformers. RMGSEF and QRD-LS adaptive processors have been used to implement the beamformers, as they have exhibited excellent numerical and nulling properties.

In all the examples presented in this section, the signal environment consists of 5 narrowband signals : a desired signal and four fully coherent interferences. Therefore, the minimum number of elements required is, respectively 10 and 8 for SSPS and FBSS beamformers. A uniform linear array of 6 isotropic elements has been assumed, as the subarray size should be atleast 6 elements. The SSPS

has been implemented by augmenting the array with four elements and forming five overlapping subarrays of 6 elements each. In the case of FBSS, the 6-element linear array has been augmented with 2 elements and three forward and three backward subarrays have been formed.

During the computer simulations it was found that, in some typical signal environments, the minimum overall array size and the number of subarrays was not sufficient to decorrelate the interferences and the desired signal. In such cases, the number of subarrays were increased till the desired decorrelation was achieved.

**Example 5.4-1**

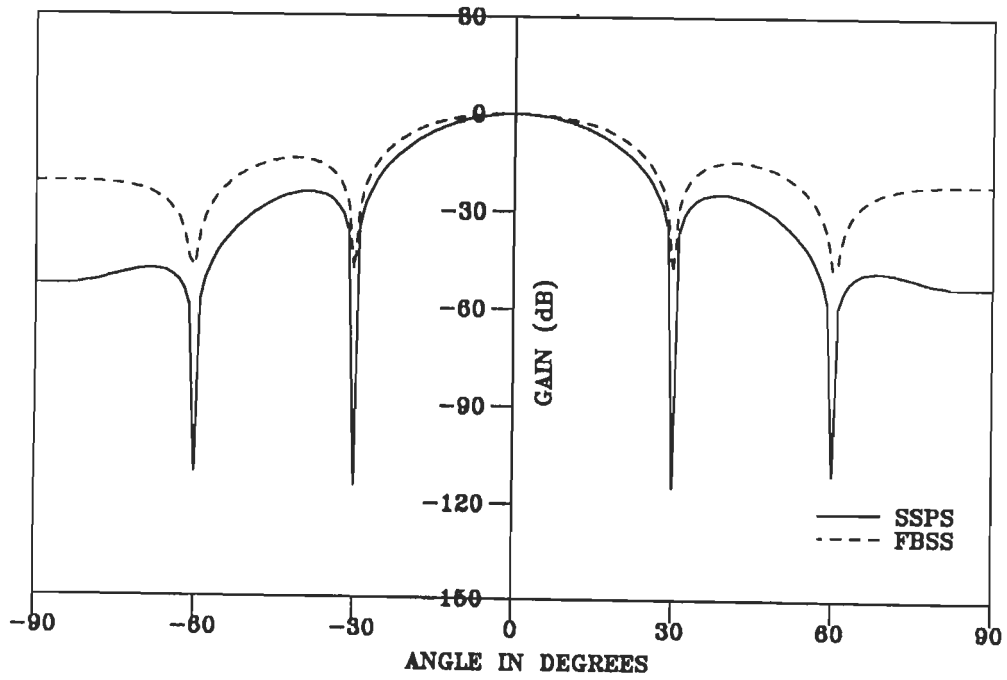
In this example, the interferences have been modelled to have wide angular separation. The various parameters of the interferences are given in Table 5.6.

**Table - 5.6**

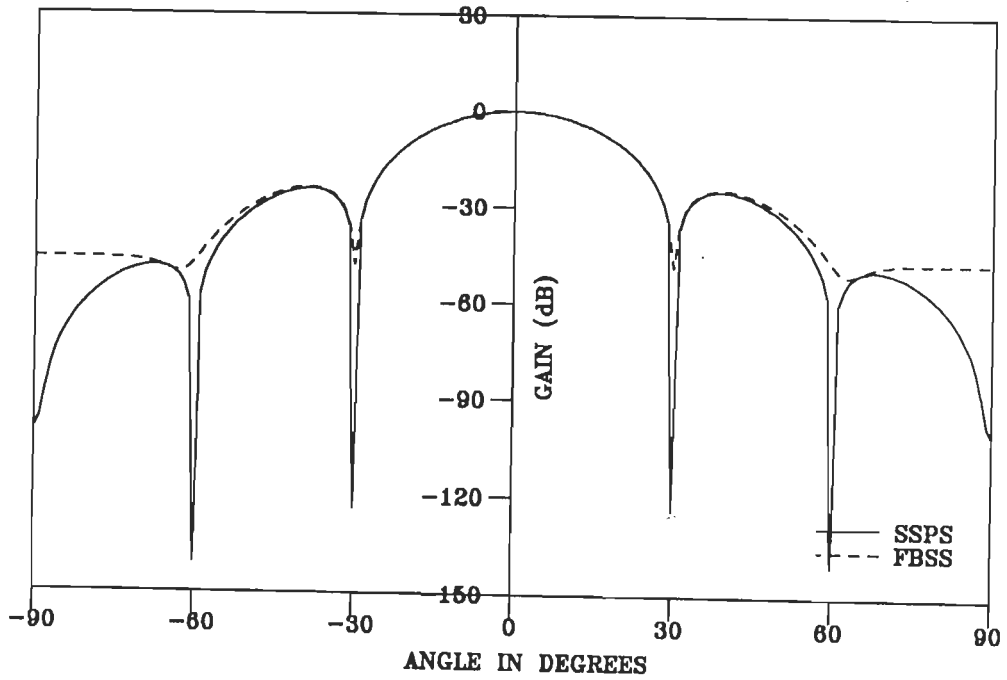
**Interference Parameters for Example 5.4-1**

Parameter	Interference 1	Interference 2	Interference 3	Interference 4
$\theta_1$	$30^\circ$	$-30^\circ$	$60^\circ$	$-60^\circ$
$S_1$	10.0	10.0	10.0	10.0
$\omega_1$	1.0	1.0	1.0	1.0

Fig. 5.6 shows the voltage patterns of the RMGSEF and QRD-LS beamformers. It can be seen from the figure that the SSPS-RMGSEF and SSPS-rotation based QRD-LS beamformers place deep nulls (-120dB) in the direction of all the four interferences. On the other hand, the FBSS-RMGSEF and FBSS-rotation based QRD-LS beamformers place



(a) RMGSEF BEAMFORMER



(b) QRD-LS BEAMFORMER

FIG.5.6 VOLTAGE PATTERNS OF SSPS AND FBSS ADAPTIVE BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ, -30^\circ, 60^\circ$  &  $-60^\circ$ .



relatively shallow nulls in the range of -45dB to -50dB. Moreover, the FBSS-rotation based QRD-LS beamformer introduces an offset of  $2^\circ$  in the nulls produced at  $\pm 60^\circ$ . It may also be noted that, in the case of FBSS-RMGSEF beamformer, the pattern nulls are sharp at  $\pm 60^\circ$  which is not the case for FBSS QRD-LS beamformer.

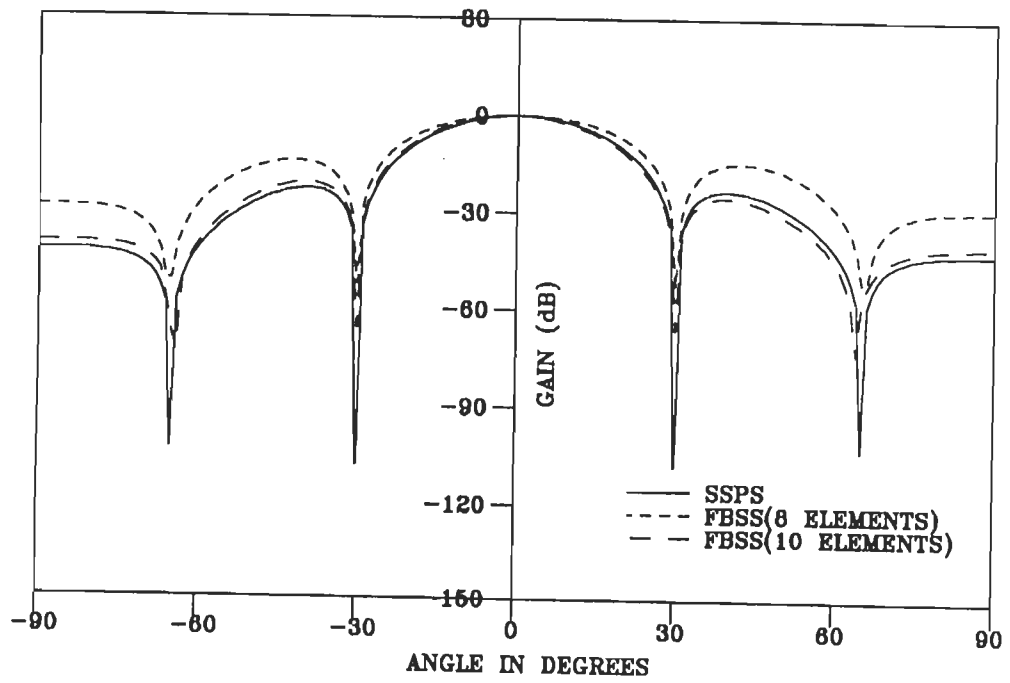
#### Example 5.4-2

In this example, we again consider a signal environment similar to the signal environment in example 5.4-1. The only difference is that the two interference arriving at  $\pm 60^\circ$  have been shifted to  $\pm 65^\circ$ .

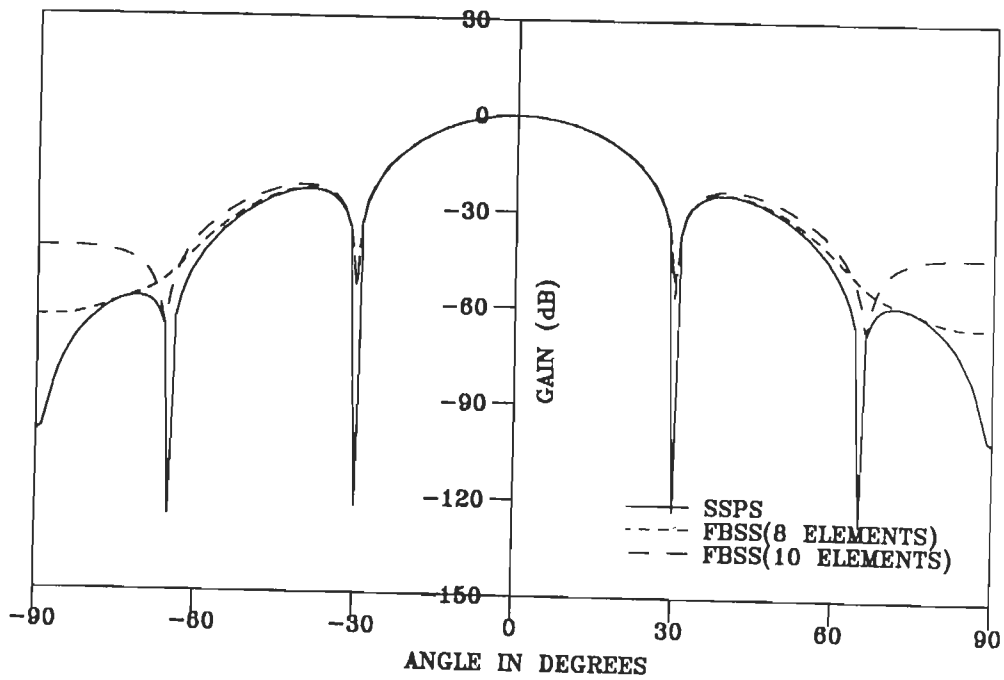
From the voltage patterns in Fig.5.7, it can be seen that both SSPS-RMGSEF and SSPS-QRD-LS beamformers work satisfactorily with minimum number of elements. The FBSS-RMGSEF beamformer also places explicit nulls in the direction of interferences. The FBSS-rotation based QRD-LS beamformer fails to place explicit null at  $\pm 65^\circ$ , though the beamformer gain is about -55dB and is equal to the null depth produced by FBSS-RMGSEF beamformer. When the number of elements in the array is increased to 10, FBSS beamformers place deeper nulls in the range of -65dB to -70dB in the direction of interferences. Also, the FBSS-QRD-LS beamformer places explicit nulls at  $\pm 65^\circ$ . It may be noted however, that even with minimum number of elements, the null depths in SSPS beamformers are much larger (-120dB) than those produced by the FBSS beamformers.

#### Example 5.4-3

In this example, the interferences have been assumed to



(a) RMGSEF BEAMFORMER



(b) QRD-LS BEAMFORMER

FIG.5.7 VOLTAGE PATTERNS OF SSPS AND FBSS ADAPTIVE BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ$ ,  $-30^\circ$ ,  $65^\circ$  &  $-65^\circ$ .

arrive at  $30^\circ$ ,  $-45^\circ$ ,  $60^\circ$  and  $-75^\circ$ . The remaining parameters are given in Table 5.6.

The voltage patterns are shown in Fig.5.8. It is found that, the SSPS beamformers perform satisfactorily by placing deep nulls ( $\geq -100\text{dB}$ ) in the direction of interferences. On the other hand, with minimum number of elements (8), the FBSS-RMGSEF array produces a bias of  $5^\circ$  in placing null at  $-75^\circ$  and the FBSS-rotation based QRD-LS array totally fails to steer a null at  $-75^\circ$ . When the overall array size is increased to 10 and 5 subarrays are formed in each direction, the FBSS places nulls at  $-75^\circ$  with a  $2^\circ$  shift in both RMGSEF and QRD-LS beamformers. Also the null depths in the direction of interferences arriving from  $30^\circ$ ,  $60^\circ$  and  $-45^\circ$  increases to  $-45\text{ dB}$  and  $-60\text{dB}$ . It may again be noted that SSPS beamformers exhibit much larger null depths compared to FBSS beamformers.

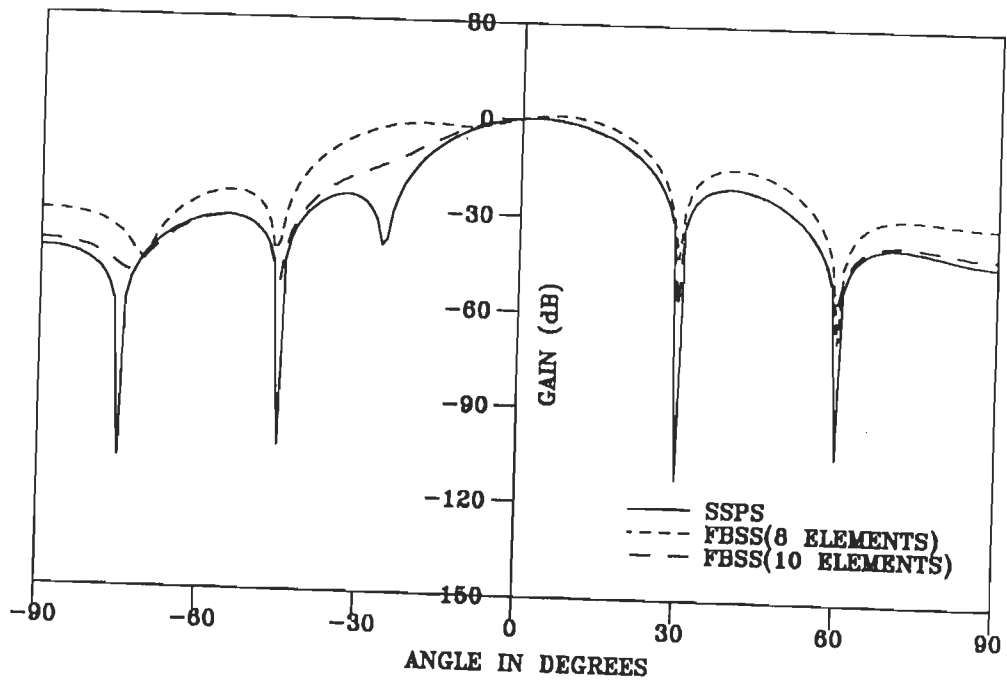
#### **Example 5.4-4**

In this example, two interferences have been assumed to arrive from directions very close to the desired signal ( $\pm 5^\circ$ ) and two from directions far away from the desired signal ( $\pm 75^\circ$ ). The remaining parameters are as given in Table 5.6.

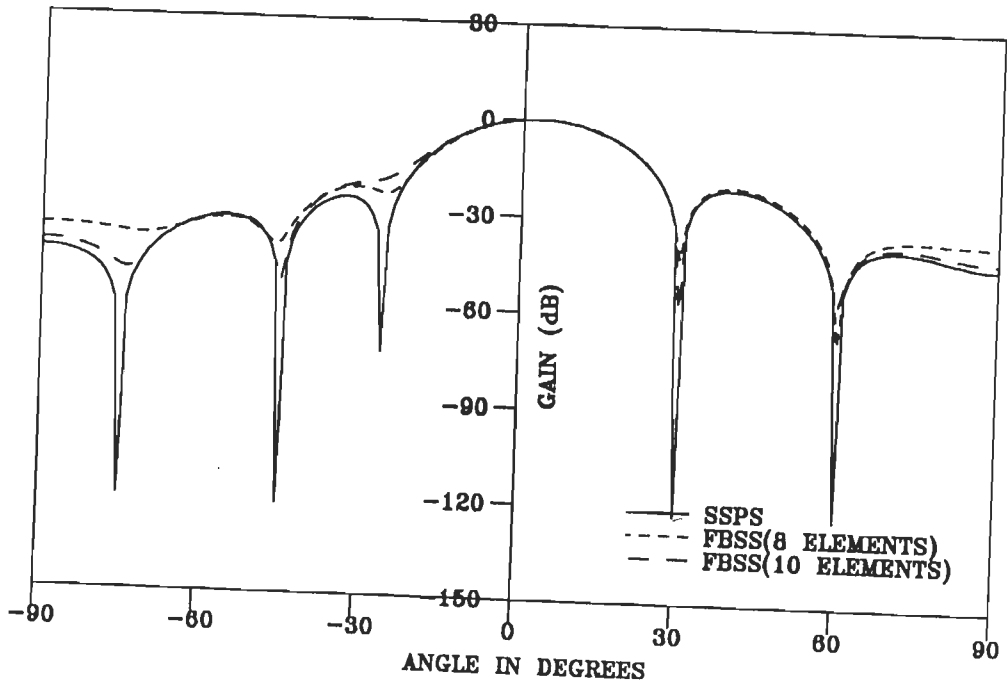
From the voltage patterns shown in Fig.5.9, it is found that both the FBSS and SSPS incorporated beamformers produce nearly identical patterns. The null depths are also more or less the same.

#### **Example 5.4-5**

In this example, the coherent interferences have been modelled to arrive from directions very close to each other but far

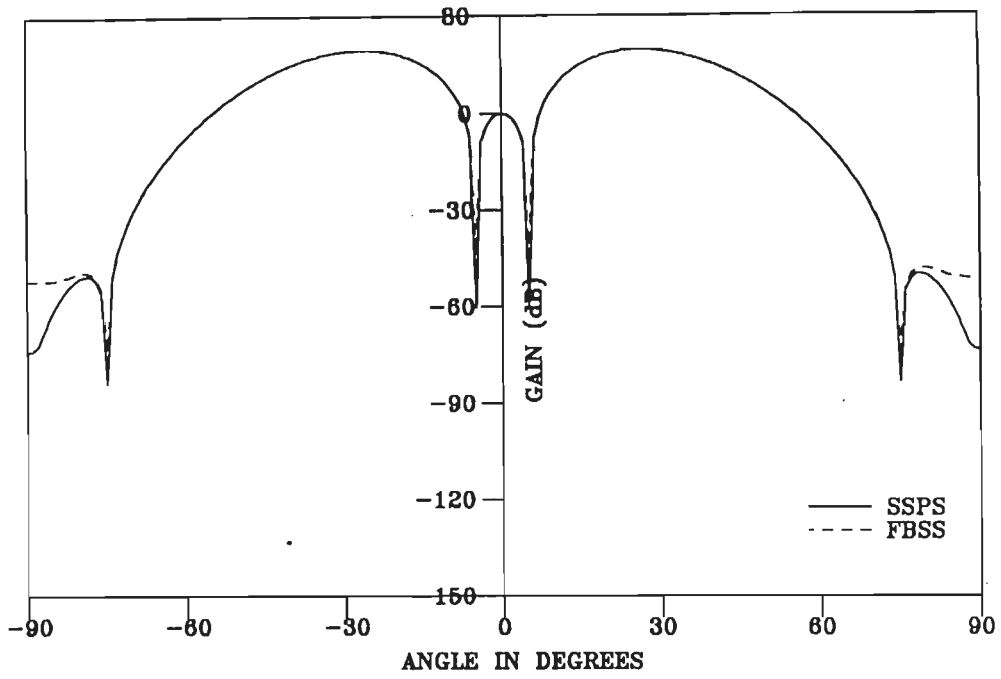


(a) RMGSEF BEAMFORMER

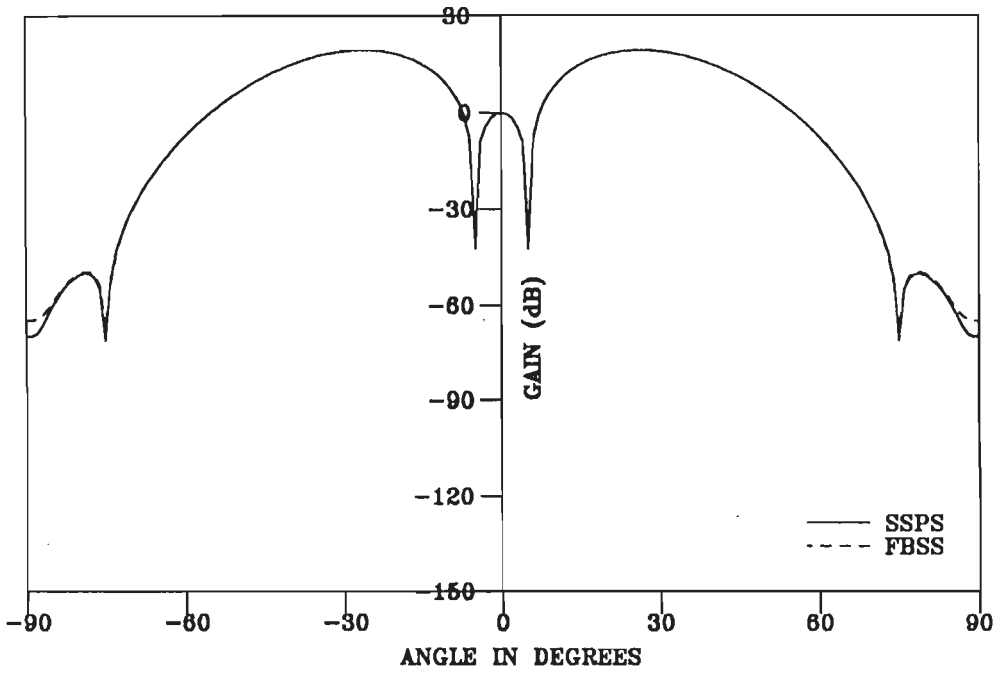


(b) QRD-LS BEAMFORMER

FIG.5.8 VOLTAGE PATTERNS OF SSPS AND FBSS ADAPTIVE BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ, -45^\circ, 60^\circ$  &  $-75^\circ$ .



(a) RMGSEF BEAMFORMER



(b) QRD-LS BEAMFORMER

FIG.5.9 VOLTAGE PATTERNS OF SSPS AND FBSS ADAPTIVE BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $5^\circ, -5^\circ, 75^\circ$  &  $-75^\circ$ .

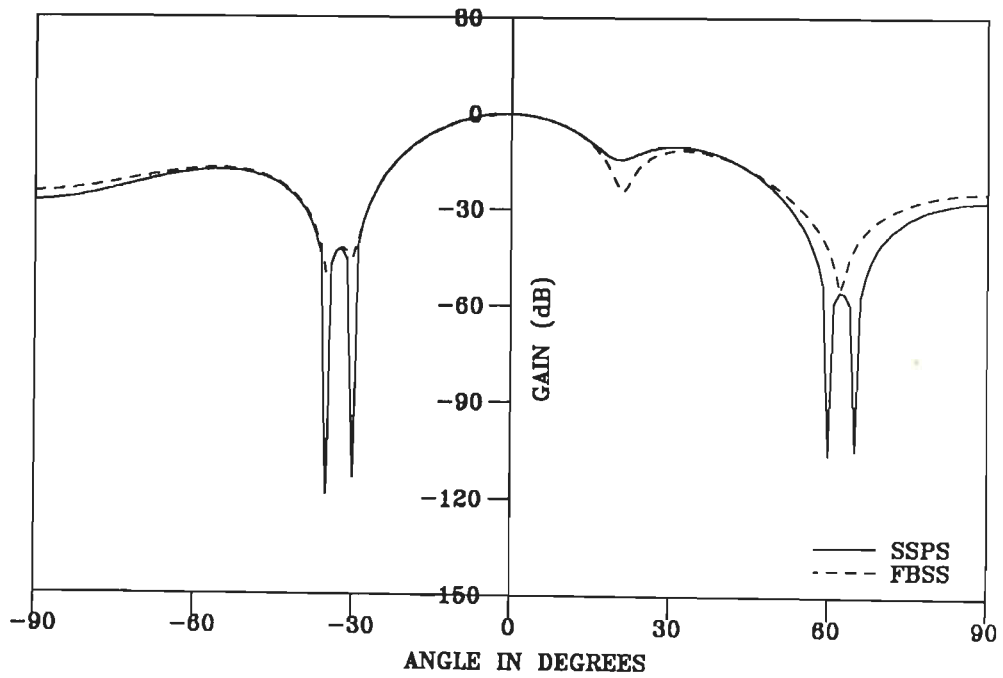
away from the desired signal. The arrival angles of the interferences are  $-30^\circ$ ,  $-35^\circ$ ,  $60^\circ$  and  $65^\circ$ .

The voltage patterns of the SSPS and FBSS RMGSEF adaptive beamformers are shown in Fig.5.10. It can be seen from the figure that the FBSS beamformer with minimum number of elements (8) fails to place nulls in the direction of interferences arriving from  $60^\circ$  and  $65^\circ$ . Also, the nulls placed at  $-30^\circ$  and  $-35^\circ$  are not sharp. It may, therefore, be concluded that minimum number of elements and subarrays formed (3 forward and 3 backward) are not sufficient to achieve the desired decorrelation. That is, FBSS fails to achieve the expected increase in effective aperture area for the present signal environment. On the other hand, the SSPS beamformer with minimum number of elements (10) places deep nulls ( $>-100\text{dB}$ ) in the direction of all the interferences.

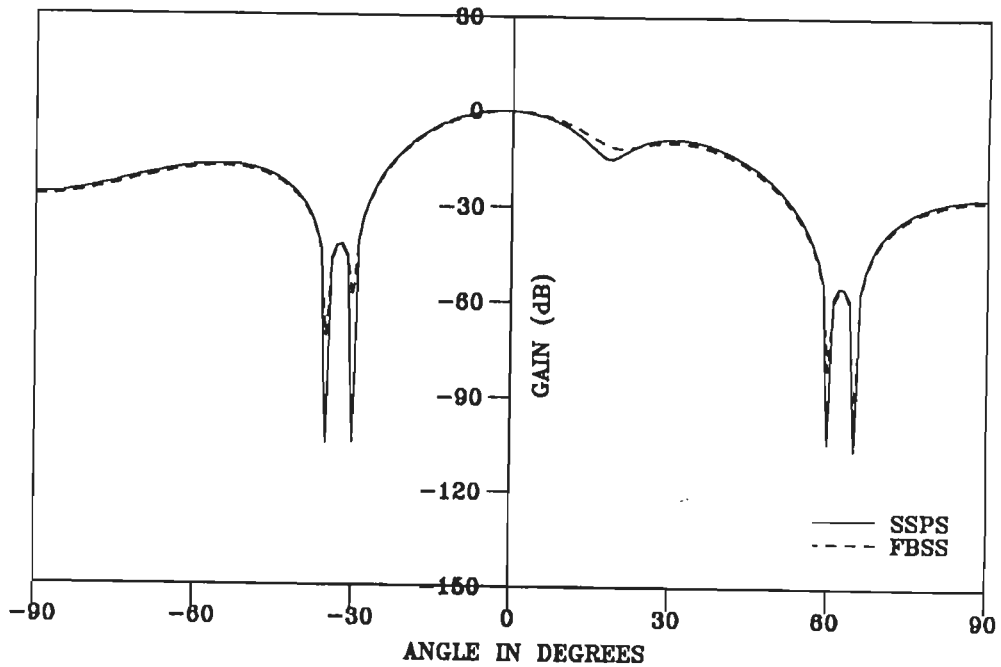
In order to obtain the desired decorrelation, the number of elements in the FBSS beamformer was then increased to 12 elements and, 7 forward and 7 backward subarrays were formed. Similarly, the number of elements in the case of SSPS beamformer was also increased to 12. It can be seen from Figure 5.10, that FBSS beamformer, now places sharp nulls in the directions of all the interferences. However, there is no significant changes in the voltage patterns of the SSPS beamformer. It may also be noted that, the nulls produced by FBSS beamformer are not as deep as those produced by the SSPS beamformer.

#### Example 5.4-6

In this example, the interferences have been assumed to arrive from the directions  $30^\circ$ ,  $40^\circ$ ,  $50^\circ$  and  $60^\circ$ . The remaining



(a) MINIMUM NUMBER OF ELEMENTS



(b) 12 ELEMENT ARRAY

FIG.5.10 VOLTAGE PATTERNS OF SSPS AND FBSS RMGSEF ADAPTIVE BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $-30^\circ, -35^\circ, 60^\circ$  &  $65^\circ$ .

parameters are given in Table 5.6.

From the voltage patterns in Fig.5.11, it is clear that the 8-element FBSS beamformer fails once again to perform satisfactorily. It does not place nulls in the direction of interferences arriving from  $40^\circ$  and  $50^\circ$  and introduces a bias in the nulls produced at  $30^\circ$  and  $60^\circ$ . On the other hand, the SSPS beamformer with minimum number of elements (10), once again exhibits superiority by placing deep nulls in the direction of all the interferences.

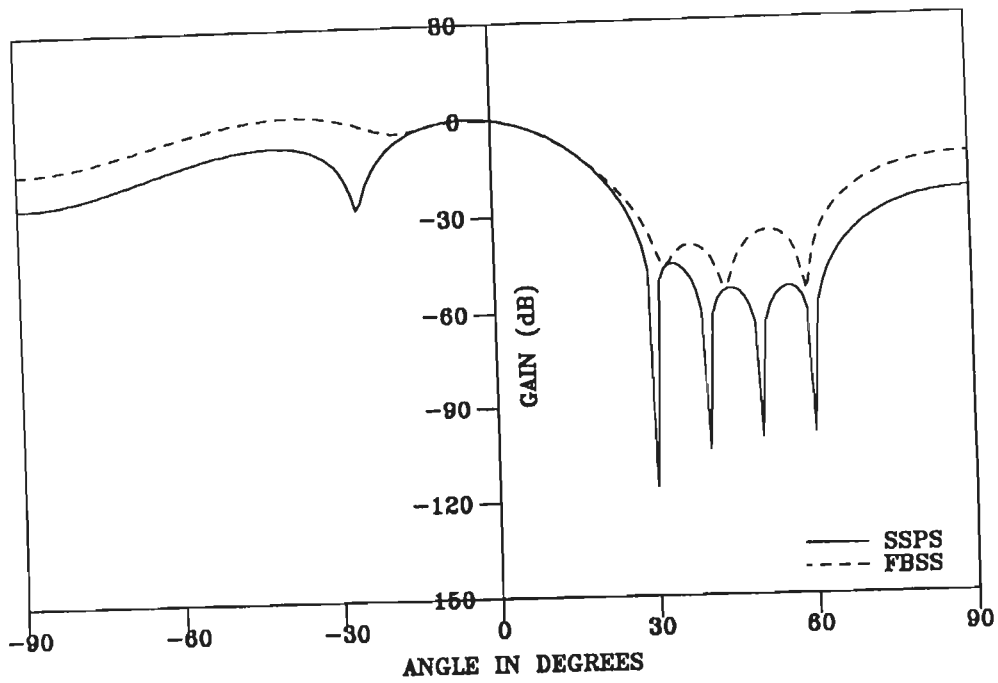
The number of elements was then increased gradually till the FBSS beamformer placed nulls in the desired directions. It was found that when the total number of elements in the array was raised to 13, that is a total of 8 forward and 8 backward subarrays were formed, the FBSS beamformer placed nulls in the direction of all the four interferences Fig.5.11(b). However, the nulls were much shallower compared to those placed by the SSPS beamformer with minimum number of elements (10). It may further be noted that when the number of elements is increased to 13 in the SSPS beamformer, a significant improvement in the null depths results in this case.

#### **Example 5.4-7**

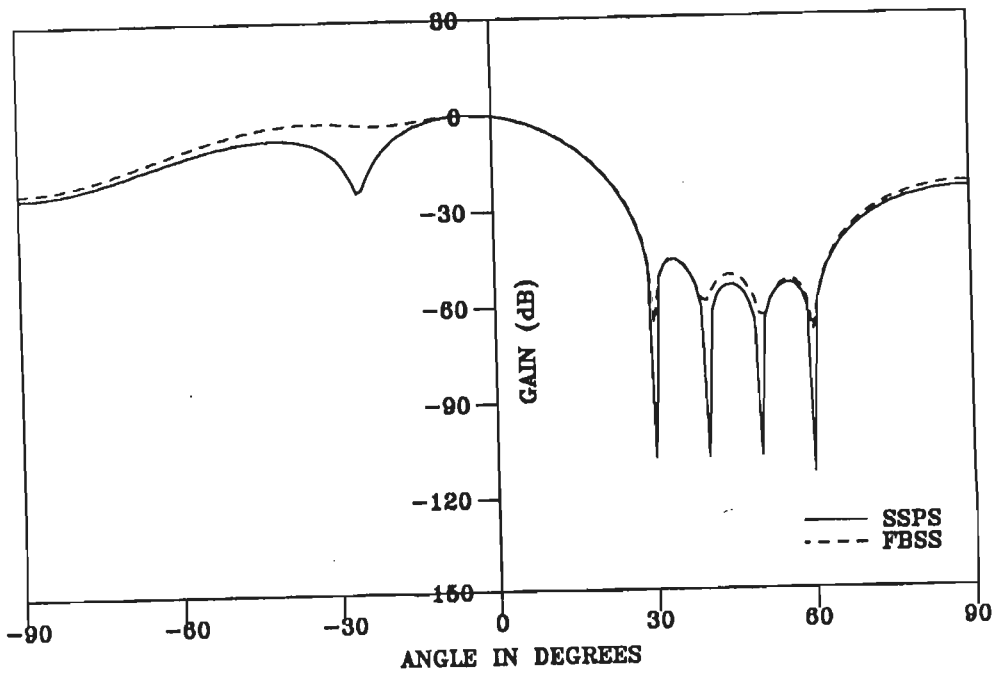
In this example, the  $10^\circ$  spacing between the interference arrival angles of the previous example has been reduced to  $5^\circ$ . That is the interference arrival angles are  $35^\circ$ ,  $40^\circ$ ,  $45^\circ$  and  $50^\circ$ .

As can be seen from the voltage patterns shown in Fig.5.12, both the SSPS and FBSS beamformers with minimum number of elements (10 and 8, respectively) fail to place the nulls in the direction of interferences.





(a) MINIMUM NUMBER OF ELEMENTS



(b) 13 ELEMENT ARRAY

FIG.5.11 VOLTAGE PATTERNS OF SSPS AND FBSS RMGSEF ADAPTIVE BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $30^\circ, 40^\circ, 50^\circ$  &  $60^\circ$ .

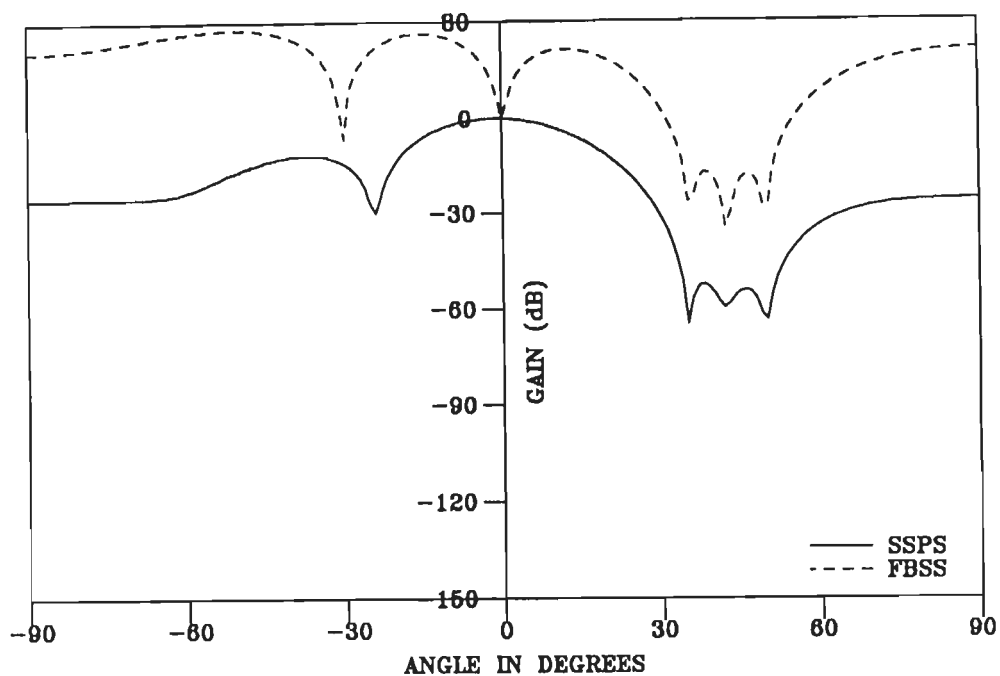
The number of elements was then gradually increased to 26 elements and 21 subarrays were formed in each direction. It is evident from Figure 5.12 that SSPS beamformer places deep nulls in the direction of interferences but the FBSS beamformer fails to place null in the direction of interference arriving from  $45^\circ$ .

It may be recalled that only minimum number of element(10) was sufficient in the case of SSPS, when the interferences arrived at the array with a  $10^\circ$  spacing. Therefore, it may be concluded that as the spacing between the interference arrival angle is decreased, larger number of subarrays need to be formed to achieve the desired decorrelation.

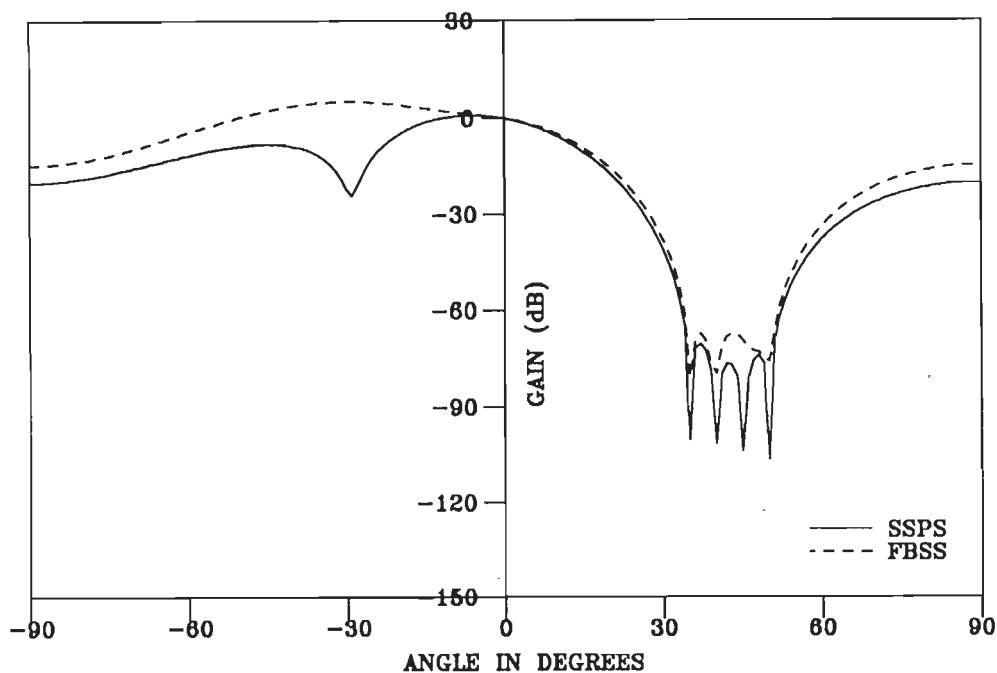
## 5.5 CONCLUSIONS

In this chapter, a detailed study has been carried out on the two spatial averaging schemes, namely, SSPS and FBSS, with regard to their ability to null coherent interferences, by incorporating them in adaptive beamformers based on RMGSEF and QRD-LS algorithms. Both the techniques have been investigated by considering typical signal environments in computer simulations.

It may be recalled that FBSS was initially proposed and studied in the context of DOA estimation and it was shown that it offers the advantage of an enhanced effective aperture area over SSPS [50,66]. However, the present investigations show that when minimum required number of elements are used, the FBSS beamformer fails to place nulls in the desired directions, except when the interferences are symmetrically located with respect to the broadside and not too near the endfire directions. When interferences are unsymmetrically



(a) MINIMUM NUMBER OF ELEMENTS



(b) 26 ELEMENT ARRAY

FIG.5.12 VOLTAGE PATTERNS OF SSPS AND FBSS RMGSEF ADAPTIVE BEAMFORMERS WITH INTERFERENCES ARRIVING AT  $35^{\circ}$ ,  $40^{\circ}$ ,  $45^{\circ}$  &  $50^{\circ}$ .

located on either side of the desired signal arrival direction, the FBSS array with minimum number of elements introduces a small bias ( $2^{\circ}$  -  $5^{\circ}$ ) in the placement of nulls. This situation can be, however, corrected by increasing the number of elements in the array. Further, as the angular separation between the interferences reduces, more and more number of elements are required in the FBSS array for satisfactory operation. Thus, FBSS fails to give the advantage of the increased aperture area in most situations.

On the other hand, the SSPS beamformers exhibit good performance with minimum number of elements in most signal environments. Only when the angular separation between the interferences becomes small, larger number of elements are needed in the array to decorrelate them. Moreover, the null depths obtained in an SSPS array are much greater than those obtained in an FBSS array with the same number of elements, resulting in much superior interference cancellation. Finally since both QRD-LS and RMGSEF algorithm based arrays give a comparable performance the latter is preferable in view of its lower computational complexity.

It may, therefore, be concluded that, for coherent interference cancellation an SSPS-RMGSEF beamformer is a better choice.

## CHAPTER - 6

### CONCLUSIONS

This dissertation addresses itself to a study of various adaptive signal processing algorithms with applications to adaptive beamforming. The study also includes spatial averaging methods which are used to preprocess the signals in a coherent signal environment. Although, there are many algorithms available in signal processing literature, we have, in this dissertation, restricted ourselves to the study of recursive exponentially windowed algorithms based on the method of least-squares.

As the adaptive array technology has been widely studied, an adequate description of some of the basic concepts in the field was found essential. Accordingly, in chapter-2, the optimum beamformer has been discussed in a narrowband signal environment, under the assumption that the desired signal, interferences and noise are all uncorrelated random processes. Following this, the well known LMS algorithm has been discussed and numerical examples based on computer simulations have been presented to demonstrate the ability of these beamformers to suppress interferences.

The study has then been extended to broadband arrays, by the addition of a tapped delay line network behind the array. The addition of constraints, by which a constant gain in a given direction can be fixed (Frost array), has been presented. The simulation procedure for the generation of broadband signal has also been described. Sample results of a simulated broadband signal and the voltage pattern of the

Frost array have then been presented.

The inability of the above mentioned beamforming techniques to null coherent interferences has then been discussed. A comparative study of the two spatial averaging techniques, namely, the SSPS and SCMM, which are used to preprocess the signals in a coherent signal and interference environment to overcome the signal cancellation phenomenon, has been carried out through computer simulations. The results show that, among the two techniques, SSPS exhibits a much superior performance. It has been found that SCMM introduces a bias in placing the nulls, which increases as the interferences are moved away from the broadside, and ultimately, leads to the array's failure in placing nulls in the direction of endfire interferences. Extensive computer simulations have revealed that only for certain combinations of arrival angles of the interferences, signal strengths and the number of elements in the array, the SCMM nulls the interferences satisfactorily. The method is not applicable to broadband arrays as the correlation matrix is nontoeplitz, even in noncoherent situations. Finally, the SCMM can not be implemented using beamformers based on different adaptive algorithms. In view of the above, it is concluded that SSPS is a better choice for coherent interference suppression.

We have next considered the application of the recursive least-square algorithms to the adaptive beamformers. The conventional RLS algorithm has been presented first and then, the Givens rotation based QRD-LS algorithm has been derived using the algebraic approach. Since a major problem with the latter is its expensive computational complexity, as an alternative, we have proposed the application of the

RMGS and RMGSEF algorithms to adaptive beamformers. These algorithms are computationally less expensive as compared to the Givens rotation based QRD-LS algorithm and, at the same time, possess all the advantages of the latter, viz, numerical stability and implementation using systolic structures. Comparison of the performance of adaptive beamformers based on these four algorithms, in both narrowband and broadband signal environments, has shown that the QRD-LS beamformer gives the best performance. A study of the residual power characteristics revealed that the beamformer exhibits the fastest convergence speed and numerically stable characteristics in both narrowband and broadband signal environments. Moreover, the QRD-LS beamformer places the deepest nulls, irrespective of the arrival angles of the interferences. The RMGS and RMGSEF beamformers have a slightly slower initial convergence speed but the residual power obtained in an RMGSEF beamformer is only marginally greater than that obtainable in the QRD-LS beamformer. Also the null depths in a RMGSEF beamformer are comparable to those in the QRD-LS beamformer. Of the four beamformers, the RLS beamformer was found to exhibit the poorest performance. It produces a large residual power when the number of interferences are large or when the interferences arrive from near endfire directions and fails to place nulls in the direction of endfire interferences. However, its performance improves in a broadband signal environment. A comparison of the computational complexity of these four adaptive beamformers shows that the RMGS algorithms, in particular the RMGSEF algorithm, has the least computational complexity. Therefore, it may be concluded that the

proposed RMGSEF beamformer is a good alternative to the Givens rotation based QRD-LS beamformer for most of the adaptive beamforming applications.

In chapter 4, the application of fast recursive least-square algorithms, viz, the LSL, QR-LSL and FTF algorithms for adaptive broadband beamformers has been investigated. Since multichannel formulations of these algorithms are necessary for beamforming applications, these have been derived using the algebraic approach. These algorithms exploit the inherent delays available in tapped delay line filters to arrive at beamforming techniques whose computational complexity is of the order  $P^3M$ . Of the three algorithms considered here, the MFTF algorithm has the least computational complexity. However, if the number of taps in the delay line are less than 5, the computational complexity of the MFTF beamformer is more than that of the RMGSEF beamformer. Through computer simulations, we have shown that the MLSL beamformer fails to achieve convergence in most of the situations. The MFTF and QR-MLSL beamformers exhibit relatively better performance characteristics but a comparison with exact RLS beamformers revealed that their performance is much inferior so far as residual power, signal tracking and interference suppression are concerned. We have, thus established that the fast RLS beamformers have neither the advantages of computational complexity, nor superior performance as compared to the exact RLS beamformers.

Finally, we have considered the adaptive implementation of the SSPS and FBSS using beamformers based on different algorithms. We have proposed schemes to implement SSPS and FBSS on weight oriented



algorithm, viz, RLS, LMS and FTF algorithm, as well as on residue oriented algorithms, that is the RMGS class of algorithms and the QRD-LS algorithms. Through computer simulations, it has been shown that when minimum required number of elements are used, the FBSS beamformer fails to place nulls in the desired directions, except when the interferences are symmetrically located with respect to broadside and are not too near the endfire directions. In other situations, such as, when the interferences are unsymmetrically located on either side of the desired signal arrival direction or when the angular separation between the interferences is small, the FBSS array fails to null the interferences when minimum required number of elements are used. In such cases, larger number of elements have to be used in order for the FBSS array to work. Thus, we have shown that the FBSS fails to give the advantage of increased aperture area in most situations.

On the other hand, SSPS beamformers have been found to exhibit good performance, with minimum number of elements, in most signal environments. Only when the angular separation between the interferences becomes small, larger number of elements are needed. Also, the null depths obtained in an SSPS array are much greater than those obtained in an FBSS array with the same number of elements, thereby resulting in much superior interference cancellation. Both the QRD-LS and RMGSEF algorithm based beamformers were found to give a comparable performance. Since, RMGSEF algorithm has a lower computational complexity, it may be concluded that, for coherent interference cancellation, an SSPS-RMGSEF beamformer is a better choice.

## 6.1 SUGGESTIONS FOR FURTHER WORK

Addition of constraints to the adaptive beamformers improves their performance, in that a constant gain in the look-direction can be maintained, or a null can be placed in the direction of interference, if the interference arrival angle is known in advance. Najm [71] has proposed constrained RLS and QRD-LS algorithms. It will be interesting to extend this approach to the RMGS and RMGSEF adaptive beamformers as these have exhibited good performance characteristics.

Although Givens rotation based QRD-LS algorithm studied in chapter 3 provides a numerically sound way of computing the recursive QR update, it has two drawbacks. The first is the square-root computation needed to compute the rotation factors, and the second is four operations per column needed to implement the rotation. Both of these drawbacks can be eliminated by the use of square-root free Givens rotations [34]. Therefore, a natural extension of this work would be to apply the square-root free Givens rotation based QRD-LS algorithm to adaptive beamforming and evaluate its performance.

In the broadband signal environment adaptive beamforming is a multichannel problem, which requires multichannel formulations of the fast RLS algorithms, viz, the LSL, FTF and the hybrid QR-LSL algorithms. It has been observed that the multichannel LSL algorithm based beamformer suffers from numerical problems, which may be due to the computation of inverses of error variance matrices repeatedly. The issue of matrix conditioning has not received much attention in the adaptive filtering community though it is well known in the context of Kalman filtering [58]. A direction of further work should be to study

the conditioning of this problem.

The multichannel FTF algorithm and its application to beamformers has been studied in chapter 4. This algorithm is a straight forward extension of the single channel form which requires matrix operations. Recently, to overcome the difficulties of MFTF algorithm, Slock and Kailath [58] have proposed the scalar implementation of the MFTF algorithm. It requires no matrix operations and can be implemented in modular architecture with a regular and highly parallel structure. An area of further study should be the evaluation of this algorithm in the context of adaptive beamforming.

## APPENDIX - A

### COMPLEX GRADIENT OPERATOR

Least-mean-square systems minimize the mean-square error between the output (normally a scalar) and the desired or the reference signal. When the signals are complex, a concise derivation of the results can be obtained using complex gradient operator without using the gradients of the real and imaginary parts separately.

If  $v$  is a complex scalar quantity, given by

$$v = x + jy$$

then, the following derivatives may be defined

$$\frac{\partial}{\partial v}(v) = \frac{\partial v}{\partial x} + j \frac{\partial v}{\partial y} = 0 \quad \dots (A-1)$$

$$\frac{\partial}{\partial v}(v^*) = \frac{\partial v}{\partial x} + j \frac{\partial v}{\partial y} = 2 \quad \dots (A-2)$$

$$\frac{\partial}{\partial v^*}(v) = \frac{\partial v}{\partial x} - j \frac{\partial v}{\partial y} = 2 \quad \dots (A-3)$$

and

$$\frac{\partial}{\partial v^*}(v^*) = \frac{\partial v}{\partial x} - j \frac{\partial v}{\partial y} = 0 \quad \dots (A-4)$$

let  $Z$  be a  $P$ -by-1 vector given by

$$\underline{Z} = [z_1, z_2, \dots, z_P]^T \quad \dots (A-5)$$

where

$$z_k = x_k + jy_k, \quad k = 1, 2, \dots, P. \quad \dots (A-6)$$

We also define the complex gradient operator  $\nabla_{\underline{z}}$  with respect to  $\underline{z}$  as

$$\nabla_{\underline{z}} = \left[ \frac{\partial}{\partial z_1}, \frac{\partial}{\partial z_2}, \dots, \frac{\partial}{\partial z_P} \right]^T \quad \dots (A-7)$$

where

$$\frac{\partial}{\partial z_k} = \frac{\partial}{\partial x_k} + j \frac{\partial}{\partial y_k}, \quad k = 1, 2, \dots, P \quad \dots (A-8)$$

Using the above definitions and eqns. (A-1) to (A-4) we can write

$$\nabla_{z_k}(z_k) = 0; \quad \nabla_{z_k}^*(z_k) = 2 \quad \dots (A-9)$$

$$\nabla_{z_k}^*(z_k^*) = 2; \quad \nabla_{z_k}(z_k^*) = 0 \quad \dots (A-10)$$

The complex gradient operator, as defined in eqn. (A-7) with respect to a complex P-by-1 vector  $\underline{z}$ , generates another complex P-by-1 vector from a scalar function of  $\underline{z}$ . The Scalar, in general, may be complex, although in physical applications it may be real. Typical scalar functions of  $\underline{z}$  arising in the adaptive beamforming problem are of the form  $\underline{A}^H \underline{z}, \underline{z}^H \underline{A}$  and  $\underline{z}^H \underline{R} \underline{z}$ , where  $\underline{A}$  is a P-by-1 vector and  $\underline{R}$  is a P-by-P Hermitian matrix. The gradient of these scalar functions with respect to  $\underline{z}$  are given by the following equations [22].

$$\nabla_{\underline{z}} \left[ \underline{A}^H \underline{z} \right] = 0 \quad \dots (A-11)$$

$$\nabla_{\underline{Z}} \left[ \underline{Z}^H \underline{A} \right] = 2\underline{A} \quad \dots (A-12)$$

$$\nabla_{\underline{Z}} \left[ \underline{Z}^H \underline{RZ} \right] = 2\underline{R} \underline{Z} \quad \dots (A-13)$$

The above equations may be verified by expanding the products and using the equations (A-9)-(A-10).

In an adaptive beamformer, the output at the time instant 'n' is given by

$$y(n) = \underline{W}^H(n) \underline{X}(n) \quad \dots (A-14)$$

If the desired response is  $d(n)$ , then the error is

$$e(n) = d(n) - \underline{W}^H(n) \underline{X}(n) \quad \dots (A-15)$$

The gradient of the  $E[e(n)e^*(n)]$  with respect to the complex weight vector  $\underline{W}(n)$  is obtained by using eqns. (A-11)-(A-13) as

$$\begin{aligned} \nabla_{\underline{W}(n)} \left[ |e(n)|^2 \right] &= \nabla_{\underline{W}(n)} \left[ \left[ d(n) - \underline{W}^H(n) \underline{X}(n) \right] \left[ d^*(n) - \underline{X}^H(n) \underline{W}(n) \right] \right] \\ &= \nabla_{\underline{W}(n)} \left[ |d(n)|^2 - \underline{\theta}^H(n) \underline{W}(n) - \underline{W}^H(n) \underline{\theta}(n) + \underline{W}^H(n) \underline{\Phi}(n) \underline{W}(n) \right] \\ &= -2\underline{\theta}(n) + 2\underline{\Phi}(n) \underline{W}(n) \quad \dots (A-16) \end{aligned}$$

We next define a P-by-M matrix  $\underline{W}$  as

$$W = \begin{bmatrix} w_{11} & w_{12} \cdots \cdots \cdots w_{1M} \\ w_{21} & w_{22} \cdots \cdots \cdots w_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ w_{P1} & w_{P2} \cdots \cdots \cdots w_{PM} \end{bmatrix} \quad \dots (A-17)$$

where

$$w_{i,k} = x_{i,k} + jy_{i,k}$$

The complex gradient operator  $\nabla_W$  with respect to the components of  $W$  is defined as

$$\nabla_W = \begin{bmatrix} \frac{\partial}{\partial w_{11}} & \frac{\partial}{\partial w_{12}} \cdots \cdots \cdots \frac{\partial}{\partial w_{1M}} \\ \frac{\partial}{\partial w_{21}} & \frac{\partial}{\partial w_{22}} \cdots \cdots \cdots \frac{\partial}{\partial w_{2M}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial}{\partial w_{P1}} & \frac{\partial}{\partial w_{P2}} \cdots \cdots \cdots \frac{\partial}{\partial w_{PM}} \end{bmatrix} \quad \dots (A-18)$$

This operator generates another complex P-by-M matrix from the trace of a matrix function of  $W$ . Typical matrix functions of  $W$  arising in adaptive arrays using multichannel fast RLS algorithms are of the form  $B^H W$ ,  $W^H B$  and  $W^H \Phi W$ , where  $B$  is a P-by-M matrix. The gradient of the trace of these matrix functions with respect to the components of  $W$  are given by the following equations

$$\nabla_W \left\{ \text{Trace of } \begin{bmatrix} B^H & W \end{bmatrix} \right\} = 0 \quad \dots (A-19)$$

$$\nabla_W \left\{ \text{Trace of } \begin{bmatrix} W^H & B \end{bmatrix} \right\} = 2 B \quad \dots (A-20)$$

$$\nabla_W \left\{ \text{Trace of } \begin{bmatrix} B^H & \Phi W \end{bmatrix} \right\} = 2 \Phi W. \quad \dots (A-21)$$



## APPENDIX - B

### DERIVATION OF TIME UPDATE EQUATION FOR $r_{ij}(n)$

In Table 3.4,  $r_{ij}(n)$  is given by

$$r_{ij}(n) = \left[ \underline{g}_j^{(i)}(n) \right]^H \underline{g}_i(n) \quad \dots (B-1)$$

To derive the time update equation for  $r_{ij}(n)$ , we use Gram-Schmidt orthogonalization theory [36] and express  $\underline{g}_j^{(i)}(n)$  and  $\underline{g}_i(n)$  in the above equation as

$$\begin{aligned} \underline{g}_j^{(i)}(n) &= \underline{X}_j(n) - A_{i-1}(n) \left[ A_{i-1}^H(n) A_{i-1}(n) \right]^{-1} A_{i-1}^H(n) \underline{X}_j(n) \\ &= \left[ I - A_{i-1}(n) \left[ A_{i-1}^H(n) A_{i-1}(n) \right]^{-1} A_{i-1}^H(n) \right] \underline{X}_j(n) \end{aligned} \quad \dots (B-2)$$

$$\underline{g}_i(n) = \left[ I - A_{i-1}(n) \left[ A_{i-1}^H(n) A_{i-1}(n) \right]^{-1} A_{i-1}^H(n) \right] \underline{X}_i(n) \quad \dots (B-3)$$

where  $A_{i-1}(n)$  is defined as

$$A_{i-1}(n) = \left[ \underline{X}_1(n), \underline{X}_2(n), \dots, \underline{X}_{i-1}(n) \right] \quad \dots (B-4)$$

Substituting (B-2) and (B-3) in (B-1),  $r_{ij}(n)$  can be rewritten as

$$\begin{aligned} r_{ij}(n) &= \underline{X}_j^H(n) \left[ I - A_{i-1}(n) \left[ R_{i-1}(n) \right]^{-1} A_{i-1}^H(n) \right]^H \\ &\quad \left[ I - A_{i-1}(n) \left[ R_{i-1}(n) \right]^{-1} A_{i-1}^H(n) \right] \underline{X}_i(n) \end{aligned} \quad \dots (B-5)$$

where

$$R_{i-1}^{-1}(n) = \left[ A_{i-1}^H(n) A_{i-1}(n) \right]^{-1} \quad \dots (B-6)$$

Eqn. (B-5) can be written as [20]

$$r_{ij}(n) = \underline{x}_j^H(n) \left[ I - A_{i-1}(n) R_{i-1}^{-1}(n) A_{i-1}^H(n) \right] \underline{x}_i(n) \quad \dots (B-7)$$

Next, we define a matrix  $A_{i-1}(n-1)$  such that

$$A_{i-1}(n) = \begin{bmatrix} \lambda^{1/2} A_{i-1}(n-1) \\ \underline{x}_{i-1}^H(n) \end{bmatrix} \quad \dots (B-8)$$

where

$$\underline{x}_{i-1}(n) = \left[ x_1(n), x_2(n), \dots, x_{i-1}(n) \right]^T \quad \dots (B-9)$$

Using (B-2) and (B-3), the last elements in the vectors  $\underline{q}_j^{(i)}(n)$  and  $\underline{q}_i(n)$ , denoted respectively by  $q_j^{(i)}(n,n)$  and  $q_i(n,n)$  can be expressed as

$$\begin{aligned} q_j^{(i)}(n,n) &= \left[ -\underline{x}_{i-1}^H(n) R_{i-1}^{-1}(n) \lambda^{1/2} A_{i-1}^H(n), \right. \\ &\quad \left. 1 - \underline{x}_{i-1}^H(n) R_{i-1}^{-1}(n) \underline{x}_{i-1}(n) \right] \underline{x}_j(n) \\ &= \left[ -\lambda^{1/2} \underline{x}_{i-1}^H(n) R_{i-1}^{-1}(n) A_{i-1}^H(n), \alpha_i(n) \right] \underline{x}_j(n) \quad \dots (B-10) \end{aligned}$$

$$q_i(n,n) = \left[ -\lambda^{1/2} \underline{x}_{i-1}^H(n) R_{i-1}^{-1}(n) A_{i-1}^H(n), \alpha_i(n) \right] \underline{x}_i(n) \quad \dots (B-11)$$

where

$$\alpha_i(n) = 1 - \underline{X}_{i-1}^H(n) R_{i-1}^{-1}(n) \underline{X}_{i-1}(n) \quad \dots (B-12)$$

Therefore,  $\alpha_1(n) = 1$

From B-8, we can show that

$$R_{i-1}(n) = \lambda R_{i-1}(n-1) + \underline{X}_{i-1}(n) \underline{X}_{i-1}^H(n) \quad \dots (B-13)$$

Using matrix inversion lemma, the inverse of  $R_{i-1}(n-1)$  can be written as

$$\frac{1}{\lambda} R_{i-1}^{-1}(n-1) = R_{i-1}^{-1}(n) + \frac{R_{i-1}^{-1}(n) \underline{X}_{i-1}(n) \underline{X}_{i-1}^H(n) R_{i-1}^{-1}(n)}{\alpha_i(n)} \quad \dots (B-14)$$

Now, we rewrite (B-7) in the following form

$$r_{ij}(n) = \underline{X}_j^H(n) \left[ I - \begin{bmatrix} \lambda^{1/2} A_{i-1}(n-1) \\ \underline{X}_{i-1}^H(n) \end{bmatrix} R_{i-1}^{-1}(n) \begin{bmatrix} \lambda^{1/2} A_{i-1}^H(n-1), \underline{X}_{i-1}^H(n) \end{bmatrix} \right] \underline{X}_i(n) \quad \dots (B-15)$$

on simplification, we get

$$r_{ij}(n) = \underline{X}_j^H(n) \begin{bmatrix} I - \lambda A_{i-1}(n-1) R_{i-1}^{-1}(n) A_{i-1}^H(n-1) & -\lambda^{1/2} A_{i-1}(n-1) R_{i-1}^{-1}(n) \underline{X}_{i-1}(n) \\ -\lambda^{1/2} \underline{X}_{i-1}^H(n) R_{i-1}^{-1}(n) A_{i-1}(n-1) & \alpha_i(n) \end{bmatrix} \underline{X}_i(n) \quad \dots (B-16)$$

Using eqn. (3.56) and (B-14) in (B-16), we obtain

$$\begin{aligned}
r_{ij}(n) = & \left[ \lambda^{1/2} \underline{X}_j^H(n-1), x_j(n) \right] \begin{bmatrix} I - A_{i-1}(n-1)R_{i-1}^{-1}(n-1)A_{i-1}^H(n-1) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda \underline{X}_i(n-1) \\ x_i(n) \end{bmatrix} \\
& + \frac{\underline{X}_j^H(n)}{\alpha_i(n)} \begin{bmatrix} -\lambda^{1/2} A_{i-1}(n-1)R_{i-1}^{-1}(n) \underline{X}_{i-1}(n) \\ \alpha_i(n) \end{bmatrix} \\
& \left[ -\lambda^{1/2} \underline{X}_{i-1}^H(n)R_{i-1}^{-1}(n)A_{i-1}(n-1), \alpha_i(n) \right] \underline{X}_i(n)
\end{aligned}$$

... (B-17)

A comparison of the above equation with (B-11) and (B-12) gives the desired update formula as

$$r_{ij}(n) = \lambda r_{ij}(n-1) + \frac{1}{\alpha_i(n)} \left[ q_j^{(i)}(n,n) \right]^* q_i(n,n) \quad \dots (B-18)$$

## APPENDIX - C

### ORDER UPDATE EQUATION FOR $\alpha_1(n)$

In Appendix-B, while deriving the time update equation for  $r_{ij}(n)$ , a quantity  $\alpha_1(n)$  has been defined. To complete the derivation of the complex RMGS algorithm, we derive here an order update equation for  $\alpha_1(n)$ .

$A_{i-1}(n)$  is given by eqn.(B-4) as

$$A_{i-1}(n) = \left[ \underline{X}_1(n), \underline{X}_2(n), \dots, \underline{X}_{i-1}(n) \right] \quad \dots (C-1)$$

Similarly we can write

$$A_i(n) = \left[ \underline{X}_1(n), \underline{X}_2(n), \dots, \underline{X}_i(n) \right] \quad \dots (C-2)$$

which can be partitioned as

$$A_i(n) = \left[ A_{i-1}(n), \underline{X}_i(n) \right] \quad \dots (C-3)$$

Substituting (C-3) in (B-6), we obtain

$$R_i(n) = \begin{bmatrix} R_{i-1}(n) & A_{i-1}^H(n) \underline{X}_i(n) \\ \underline{X}_i^H(n) A_{i-1}(n) & \underline{X}_i^H(n) \underline{X}_i(n) \end{bmatrix} \quad \dots (C-4)$$

Applying partitioned matrix inversion lemma [48], the inverse of  $R_i(n)$  can be written as

$$R_i^{-1}(n) = \begin{bmatrix} R_{i-1}^{-1}(n) & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{r_{ii}(n)} \begin{bmatrix} R_{i-1}^{-1}(n) A_{i-1}^H(n-1) \underline{x}_i(n) \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} \underline{x}_i^H(n) A_{i-1}(n) R_{i-1}^{-1}(n), -1 \end{bmatrix}$$

... (C-5)

where

$$r_{ii}(n) = \underline{x}_i^H(n) \underline{x}_i(n) - \underline{x}_i^H(n) A_{i-1}(n) R_{i-1}^{-1}(n) \underline{x}_i(n)$$

... (C-6)

Substitution of (C-5) into (B-12) gives

$$\alpha_{i+1}(n) = 1 - \underline{x}_{i-1}^H(n) \begin{bmatrix} R_{i-1}^{-1}(n) & 0 \\ 0 & 0 \end{bmatrix} \underline{x}_{i-1}(n)$$

$$- \underline{x}_{i-1}^H(n) \cdot \frac{1}{r_{ii}(n)} \begin{bmatrix} R_{i-1}^{-1}(n) A_{i-1}^H(n-1) \underline{x}_i(n) \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} \underline{x}_i^H(n) A_{i-1}(n) R_{i-1}^{-1}(n), -1 \end{bmatrix} \underline{x}_{i-1}(n)$$

... (C-7)

using (B-12) and (B-11), eqn. (C-7) can be written as

$$\alpha_{i+1} = \alpha_i(n) - \frac{1}{r_{ii}(n)} |q_i(n, n)|^2$$

... (C-8)

which is the required time update equation for  $\alpha_i(n)$ .

## APPENDIX-D

### RELATIONS BETWEEN MLSL AND QR-MLSL ALGORITHMS

In order to derive eqns.(4.58) and (4.59), the matrices  $A_1, A_2, B_1$  and  $B_2$  are defined with the help of eqn.(4.53) in the following manner.

$$A_1 = \begin{bmatrix} \sqrt{\lambda} R_m^f(n-1) \\ [\tilde{e}_m^f(n)]^H \end{bmatrix}, \quad A_2 = \begin{bmatrix} \sqrt{\lambda} \Gamma_m^f(n-1) & \underline{0} \\ [\tilde{e}_m^b(n-1)]^H & \tilde{\alpha}_m(n-1) \end{bmatrix} \quad \dots (D-1)$$

and

$$B_1 = \begin{bmatrix} R_m^f(n) \\ \underline{0}^H \end{bmatrix}, \quad B_2 = \begin{bmatrix} \Gamma_m^f(n) & \beta_m^f(n) \\ [\tilde{e}_{m+1}^b(n)]^H & \tilde{\alpha}_{m+1}(n) \end{bmatrix} \quad \dots (D-2)$$

Applying the matrix identity in eqn.(4.55), we get

$$\left[ R_m^f(n) \right]^H \Gamma_m^f(n) = \lambda \left[ R_m^f(n-1) \right]^H \Gamma_m^f(n-1) + \tilde{e}_m^f(n) \left[ \tilde{e}_m^b(n-1) \right]^H \quad \dots (D-3)$$

$$\left[ R_m^f(n) \right]^H \beta_m^f(n) = \tilde{e}_m^f(n) \tilde{\alpha}_m(n-1) \quad \dots (D-4)$$

We next, define  $A_1, A_2, B_1$  and  $B_2$  using eqn. (4.53) as

$$A_1 = \begin{bmatrix} \sqrt{\lambda} R_m^f(n-1) & \sqrt{\lambda} \Gamma_m^f(n-1) \\ [\tilde{e}_m^f(n)]^H & [\tilde{e}_m^b(n-1)]^H \end{bmatrix}, \quad A_2 = \begin{bmatrix} \underline{0} \\ \tilde{\alpha}_m(n-1) \end{bmatrix} \quad \dots (D-5)$$

$$B_1 = \begin{bmatrix} R_m^f(n) & \underline{r}_m^f(n) \\ \underline{0}^H & [\tilde{e}_{m+1}^b(n)]^H \end{bmatrix}, \quad B_2 = \begin{bmatrix} \underline{\beta}_m^f(n) \\ \tilde{\alpha}_{m+1}^b(n) \end{bmatrix} \quad \dots (D-6)$$

Applying the matrix identity in eqn. (4.55), we get

$$[R_m^f(n)]^H \underline{\beta}_m^f(n) = \tilde{e}_m^f(n) \tilde{\alpha}_m^{(n-1)} \quad \dots (D-7)$$

$$[\underline{r}_m^f(n)]^H \underline{\beta}_m^f(n) + \tilde{e}_{m+1}^b(n) \tilde{\alpha}_{m+1}^b(n) = \tilde{e}_m^b(n-1) \tilde{\alpha}_m^{(n-1)} \quad \dots (D-8)$$

By defining,

$$A_1 = \begin{bmatrix} \underline{0} \\ \tilde{\alpha}_m^{(n-1)} \end{bmatrix}, \quad A_2 = \begin{bmatrix} \underline{0} \\ \tilde{\alpha}_m^{(n-1)} \end{bmatrix} \quad \dots (D-9)$$

and

$$B_1 = \begin{bmatrix} \underline{\beta}_m^f(n) \\ \tilde{\alpha}_{m+1}^b(n) \end{bmatrix}, \quad B_2 = \begin{bmatrix} \underline{\beta}_m^f(n) \\ \tilde{\alpha}_{m+1}^b(n) \end{bmatrix} \quad \dots (D-10)$$

Using the identity, we get

$$[\underline{\beta}_m^f(n)]^H \underline{\beta}_m^f(n) + \tilde{\alpha}_{m+1}^b(n) = \tilde{\alpha}_m^2(n-1) \quad \dots (D-11)$$

We next consider eqn. (4.54).

If,

$$A_1 = \begin{bmatrix} \sqrt{\lambda} R_m^b(n-2) \\ [\tilde{e}_m^b(n-1)]^H \end{bmatrix}, \quad A_2 = \begin{bmatrix} \sqrt{\lambda} R_m^b(n-2) \\ [\tilde{e}_m^b(n-1)]^H \end{bmatrix} \quad \dots (D-12)$$



and

$$B_1 = \begin{bmatrix} R_m^{b(n-1)} \\ \underline{0}^H \end{bmatrix}, \quad B_2 = \begin{bmatrix} R_m^{b(n-1)} \\ \underline{0}^H \end{bmatrix} \quad \dots (D-13)$$

using the identity once again, we have

$$\left[ R_m^{b(n-1)} \right]^H \left[ R_m^{b(n-1)} \right] = \lambda \left[ R_m^{b(n-2)} \right]^H \left[ R_m^{b(n-2)} \right] + \tilde{e}_m^{b(n-1)} \left[ \tilde{e}_m^{b(n-1)} \right]^H \quad \dots (D-14)$$

Substituting for  $\tilde{e}_m^{b(n-1)}$  and  $R_m^{b(n-1)}$ , the above equation can be simplified as

$$r_m^{b(n-1)} = \lambda r_m^{b(n-2)} + \frac{e_m^{b(n-1)} \left[ e_m^{b(n-1)} \right]^H}{\alpha_m(n-1)} \quad \dots (D-15)$$

By defining,

$$A_1 = \begin{bmatrix} \sqrt{\lambda} R_m^{b(n-2)} & \sqrt{\lambda} \Gamma_m^{b(n-1)} \\ \left[ \tilde{e}_m^{b(n-1)} \right]^H & \left[ \tilde{e}_m^f(n) \right]^H \end{bmatrix}, \quad A_2 = \begin{bmatrix} \sqrt{\lambda} \Gamma_m^{e(n-2)} \\ \tilde{e}_m^*(n-1) \end{bmatrix} \quad \dots (D-16)$$

and

$$B_1 = \begin{bmatrix} R_m^{b(n-1)} & \Gamma_m^{b(n)} \\ \underline{0}^H & \tilde{e}_{m+1}^f(n) \end{bmatrix}, \quad B_2 = \begin{bmatrix} \Gamma_m^{e(n-1)} \\ \tilde{e}_{m+1}^*(n-1) \end{bmatrix} \quad \dots (D-17)$$

and applying the identity

$$\left[ R_m^{b(n-1)} \right]^H \Gamma_m^{e(n-1)} = \lambda \left[ R_m^{b(n-2)} \right]^H \Gamma_m^{e(n-2)} + \tilde{e}_m^{b(n-1)} \tilde{e}_m^*(n-1) \quad \dots (D-18)$$

which can be simplified as

$$\underline{d}_m(n-1) = \lambda \underline{d}_m(n-2) + \underline{e}_m^b(n-1) \underline{e}_m^*(n-1) / \alpha_m(n-1) \quad \dots (D-19)$$

Using eqn.(4.54), we next chose

$$A_1 = \begin{bmatrix} \sqrt{\lambda} R_m^b(n-2) & \sqrt{\lambda} \underline{r}_m^b(n-1) \\ [\underline{\tilde{e}}_m^b(n-1)]^H & [\underline{\tilde{e}}_m^f(n)]^H \end{bmatrix}, \quad A_2 = \begin{bmatrix} \underline{0} \\ \underline{\tilde{\alpha}}_m(n-1) \end{bmatrix} \quad \dots (D-20)$$

and

$$B_1 = \begin{bmatrix} R_m^b(n-1) & \underline{r}_m^b(n) \\ \underline{0}^H & [\underline{\tilde{e}}_{m+1}^f(n)]^H \end{bmatrix}, \quad B_2 = \begin{bmatrix} \underline{\beta}_m^b(n-1) \\ \underline{\tilde{\alpha}}_{m+1}(n-1) \end{bmatrix}. \quad \dots (D-21)$$

We get

$$\left[ R_m^b(n-1) \right]^H \underline{\beta}_m^b(n-1) = \underline{\tilde{e}}_m^b(n-1) \underline{\tilde{\alpha}}_m(n-1) \quad \dots (D-22)$$

and

$$\left[ \underline{r}_m^b(n) \right]^H \underline{\beta}_m^b(n-1) + \underline{\tilde{e}}_{m+1}^f(n) \underline{\tilde{\alpha}}_{m+1}(n-1) = \underline{\tilde{e}}_m^f(n) \underline{\tilde{\alpha}}_m(n-1) \quad \dots (D-23)$$

Replacing the internal variables by the definitions in eqn.(4.51), the above equations can be simplified as

$$\underline{e}_m^b(n-1) = \underline{e}_m^b(n-1), \quad \dots (D-24)$$

$$\underline{e}_{m+1}^f(n) = \underline{e}_m^f(n) - K_m^H(n) \left[ r_m^b(n-1) \right]^{-1} \underline{e}_m^b(n-1). \quad \dots (D-25)$$

Finally, we assign

$$A_1 = \begin{bmatrix} \sqrt{\lambda} \underline{r}_m^e(n-2) & \sqrt{\lambda} R_m^b(n-2) \\ \tilde{e}_m^*(n-1) & [\tilde{e}_m^b(n-1)]^H \end{bmatrix}, \quad A_2 = \begin{bmatrix} \underline{0} \\ \tilde{\alpha}_m(n-1) \end{bmatrix} \quad \dots (D-26)$$

and

$$B_1 = \begin{bmatrix} \underline{r}_m^e(n-1) & R_m^b(n-1) \\ \tilde{e}_{m+1}^*(n-1) & \underline{0}^H \end{bmatrix}, \quad B_2 = \begin{bmatrix} \underline{\beta}_m^b(n-1) \\ \tilde{\alpha}_{m+1}(n-1) \end{bmatrix} \quad \dots (D-27)$$

Once again applying the matrix identity, we get

$$[\underline{r}_m^e(n-1)]^H \underline{\beta}_m^b(n-1) + \tilde{e}_{m+1}(n-1) \tilde{\alpha}_{m+1}(n-1) = \tilde{e}_m(n-1) \tilde{\alpha}_m(n-1) \quad \dots (D-28)$$

Which can be simplified as

$$e_{m+1}(n-1) = e_m(n-1) - \underline{d}_m^H(n-1) \left[ \underline{r}_m^b(n-1) \right]^{-1} \underline{e}_m^b(n-1) \quad \dots (D-29)$$



APPENDIX - E

DERIVATION OF EQUATION FOR  $\underline{C}_{P(M+1)}^{(n+1)}$

The correlation matrix  $\Phi_{P(M+1)}^{(n+1)}$  in eqn.(4.72) can be updated as

$$\Phi_{P(M+1)}^{(n+1)} = \lambda \Phi_{P(M+1)}^{(n)} + X_{P(M+1)}^{(n+1)} \left[ X_{P(M+1)}^{(n+1)} \right]^H \quad \dots (E-1)$$

From eqn.(4.21b), the inverse of  $\Phi_{P(M+1)}^{-1}(n)$  is written as

$$\Phi_{P(M+1)}^{-1}(n) = \begin{bmatrix} 0 & 0 \\ 0 & \Phi_{PM}^{-1}(n-1) \end{bmatrix} + \begin{bmatrix} I \\ -A_{PM}(n) \end{bmatrix} \left[ r_M^f(n) \right]^{-1} \begin{bmatrix} I & -A_{PM}^H(n) \end{bmatrix} \quad \dots (E-2)$$

Post multiplying (E-2) by  $\frac{1}{\lambda} X_{P(M+1)}^{(n+1)}$ ,

$$\begin{aligned} \frac{1}{\lambda} \left[ \Phi_{P(M+1)}^{(n)} \right]^{-1} X_{P(M+1)}^{(n+1)} &= \frac{1}{\lambda} \begin{bmatrix} 0 & 0 \\ 0 & [\Phi_{PM}(n-1)]^{-1} \end{bmatrix} X_{P(M+1)}^{(n+1)} \\ &+ \frac{1}{\lambda} \begin{bmatrix} I \\ -A_{PM}(n) \end{bmatrix} \left[ r_M^f(n) \right]^{-1} \begin{bmatrix} I & -A_{PM}^H(n) \end{bmatrix} X_{P(M+1)}^{(n+1)} \end{aligned}$$

$$\underline{C}_{P(M+1)}^{(n+1)} = \begin{bmatrix} 0 \\ \underline{C}_{PM}(n) \end{bmatrix} + \frac{1}{\lambda} \begin{bmatrix} I \\ -A_{PM}(n) \end{bmatrix} \left[ r_M^f(n) \right]^{-1} \underline{\eta}_M^f(n+1) \quad \dots (E-3)$$

or

$$\underline{C}_{P(M+1)}^{(n+1)} = \begin{bmatrix} \frac{1}{\lambda} \left[ r_M^f(n) \right]^{-1} & \underline{\eta}_M^f(n+1) \\ \underline{C}_{PM}(n) - \frac{1}{\lambda} A_{PM}(n) \left[ r_M^f(n) \right]^{-1} & \underline{\eta}_M^f(n+1) \end{bmatrix} \quad \dots (E-4)$$

Similarly, it can also be written in terms of  $C_{-PM}^{(n+1)}$  as

$$C_{-P(M+1)}^{(n+1)} = \begin{bmatrix} C_{-PM}^{(n+1)} - \frac{1}{\lambda} B_{PM}^{(n)} \left[ r_M^{b(n)} \right]^{-1} & \underline{\eta}_M^{b(n)} \\ \frac{1}{\lambda} \left[ r_M^{b(n)} \right]^{-1} & \underline{\eta}_M^{b(n)} \end{bmatrix} \quad \dots (E-5)$$

## REFERENCES

1. Altman, F.J. and Sichak, W., 'A simplified diversity communication system for behind the horizon links', IRE Trans. Comm. Sys., Vol. CS-4, PP. 50-55, March 1956.
2. Applebaum, S.P., 'Adaptive Arrays', IEEE Trans. Antennas and propagat., Vol. AP-24, PP. 585-594, Sept. 1976.
3. Baird, C.A., 'Recursive processing for adaptive arrays', proceedings of the adaptive Antenna workshop, Naval research Laboratory, Washington, March 11-13, 1974.
4. Brennan, L.E., Mallet, J.D. and Reed, I.S., 'Adaptive arrays in airborne MTI Radar', IEEE Trans. Antennas and propagat., Vol. AP-24, PP. 607-613, Sept. 1976.
5. Cioffi, J.M., and Kailath, T., 'A classification of fast fixed order RLS algorithms', Presented at ASSP digital signal processing workshop, Chatham, Mass, Oct. 1984, Paper 1.1.1.
6. Cioffi, J.M. and Kailath, T., 'Fast, recursive least-square transversal filter for adaptive filtering', IEEE Trans. Acoust. Speech and signal process., Vol. ASSP-32, PP. 304-337, April 1984.
7. Cioffi, J.M., 'The fast adaptive ROTOR'S algorithm', IEEE Trans. Acoust. Speech. Signal Process., Vol. ASSP-38, PP. 631-653, April 1990.
8. Compton. Jr., R.T., 'Adaptive Antennas : Concepts and performance', Prentice-Hall, Englewood cliffs, New-Jersey, 1987.
9. Dahlquist, G., 'Numerical Methods', Prentice-Hall, New York, 1974.

10. Du, W. and Kirilin, R.L., 'Improved spatial smoothing technique for DOA estimation of coherent signals', IEEE Trans. Sig. process., Vol.39, PP. 1208-1210, May 1991.
11. Evans, J.E., 'Aperture sampling techniques for precision direction finding', IEEE Trans. Aerospace and Electronic sys., Vol. AES-15, PP. 891-895, June 1979.
12. Friedlander, B., 'Lattice filter for Adaptive processing', Proc. IEEE, Vol.70, PP.829-860, Aug. 1982.
13. Frost, O.L., 'An algorithm for linearly constrained adaptive array processing', Proc. IEEE, Vol.60, PP. 926-935, Aug. 1972.
14. Gabriel, W.F., 'Adaptive Arrays - an introduction', Proc. IEEE, Vol.64, PP.239-272, Feb. 1976.
15. Gentleman, W.M. and Kung, H.T., 'Matrix triangularization by systolic arrays', Proc. SPIE, Real time signal processing-IV, Vol.298, PP.298-303, 1981
16. Griffiths, L.J., 'A Simple adaptive algorithm for real time processing in antenna arrays', Proc. IEEE, Vol.57, PP.1696-1704, Oct. 1969.
17. Godard, D., 'Channel estimation using a Kalman filter for fast data transmission', IBM Journal Research Development, Vol.18, PP.267-273, May 1974.
18. Godara, L.C., 'Beamforming in the presence of correlated arrivals using structured correlation matrix', IEEE Trans. Acoust. Speech Sig. Process., Vol.38, PP.1-29, Jan. 1990.
19. Heiligman, G.M., and Purdy, R.J., 'Graceful degradation of an adaptive beamforming processor', IEEE Trans. Aerospace and Electron. Sys., Vol.28, PP.305-314, Jan. 1992.



20. Howells, P.W., 'Exploration in fixed and adaptive resolution of GE and SURC', IEEE Trans. Antennas and propagat, Vol.24, PP.575-583, Sept. 1976.
21. Hudson, J.E., 'Adaptive Array Principles', Peter Perigrinus, London 1981.
22. Haykin, S., 'Adaptive filter theory', Prentice-Hall, New-Jersey, 1986.
23. IEEE Transactions on Antennas and propagation, Vol.AP-12, March 1964.
24. Indukumar, K.C., and Reddy, V.U., 'A Note on Redundancy Averaging', IEEE Trans. Signal Process., Vol.40, PP.466-469, Feb.1992.
25. Jagadeesha, S.N., Sinha, S.N. and Mehra, D.K., 'Spatial averaging techniques for coherent interference suppression in optimum beamformers' JINA 1992, International symposium on antennas, Nice, France, 12-14 Nov. 1992.
26. Jagadeesha, S.N., Sinha, S.N. and Mehra, D.K., 'A recursive modified Gram-Schmidt algorithm based adaptive beamformer', To be published in Signal processing, Vol. 39, 1994.
27. Kalman, H.J., 'Transversal filters', Proc.IRE, Vol.28, PP.302-310, March 1940.
28. Kawase, H.S.T., and Tokumaru, H., 'Recursive least squares circular lattice and estimation algorithms', IEEE Trans. Acoust. Speech. Sig. Process., Vol.31, PP. 228-231, Feb. 1993.
29. Lawson, C.L. and Hanson, R.J., 'Solving least square problems', Prentice-Hall, Englewood Cliffs, New-Jersey, 1974
30. Lee, J.H., and Wu, J.F., 'Adaptive beamforming without signal

- cancellation in the presence of coherent jammers', IEEE Proc. Vol.136, Part-F, PP. 169-173, Aug. 1989.
31. Lee, B.H., Chang, B.K., Cha, I.W., Kim, W.K. and Youn, D.H., 'Realization of a generalized sidelobe cancellor', IEEE Trans. CktS. and Sys., Vol. 34, PP.759-764, July 1987.
  32. Lev-Ari. H., 'Modular architectures for adaptive multichannel lattice algorithm', IEEE Trans. Acoust. Speech Sig. Process., Vol. 35, PP. 543-552, 1987.
  33. Lineberger and Johnson, 'The effect of spatial averaging on spatial correlation matrices in the presence of coherent signals', IEEE Trans. Acoust. Speech Sig. Process., Vol.38, PP.880-884, May 1990.
  34. Lewis, P.S., 'Multichannel recursive least-squares adaptive filtering without a desired signal', IEEE Trans. Sig. Process., Vol.19, PP.359-365, Feb. 1991.
  35. Lewis, P.S., 'QR based algorithms for multichannel adaptive least-square lattice filters', IEEE Trans. Acoust. Speech Sig. Process., Vol.38, PP.421-432, March 1990.
  36. Ling, F., Manolakis, D., and Proakis, J.G., 'A recursive modified Gram-Schmidt algorithm for least-squares estimation', IEEE Trans. Acoust. Speech Sig. Process., Vol.34, PP.829-836, Aug.1986.
  37. Ling. F, 'Givens rotation based least-square lattice and related algorithms', IEEE Trans. Sig. Processing, Vol.39, PP.1541-1551, July 1991.
  38. Ling, F., and Proakis, J.G., 'Numerical accuracy and stability : two problems of adaptive algorithm caused by round off errors', IEEE Proc. ICASSP 84, San Diego CA, March 1986.

39. Ling, F., and Proakis, J.G., 'A generalized multichannel least-square lattice algorithm based on sequential processing stages', IEEE Trans. Acoust. Speech Sig. Process., Vol.32, PP.381-389, 1984.
40. McWhirter, J.G., and Proudler, I.K., 'Orthogonal lattice algorithm for adaptive filtering and beamforming', INTEGRATION, the VLSI Journal 14, PP.231-247, 1993.
41. Monzingo, R.A., and Miller, T.W., 'Introduction to adaptive arrays', Wiley Interscience, John Wiley and Sons, 1980.
42. Murray, W., editor, 'Numerical methods for unconstrained optimization', Academic Press, New York, 1972.
- ✓43. Mehra, D.K., 'A generalized least-squares fast transversal filter algorithm for the decision feedback equalization of dispersive channels', Sig. Processing, Vol-21, PP.241-250, 1990.
44. Orfanidis, S.J., 'Optimum Signal processing-An Introduction, New York, Macmillan publishing company, 1988.
45. Park, S., and Un, C.K., 'Parallel modified Spatial smoothing algorithm for coherent interference cancellation', Signal processing, Vol.24, PP.219-317. 1991.
46. Pei, S.C., Yeh, C.C., and Chiu, S.C., 'Modified spatial smoothing for coherent jammer suppression without signal cancellation', IEEE Trans. acoust. speech sig. process., Vol.36, PP.412-415, March 1988.
- ✓47. Plackett, R.L., 'Some theorems in least-squares', Biometrika, Vol.37, PP.149, 1972
48. Proakis, J.G., 'Digital communications', Second edition, New York, McGrawHill, 1989.

49. Proudler, I.K., McWhirter, J.G. and Shepered, T.J., 'QRD based lattice filter algorithms', proc. SPIE, Int. Soc. Opt. Eng., Aug. 1989.
50. Pillai, S.U. and Kwon, B.H., 'Forward/backward spatial smoothing for coherent signal identification', IEEE Trans. Acoust. speech Sig. Process., Vol.37, PP.8-15, Jan 1989.
51. Regalia, P.A., and Bellanger, M.G., 'On the duality between fast QR methods and lattice methods in least-square filtering', IEEE Trans. Signal processing, Vol.39, PP.879-891, April 1991.
52. Reddy, V.U., Paulraj, A. and Kailath, T., 'Performance analysis of the optimum beamformer in the presence of correlated sources and its behaviour under spatial smoothing', IEEE Trans. Acoust. Speech Sig. Process., Vol.35, PP.927-936, July 1987.
53. Satorius, E.H., and Pack, J.D., 'Application of least-square lattice algorithms to adaptive equalization', IEEE Trans. Comm. Sys., Vol. Com-29, PP.136-142, Feb.1981.
54. Shor, S.W.W., 'Adaptive techniques to discriminate against coherent noise in a narrowband system', Journal of Acoustic society of America, Vol.39, PP. 74-78, Jan. 1966.
55. Stearns, S.D. and Hush, D.R., 'Digital signal analysis', Englewood Cliffs, New-Jersey, Prentice-Hall, 1990.
56. Su, Y.L., Shan, T.J. and Widrow, B., 'Parallel spatial processing : a cure for signal cancellation in adaptive arrays', IEEE Trans. Antennas and propagat., Vol. 34, March 86, PP. 347-354.
57. Shan, T.J., and Kailath, T., 'Adaptive beamforming for coherent signals and interference', IEEE Trans. Acoust. Speech Sig. process., Vol.33, PP : 527-536, June 1986.

58. Slock, D.T.M., and Kailath, T., 'Multichannel fast transversal filter algorithms for adaptive broadband beamforming', 22, SPIE, Vol.1152, Advanced algorithms and architectures for signal processing IV, 1989.
59. Takao, K., and Kikuma, N., 'An adaptive array utilizing an adaptive spatial averaging technique for multipath environments', IEEE Trans. Antenna propagat., Vol.35, PP. 1389-1395, Dec.1987.
60. Van Atta, L.C. , 'Electromagnetic Reflection', U.S. patent 2908002, October 6, 1959 (As cited in Ref.41).
61. Vanloan, C.F. and Golub, G.H., 'Matrix computations', Oxford Academic, 1983.
62. Ward, C.R., Hargrave, P.J., and McWhirter, J.G., 'A novel algorithm and architecture for adaptive digital beamforming', IEEE Trans. Antennas and propagat., Vol. AP-34, PP. 338-345, March 1986.
63. White, W.D., 'Angular spectra in radar application', IEEE Trans. Aerospace and Electron. Sys., Vol.15, PP.895-899, 1979.
64. Widrow, B., Duvall, K.T. , Gooch, B.P., and Newman, W.C., 'Signal cancellation phenomenon in adaptive antennas: causes and cures' IEEE Trans. Antennas and propagat., Vol. 30, PP.469-478, May 1982.
65. Widrow, B., Mantey, P.E., Griffiths, L.J. and Goode, B.B., 'Adaptive antenna systems', Proc. IEEE, Vol.55, PP.2143-2157, Dec. 1967.
66. Williams, R.T., Surendra Prasad, Mahalanabis, A.K., and Sibul, L.H., 'An improved spatial smoothing technique for bearing

- estimation in multipath environment', IEEE Trans, Acoust. Speech Signal process., Vol.36, PP.425-431, April 1988.
67. Yang, B., and Bohme, J.F., 'Rotation Based RLS algorithm; unified derivations, numerical properties and parallel implementation', IEEE Trans. Sig. Process., Vol.40, PP.1151-1167, May 1992.
68. Yang, B., and Bohme, J.F., 'On a parallel implementation of the multichannel adaptive least-square lattice filter', Proc. URSI ISSSE, PP.236-279, Sept. 1989.
69. Yeh, C.C., Pei, S.C., and Pei, S.C., 'On the coherent interference suppression using a spatially smoothing adaptive array', IEEE Trans. Antennas and propagat., Vol.37, PP.851-857, July 1989.
70. Yuen, S.M., 'Exact least-square adaptive beamforming using an orthogonalization network', IEEE Trans. Aerospace and Electron. Sys., Vol.27, PP.311-330, March 1991.
71. Najm, W.G., 'Constrained least-squares in adaptive imperfect arrays', IEEE Trans. Antennas and propagat., Vol.38, PP.1874-1876, Nov. 1990.

## RESEARCH PAPERS OUT OF THE PROPOSED WORK

1. Jagadeesha, S.N., Sinha, S.N. and Mehra, D.K., 'Spatial averaging techniques for coherent interference suppression in optimum beamformers' JINA 1992, International symposium on antennas, Nice, France, 12-14 Nov. 1992.
2. Jagadeesha, S.N., Sinha, S.N. and Mehra, D.K., 'A recursive modified Gram-Schmidt algorithm based adaptive beamformer', To be published in Signal processing, Vol. 39, 1994.

