Australian Telecommunications Research Institute

An Adaptive Antenna Array Processor with Derivative Constraints

John Tuthill

This thesis is presented as part of the requirements for the award of the Degree of Master of Engineering of Curtin University of Technology.

September 1995

130 Hollis Road
WILSON WA 6107
22 September 1995

Director Postgraduate Studies
Australian Telecommunications Research Institute
Curtin University of Technology
BENTLEY WA 6102

Dear Sir/Madam

This thesis entitled *Adaptive Antenna Array Processors with Derivative Constraints* is presented for the Degree of Master of Engineering. I affirm that the work described in this thesis is the result of original research and has not been submitted for a higher degree at any other university or institution.

Yours faithfully

John Tuthill.

I. Acknowledgments

I take this opportunity to thank the many people who have contributed to the production of this thesis. In particular my grateful thanks go to:

Dr Yee Hong Leung for his excellent supervision, advice and enormous amounts of patience and time spent refining my publications and drafts of this thesis.

Professor Antonio Cantoni again for his excellent supervision and motivation, for his confidence in me at the beginning of my studies and for his guiding hand throughout the course of this research.

My colleague *Dr Ian Thng* for his advice on many aspects of this research, for his help in getting my computer to behave its-self and for his friendship.

My wife, *Sandra* for her love, support and understanding, without which this work would simply have been impossible.

My parents *Vic and Elizabeth* and my brother *Peter* for being the best Dad, Mum and Brother I could ask for.

My colleagues at the Australian Telecommunications Research Institute for their support, friendship and much needed distractions and for providing a friendly and inspiring research environment.

II. Abstract

In antenna array processing it is generally required to enhance the reception or detection of a signal from a particular direction while suppressing noise and interference signals from other directions. An optimisation problem often posed to achieve this result is to minimise the array processor mean output power (or variance) subject to a fixed response in the array look direction. The look direction requirement can be met by imposing a set of linear constraints on the processor weights to yield what is known as the *Linearly Constrained Minimum Variance* (LCMV) processor. It has been found, however, that LCMV processors are susceptible to errors in the assumed direction of arrival of the desired signal. To achieve robustness against directional mismatch, additional constraints known as *derivative constraints* can be introduced. These constraints force the first and second order spatial derivatives of the array power response in the look direction to zero. However, constraints corresponding to necessary and sufficient (NS) conditions for these spatial derivatives to be zero are in general quadratic, and the resulting weight vector solution space is non-convex. One approach to this complex problem has been to consider conditions which are only sufficient for the spatial derivatives to be zero. Whilst this results in linear constraints, it exhibits certain anomalous behaviour, for example, dependence on the choice of array phase centre.

Recent work in the area of derivative constraints has resulted in a method for efficiently solving the non-convex output power minimisation problem with quadratic derivative constraints. The optimisation problem addressed assumes that the input signal statistics and hence the input signal autocorrelation matrix $\boldsymbol{R}$ are known. In practice, $\boldsymbol{R}$ must be estimated from the receiver data.

The main contribution of this thesis is the derivation of a new adaptive algorithm which implements an adaptive array processor with look direction plus 1st and 2nd order NS derivative constraints. The new algorithm is derived from the well-known Recursive Least Squares (RLS) technique but allows linear and quadratic constraints to be incorporated within the recursive framework. The algorithm offers the high

performance characteristics associated with RLS methods, namely, fast convergence and high steady-state accuracy. The work encompasses a study of the characteristics of the algorithm in terms of numerical robustness, convergence properties, tracking and computational complexity.

The study of the numerical properties of the algorithm has led to the second important contribution of this thesis: the identification of a parameter which is central to the numerical stability of the algorithm in a practical fixed precision environment. We show that this parameter is bounded during stable operation and can therefore be used to detect the onset of numerical instability within the algorithm. In addition, we show how existing techniques can be used to significantly improve the numerical robustness of the algorithm.

Another important contribution of the thesis stems from an investigation into the multimodal nature of the quadratic, equality constrained optimisation problem resulting from the use of second order NS derivative constraints. In particular, we show that for a linear antenna array operating under certain conditions, the complex multimodal optimisation problem can be greatly simplified. This has important implications in both optimum and adaptive array signal processing.

III. Contents

Appendices:

IV. List of Figures

V. List of Tables

VI. Acronyms and General Mathematical Symbols

VI.1: Acronyms

Acronym	Description
BPF	Band Pass Filter
CRLS	Conventional Recursive Least Squares
GSC	Generalised Sidelobe Canceller
IF	Intermediate Frequency
IQ	Inphase and Quadrature
LCMV	Linearly Constrained Minimum Variance
LMS	Least Mean Squares
LS	Least Squares
MVDR	Minimum Variance Distortionless Response
NS	Necessary and Sufficient
QF	Quadrature Filter
RLS	Recursive Least Squares
SMI	Sample Matrix Inversion
2D	2-Dimensional
3D	3-Dimensional

VI.2: General Mathematical Symbols

$\boldsymbol{A}^H$	Hermitian or conjugate-transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^T$	Transpose of matrix $\boldsymbol{A}$
C	Field of complex numbers
C^L	$L \times 1$ dimensional field of complex numbers
$C^{L \times L}$	$L \times L$ dimensional field of complex numbers
$\mathrm{E}[x]$	Expected value of x
$f'(x)$	First derivative of $f(x)$ with respect to x
$f^{(m)}(x)$	mth derivative of $f(x)$ with respect to x
H^*	Conjugate of the complex scalar H
$\boldsymbol{I}_n$	$n \times n$ identity matrix
$\mathrm{Im}[x]$	The imaginary part of the complex variable x
$\min_x[f(x)]$	The value of x which minimises the function $f(x)$
$O(L)$	Of the Order of L
$\mathcal{R}$	Field of real numbers
$\mathcal{R}^L$	$L \times 1$ dimensional field of real numbers
$\mathcal{R}^{L \times L}$	$L \times L$ dimensional field of real numbers
$\mathrm{Re}[x]$	The real part of the complex variable x
$\overset{\vee}{s}(t)$	The Hilbert transform of $s(t)$

$\mathrm{tr}[\boldsymbol{R}]$	Trace of the matrix $\boldsymbol{R}$
Z	Ring of integers
Z_r^+	Ring of positive integers greater than 0 and less than or equal to r, i.e., $\{x \in Z : 0 < x \leq r\}$
$\boldsymbol{0}_n$	Zero vector of dimension n
$\forall$	For all
$\in$	An element of
$\frac{\partial^m \rho(\theta)}{\partial \theta^m}$	mth partial derivative of $\rho(\theta)$ with respect to θ
$\left.\frac{\partial^m \rho(\theta)}{\partial \theta^m}\right\|_{\theta=\theta_0}$	$\frac{\partial^m \rho(\theta)}{\partial \theta^m}$ evaluated at θ_0
∇_w	The gradient operator, $\frac{\partial}{\partial w}$

1. Introduction

1.1 Background and Motivation

1.1.1 Antenna Array Processing

Antenna array processing has received a great deal of attention in the last three decades [1-10, 17]. The vast amount of research in this area has seen applications in such diverse fields as seismology [7], acoustics [8], SONAR [9] and RADAR [10].

An antenna array consists of two or more spatially distributed antenna elements. By combining the signals received by these elements and exploiting the spatial characteristics of the spatially propagating signals, the reception or detection of a signal from a particular direction in the array environment can be enhanced [11-15].

Array processors can be separated into a number of broad classes depending, for example, on the bandwidth of the signals they are required to process or the space in which the processing is carried out [12].

Narrowband techniques for array signal processing assume signals whose bandwidths are much smaller than the mean frequency of their power spectrum and whose modulation envelope is almost constant across the array at any given instant [13, 16]. In contrast, these conditions are not assumed for the case of broadband processing where the propagating signals concerned are assumed to have large bandwidths. In narrowband processing the time-dependent and space-dependent quantities of the signals are separable and beamforming can be achieved simply by phase shifting the signals received from individual elements by appropriate amounts before summing [16].

The actual array signal processing can be carried out in either *element space* or *beam space* [4, 12]. In an element space processor, the signals derived from the elements are processed directly while in a beam space processor the antenna data is first transformed to form a set of beams in space upon which the processor operates.

In this thesis we are concerned only with narrowband, element space antenna array processors.

1.1.2 Optimum Antenna Array Processors

In an optimum array processor, the processor's parameters, or weights, are chosen so that some aspects of the array's performance are optimised, for example, maximum signal to noise ratio or a certain beam pattern.

The optimisation may depend on the input data, for example, by minimising the signal to noise ratio for a given source scenario [8, 13]. Alternatively, the optimisation may be independent of the input data and be dictated rather by some user specified criteria, for example, producing a certain beam pattern [17]. An optimisation process that relies on knowledge of the input data is known as *data dependent* and one which does not rely on the input data is known as *data independent*.

The optimum array processing problem of interest in this thesis is data dependent.

1.1.3 The LCMV Antenna Array Processor

One commonly used approach in optimum, data dependent antenna array processing is the *Linearly Constrained Minimum Variance* (or LCMV) beamformer [3, 5, 13, 18-23]. As the name suggests, the design of an LCMV beamformer is based on minimising the output variance or power of the array processor thus minimising the output due to any noise and interference sources. However, if this was to be the only design criterion the processor would null not only the noise and interference but also the desired signal. To prevent this, a set of linear constraints can be imposed on the weights of the processor which ensure a beam pattern which has unity gain in the direction of arrival of the desired signal. These constraints effectively protect the *look direction* in the array's operating environment from signal suppression.

In practice the LCMV beamformer is known to produce super-resolution in the look direction of the array and can reject signals close to this direction. Accordingly, if there is a small difference between the actual direction of arrival of the desired signal

and the array look direction, the desired signal may actually be treated as interference and be suppressed. Pointing errors such as these may be introduced in a number of ways, for example, uncertainty in the actual target signal direction, quantisation effects in the weights from the finite precision implementation of the array processor and uncertainties in the antenna element positions.

The above difficulty has important implications in the two broad classes of application scenario we are interested in:

1. spatial estimation applications where the array is used to scan the bearing space for signal sources, and

2. signal reception applications where the direction of a particular desired signal source is known to within some tolerance and there are interference signals coming from unknown directions.

In the case of spatial estimation, the bearing space is divided up into small regions or *bins* whose signal characteristics can be individually determined by the array processor as the look direction is stepped through the bearing space. In this application the number of bins required is determined by the beamwidth of the array processor, i.e., more bins will be required for a narrower beam in order to reduce the risk of missing a signal source. As an optimum weight vector must be determined for each new look direction, the computational load on the processor will depend largely on the number of bins.

In the case of signal reception applications, the actual direction of arrival of the desired source signal may not be known exactly or there may be some error in the beam steering angle of the array.

Therefore, in both classes of application, it would be beneficial if the acceptance angle of the processor could be increased in the look direction. A number of methods to achieve this have been suggested in the literature, for example, through the use of additional constraints, known as *point constraints* [13, 27, 28], which force the array

response to satisfy certain gains in various directions about the look direction. It has been found, however, that closely spaced point constraints can result in a poorly conditioned constraint matrix [29]. This will lead to numerical difficulties in the solution of the optimisation problem.

1.1.4 Derivative Constraints

Another method of broadening the beam in the look direction but which avoids the ill-conditioning problem is to use spatial derivative constraints [2, 4, 13, 14, 18, 19, 21-25, 27-30].

Spatial derivative constraints were first suggested by Er and Cantoni [23] and act by forcing conditions on the weights such that the first and up to the nth-order spatial derivatives of the array power response in the look direction are zero. The idea of derivative constraints is closely related to the theory of maximally flat filters and an nth order derivative constrained array processor is said to have an nth order maximally flat response in the look direction. In practice, only the first and second order derivatives are considered, as imposing higher order constraints reduces the noise rejection capabilities of the processor [21-23].

A major difficulty with derivative constraints is that they yield a set of constraints which are quadratic in the processor weights. A complex, multimodal optimisation [32] problem thus results. To avoid this difficulty a number of linear approximations to the quadratic constraints have been suggested in the literature [18, 19, 23]. While these approximations allow the much simpler linear optimisation problem to be solved, they nevertheless represent only sufficient conditions for a maximally flat response. Moreover, they have been found to depend on the choice of array phase centre [18, 21, 22, 30] and, on occasion, these constraints may not even be consistent [22, 24].

A recently developed method for efficiently simplifying and solving a general quadratic equality constrained optimisation problem is presented in [22]. The technique reduces the optimisation problem with quadratic constraints to a two-stage minimisation problem where the first stage is a linearly constrained optimisation problem parameterised by a number of scalar variables and the second stage involves finding the optimum values for these variables. Note that the second stage of the two stage minimisation problem can be reduced essentially to one of finding the roots of a high order univariate polynomial.

We will refer to the quadratic derivative constraints as NS constraints as they correspond to Necessary and Sufficient conditions for the spatial derivatives to be zero.

1.1.5 Adaptive Arrays

Although the method presented in [22] solves the optimum but not the adaptive problem, it, nevertheless gives significant insight into how an adaptive, derivative constrained antenna array processor might be found.

The design of the optimum array processor requires *a priori* information about the statistics of the input signals to be processed, i.e., the input signal correlation matrix. The array processor is optimum, therefore, only when the statistical characteristics of the input signals match the *a priori* information. In practice the input signal statistics may not be known or may be changing with time. One approach is to estimate the correlation matrix using a number of samples of the input data and to use this estimate in place of the true matrix in the optimisation problem. This open loop or *non-recursive* approach, often called "Estimate and Plug" [31], can be computationally expensive in a real-time implementation. A more efficient approach is to use an *adaptive* method which relies on a *recursive* algorithm and which makes it possible for the processor not only to perform satisfactorily in an environment where the signal statistics are not known completely but also to track slow changes in the signal environment [41].

In essence, then, we seek some *adaptive* algorithm for estimating the optimum weight vector given the input data available. We require the antenna array processor to maintain unity gain and second order flatness of the power response in the look direction, according to the NS derivative constraints, thus ensuring robustness against mismatches between the array look direction and the actual signal direction. We also require the processor to adaptively null any interference signals from all other directions.

Note that the generally accepted meaning of an ***adaptive*** algorithm in the signal processing context is [41]:

(i) that the algorithm starts from pre-defined initial conditions with little or no knowledge of the signal environment,

(ii) that in a stationary input signal environment, the weight vector produced by the algorithm converges toward the optimum solution, and

(iii) in a non-stationary environment, that the algorithm will respond to slow changes in the input signal statistics, i.e. track slow changes in the input signal environment.

A number of factors are important in measuring the overall effectiveness of an adaptive algorithm [41], these include:

- Rate of convergence. This measures of the speed with which the algorithm converges to a steady state solution in a stationary input environment.

- Tracking. This characterises the effectiveness of the algorithm in responding to changes in the signal statistics.

- Misadjustment. This is a quantitative measure of the difference between the optimum weight vector and the ensemble averaged weight vector of the adaptive processor.

- Numerical stability. Since adaptive algorithms are recursive by nature, small errors that creep into the algorithm as a result of a finite precision implementation can become very large if left uncorrected and may eventually cause the algorithm to become unstable and fail [44, 45, 47-49].

- Computational requirements. As one of the primary reasons for considering adaptive methods is efficiency in computation, factors such as the number of mathematical operations and data and program storage required for one complete iteration of the algorithm are obviously important considerations.

Many adaptive array processing techniques have been proposed to solve a wide variety of optimum problems in an adaptive manner [1-6, 9, 10, 12, 13, 15, 18, 20, 28, 33, 35-41, 45-47]. The optimum problems concerned, however, are generally formulated so as to ensure there is only one optimum solution which the adaptive algorithm is required to estimate and track, i.e. they are unimodal. The adaptive application we are investigating here is, however, different as the adaptive algorithm must try to estimate an optimum weight vector which is found from the solution of a multimodal optimisation problem. Since the optimum weight vector may not be unique, depending on factors such as the array structure, signal scenario and the look direction, this poses some interesting questions about the meaning of adaptation in this context as conventional measures of an adaptive algorithms performance, in particular the misadjustment, become unclear.

1.2 Contributions of the Thesis

The major contributions of the thesis are in the area of adaptive algorithms for narrowband beamformers with look direction and first and second order NS derivative constraints. Specifically, the thesis reports original work as summarised below:

(i) A New Adaptive Algorithm

- We present a new adaptive algorithm for narrowband beamformers with look direction and first and second order necessary and sufficient derivative constraints.

(ii) Numerical Stability of the Algorithm

- We investigate the numerical characteristics of the algorithm and provide methods by which the algorithm can be made numerically robust in a finite precision implementation.
- We identify a parameter which is central to the round-off error propagation mechanism within the algorithm and which can be used to detect the onset of instability.
- We propose methods for stabilising the algorithm by preventing the accumulation of round-off error.

(iii) Adaptive Algorithms for Multimodal Problems

- We investigate the effects of multiple optimum solutions on both the optimum and adaptive processors.

Some of the work presented in this thesis has already been reported in [50].

1.3 Organisation of the Thesis

This thesis is organised into seven chapters.

Chapter 2 outlines the optimum narrowband array processing problem addressed in this thesis and presents the signal model, array structures and the basic assumptions and definitions used throughout the work.

Chapter 3 identifies the key components which must be contained in a suitably formulated adaptive antenna array processor with derivative constraints. We investigate two initial approaches to the design of the adaptive processor and use computer studies to highlight the shortcomings in these methods. Although impractical, these algorithms, nevertheless, give a starting point for the development of better methods.

In Chapter 4, we present two of the major contributions of the thesis, namely, (i), the development of the new two-stage adaptive algorithm, and (ii), the identification of a parameter which is critical to the stability of the algorithm. In addition, existing methods for stabilising the standard *Recursive Least Squares* (RLS) algorithm are discussed briefly. These methods not only form part of the new algorithm but also provide an insight into further numerical stability aspects of the new algorithm.

In Chapter 5, we investigate some techniques known as *Global Methods* which allow all the roots of a univariate polynomial to be found simultaneously. These methods are of particular interest as they allow the second stage of the new algorithm to be implemented in a very efficient fashion.

In Chapter 6, the multimodal nature of the optimisation problem is studied and the effect of more than one optimum solution on both the optimum and the adaptive processors is investigated.

In Chapter 7, we present a summary of the work reported in this thesis and draw some conclusions. We also provide some suggestions for further work.

1.4 References

[1] L. J. Griffiths, "A Simple Adaptive Algorithm for Real-Time Processing in Antenna Arrays", *IEEE Proc.*, vol. 57, no. 10, pp. 1696-1969, Oct 1969.

[2] S. P. Applebaum and D. J. Chapman, "Adaptive Arrays with Main Beam Constraints", *IEEE Trans. Antennas and Propagat.*, vol. AP-24, no. 5, pp. 650-662, Sep. 1976.

[3] K. Takao, M. Fujita and T. Nishi, "An Adaptive Array under Directional Constraint", *IEEE Trans. Antennas and Propagat.*, vol. AP-24, no. 5, pp. 662-669, Sep. 1976.

[4] A. M. Vural, "A Comparative Performance Study of Adaptive Array Processing", *ICASSP 77 Conference Record*, pp. 695-700, May 1977.

[5] O. L. Frost, III, "An Algorithm for Linearly Constrained Adaptive Array Processing", *Proc. IEEE*, vol. 60, no. 8, pp. 926-935, August 1972.

[6] I. S. Reed, J. D. Mallett, L. E. Brennan, "Rapid Convergence Rate in Adaptive Arrays", *IEEE Trans. Aerospace and Electronic Systems*, vol. 10, pp 853 - 863, November 1974.

[7] P. E. Green Jr, E. J. Kelly Jr and M. J. Levin, "A Comparison of Seismic Array Processing Methods", *Geophys. J. R. Astron. Soc.*, vol. 11, pp. 67-84, 1966.

[8] W. Vanderkulk, "Optimum Processing of Acoustic Arrays", *J. Br. Inst. Radio Eng.*, vol. 34, pp. 285-292, 1963.

[9] J. W. R. Griffiths and J. E. Hudson, "An Introduction to Adaptive Processing in a Passive Sonar System", *G. Tacconi (Editor), Aspects of Signal Processing*, Pt. 1, NATO Advanced Study Institute Series, D. Reidel Publishing Co., Dordrecht, Holland, 1977.

[10] L. E. Brennan and I. S. Reed, "Theory of Adaptive Radar", *IEEE Trans. Aerospace and Electronic Systems*, AES-9 pp. 237-252, Nov. 1973.

[11] I. Thng, *Derivative Constrained Broadband Antenna Array Processors*, PhD Thesis, Australian Telecommunications Research Institute, Curtin University of Technology, WA, Australia, Mar. 1995.

[12] E. Nicolau and D. Zaharia, *Adaptive Arrays*, Elsevier Science Publishers, Amsterdam, Netherlands, 1989.

[13] J. E. Hudson, *Adaptive Array Principles*, Peter Peregrinus, London, 1981.

[14] M. H. Er, *Optimum Antenna Array Processors with Linear and Quadratic Constraints*, PhD Thesis, Electrical and Computer Engineering, University of Newcastle, NSW, Australia, May 1985.

[15] L. W. R. Griffiths, "Adaptive Array Processing - A Tutorial", *IEE Proc.*, vol. 130, Pts. F and H, no. 1, pp. 3-10, Feb. 1983.

[16] L. C. Godara and A. Cantoni, "Signal Representation for Array Processing", Tech. Report EE8203, Dept. of Electrical and Computer Engineering, University of Newcastle, NSW, 2308, Australia, August 1982

[17] C. L. Dolph, "A Current Distribution for Broadside Arrays which Optimises the Relationship between Beamwidth and Sidelobe level", *Proc. IRE*, vol. 34, pp. 335-348, 1946.

[18] K. M. Buckley and L. J. Griffiths, "An Adaptive Generalised Sidelobe Canceller with Derivative Constraints", *IEEE Trans. Antennas Propagat.*, vol. 34, no. 3, pp. 311-319, March 1986.

[19] C. Y. Tseng, "Minimum Variance Beamforming with Phase-Independent Derivative Constraints", *IEEE Trans. Antennas Propagat.*, vol. 40, no. 3, pp. 285-294, Mar. 1992.

[20] R. T. Compton, Jr, *Adaptive Antennas: Concepts and Performance*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[21]I. Thng, A. Cantoni and Y. H. Leung, "Derivative Constrained Optimum Broad-Band Antenna Arrays", *IEEE Trans. Signal Proc.*, vol. SP-41, no. 7, pp. 2376-2388, July 1993.

[22] I. Thng, A. Cantoni and Y. H. Leung, "Constraints for Maximally Flat Optimum Broadband Antenna Arrays", *ASPL Report 1994-2*, Curtin University of Technology, Bentley WA 6021, Australia, 1994. (Accepted for publication IEEE Trans. Sig. Proc.)

[23]M. H. Er and A. Cantoni, "Derivative Constraints for Broad-Band Element Space Antenna Array Processors", *IEEE Trans. Acoustics, Speech and Signal Proc.*, vol. 31, no. 6, pp 1378-1393, Dec. 1983.

[24] I. Thng, A. Cantoni and Y. H. Leung, "A New Set of Constraints for Maximally Flat Optimum Broadband Antenna Arrays", *IEEE ICASSP 94 Conference Proc.*, vol. 4, pp 533-536, April 1994.

[25] M. H. Er and A. Cantoni, "Techniques in Robust Broadband Beamforming", *C. T. Leondes (Editor) Control and Dynamic Systems*, Academic Press Inc. San Diego, CA, 1992.

[26] H. Cox, "Resolving Power and Sensitivity to Mismatch in Optimum Array Processors", *The Journal of Acoustical Society of America*, vol. 54, no. 3, pp. 771-785, 1973.

[27] D.Nunn, "Performance Assessments of a Time-Domain Adaptive Processor in a Broadband Environment", *IEE Proc.*, vol. 130, pts. F and H, no. 1, pp. 139-146, Feb. 1983.

[28] D. Nunn, "Suboptimal Frequency-Domain Adaptive Antenna Processing Algorithm for Broadband Environments", *IEE Proc.*, vol. 134, pt. F, no. 4, pp. 341-351, July 1987.

[29] A. K. Steele, "Comparison of Directional and Derivative Constraints for Beamformers Subject to Multiple Linear Constraints", *IEE Proc. F, Commun. Radar Signal Processing.*, and *IEE Proc. H, Microwaves, Opt. and Antennas*, (joint issue), vol. 130, pts. F and H, (1), pp. 41-45, 1983.

[30] A. Cantoni and M. C. Blasikiewicz, "Derivative Constrained Broadband Antenna Array Processors Revisited", *IASTED 2nd International Symp. Signal Proc. and its Applic.*, Gold Coast, Australia, pp. 319-326, Aug. 1990.

[31] W. S. Hodgkiss and L. W. Nolte, "Bayes Optimal versus Estimate and Plug Array Processor Performance when there is Directional Uncertainty", *G. Tacconi (Editor), Aspects of Signal Processing*, Pt. 1, NATO Advanced Study Institute Series, D. Reidel Publishing Co., Dordrecht, Holland, pp. 203-212, 1977.

[32] P. E. Gill, W. Murray and M. H. Wright, *Numerical Linear Algebra and Optimisation*, vol. 1, Addison-Wesley Pub. Co., Redwood City, Calif., 1991.

[33] L. J. Griffiths and C. W. Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming", *IEEE Trans. Antennas and Propagat.*, vol. AP-30, no. 1, pp. 27-33, Jan. 1982.

[34] M. H. Er, "On the Limiting Solution of Quadratically Constrained Broad-Band Beam Formers", *IEEE Trans. Signal Proc.*, vol. 41, no. 1, pp. 418-419, Jan. 1993.

[35] W. G. Najm, "Constrained Least Squares in Adaptive, Imperfect Arrays", *IEEE Trans. Antennas and Propagat.*, vol. 38, no. 11, pp. 1874-1878, Nov. 1990.

[36] L. B. Fertig and J. H. McClellan, "Dual Forms for Constrained Adaptive Filtering", *IEEE Trans. Signal Proc.*, vol. 42, no. 1, pp. 11-23, Jan 1994.

[37] M. H. Er and A. Cantoni, "On an Adaptive Antenna Array Under Directional Constraint", *IEEE Trans. Acoust. Speech and Signal Proc.*, vol. ASSP-33, no. 4, pp. 1326-1328, Oct. 1985.

[38] M. H. Er, "Adaptive Antenna Array under Directional and Spatial Derivative Constraints", *IEE Proc.*, vol. 135, Pt. H, no. 6, pp. 414-419, Dec. 1988.

[39] A. Cantoni and L. C. Godara, "Fast Algorithms for Time Domain Broadband Adaptive Processing", *IEEE Trans. Aerospace and Electric Systems*, vol. AES-18, no. 5, pp 682-699. Sep. 1982.

[40] P. M. Clarkson, *Optimal and Adaptive Signal Processing*, CRC Press, Boca Raton, Florida USA, 1993.

[41] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1991.

[42] S. Lawrence Marple, Jr., *Digital Spectral Analysis with Applications*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1987.

[43] Maurice G. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker Inc., New York, 1987.

[44] L. S. Resende, J. M. T. Romano and M. G. Bellanger, "A Robust FLS Algorithm For Linearly-Constrained Adaptive Filtering", *Proc. IEEE ICASSP*, Adelaide, Australia, pp. III 381- III 384, April 1994.

[45] L. S. Resende, J. M. T. Romano and M. G. Bellanger, "A Fast Least-Squares Algorithm for Linearly -Constrained Adaptive Filtering", Submitted for publication in *IEEE Trans. Signal Processing*, August 1994.

[46] L. S. Resende, J. M. T. Romano and M. G. Bellanger, "A Fast Least Squares Algorithm for Constrained Adaptive Filtering", *Proc. IEEE ICASSP*, vol. IV, pp. 21-24, San Francisco, USA, Mar 1992.

[47] M. H. Verhaegen, "Round-off Error Propagation in Four Generally Applicable, Recursive, Least-squares Estimation Schemes", *Automatica*, vol. 25, no. 3, pp. 437-444, 1989.

[48] G. E. Bottomley and S. T. Alexander, "A Theoretical Basis for the Divergence of Conventional Recursive Least Squares Filters", *Proc. IEEE ICASSP*, Glasgow, Scotland, pp. 908-911, May 1989.

[49] G. E. Bottomley, S. T. Alexander, "A Novel Approach for Stabilising Recursive Least Squares Filters", *IEEE Trans. Signal Processing*, vol. 39, no. 8, pp. 1770-1779, August 1991.

Authors Publications

[50] J. Tuthill, Y. H., Leung and I. Thng, "Adaptive RLS Filters with Linear and Quadratic Constraints", *Proc. IEEE ICASSP*, Detroit, USA, pp. 1424-1427, May, 1995.

[51] J. Tuthill, "On Global Methods for Finding the Roots of a Polynomial Simultaneously", *Adaptive Signal Processing Laboratory Report SPL-TM-007*, ATRI, Curtin University of Technology, Bentley, Australia.

2. *Narrowband Optimum Array Processors*

This chapter outlines the optimum array processing problem addressed by this thesis and presents some of the basic assumptions and definitions employed in the rest of the work.

We begin by introducing the signal model for a general narrowband array of elements positioned in 3D space. The assumptions under which the array signals can be considered narrowband are reviewed and we derive a mathematical representation of the element signals.

Narrowband beamforming structures and the signal representations associated with these structures are then considered and we introduce vector notation for the array weights and input data.

We then arrive at the narrowband Linearly Constrained Minimum Variance, or *LCMV*, beamformer and the optimisation problem for determining the optimum set of weights for this processor.

The discussion on the LCMV beamformer motivates the final section of the chapter on derivative constraints. Here we give the Necessary and Sufficient (NS) conditions on the processor weights for the first and second order spatial derivatives of the array power response to be zero in the look direction of the array. A simple optimum array processing example is presented to demonstrate the efficacy of using derivative constraints.

2.1 Signal Model

Consider the array of omnidirectional, distortionless elements positioned arbitrarily in 3D space as shown in Fig. 2.1.

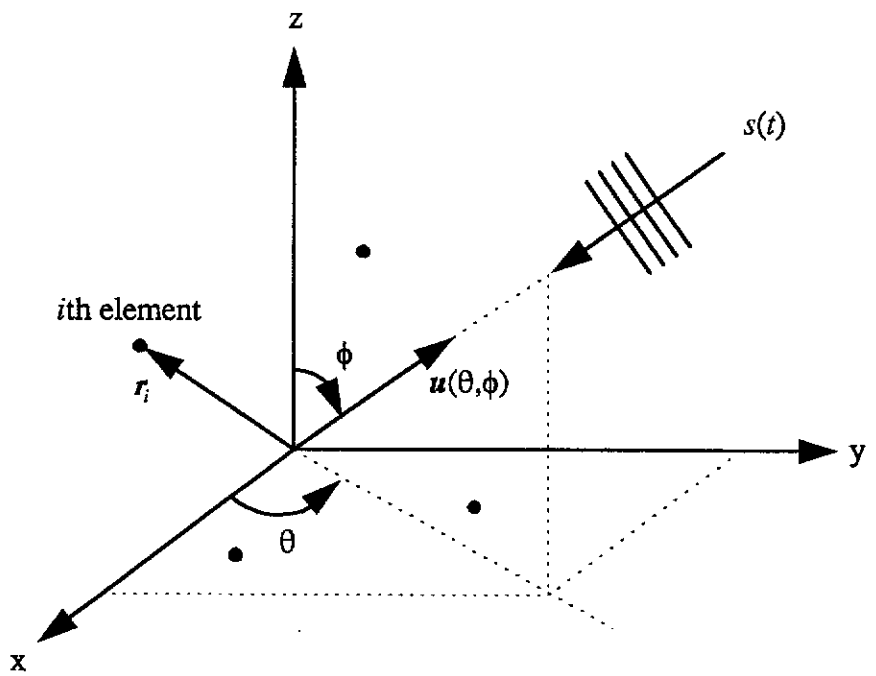


Figure 2.1 Array Geometry

Let the real scalar $s(t)$ represent a directional source located in the far field of the array. By far field we imply that the source is located far enough away from the array that wavefronts generated by the source and incident on the array can be considered to be parallel, i.e., planewaves.

For an array of omnidirectional sensors operating in an ideal isotropic, non-dispersive propagation environment, it can be shown [2] that the output waveform of the ith array element is given by

$$x_i(t) = s\big(t + \tau_i(\theta,\phi)\big) + n_i(t) \tag{2-1}$$

where $n_i(t)$ is the noise at the output and $\tau_i(\theta,\phi)$ is a pure time delay given by

$$\tau_i(\theta,\phi) = \frac{\boldsymbol{r}_i \cdot \boldsymbol{u}(\theta,\phi)}{v} \tag{2-2}$$

and where r_i is the position vector for the ith element, $u(\theta,\phi)$ is a unit vector in the direction of the source and v is the velocity of propagation in the medium. Note that the time reference is taken to be at the origin of the coordinate system and so a plus sign appears in front of the time delay term in (2-1).

The source $s(t)$ can be represented as

$$s(t) = \mathrm{Re}(a(t)) \tag{2-3}$$

and

$$A(f) = \begin{cases} 2S(f) & f > 0 \\ 0 & f < 0 \end{cases} \tag{2-4}$$

where $S(f)$ is the power spectral density of $s(t)$ and $a(t)$ is a complex signal with power spectral density $A(f)$.

Note that $a(t)$ as defined above can be expressed in terms of $s(t)$ as follows:

$$a(t) = s(t) + j\check{s}(t) \tag{2-5}$$

where $\check{s}(t)$ is the Hilbert transform [3, p.74-77] of $s(t)$.

The complex signal $a(t)$ can also be expressed as a modulated carrier in the form

$$a(t) = m(t)e^{j\omega_0 t} \tag{2-6}$$

where $m(t)$ is a complex modulating function and $\omega_0 = 2\pi f_0$ where f_0 is the carrier frequency.

Let $m_I(t)$ and $m_Q(t)$ represent the real (in-phase) and imaginary (quadrature) parts of $m(t)$. The expression for $a(t)$ in (2-6) can be written as

$$a(t) = \left(m_I(t) + jm_Q(t)\right)e^{j\omega_0 t}. \tag{2-7}$$

Note that $m(t)$ is not an analytic function in general and so $m_I(t)$ and $m_Q(t)$ need not be related by the Hilbert transform.

Expanding the exponential term in (2-7), we get

$$a(t) = m_I(t)\cos\omega_0 t - m_Q(t)\sin\omega_0 t + j\left(m_I(t)\sin\omega_0 t + m_Q(t)\cos\omega_0 t\right). \tag{2-8}$$

From (2-3) and (2-8), we arrive at the following representation of the real signal in the medium

$$s(t) = m_I(t)\cos\omega_0 t - m_Q(t)\sin\omega_0 t \tag{2-9}$$

and combining (2-1) and (2-9) gives the following expression for the observable signal at the output of the ith element

$$x_i(t) = m_I\left(t+\tau_i(\theta,\phi)\right)\cos\omega_0\left(t+\tau_i(\theta,\phi)\right) - m_Q\left(t+\tau_i(\theta,\phi)\right)\sin\omega_0\left(t+\tau_i(\theta,\phi)\right) + n_i(t). \tag{2-10}$$

In the case of purely temporal processing, a signal is considered to be narrowband if its bandwidth is much smaller than the mean frequency of its power spectrum. Array processing, however, involves both temporal and spatial sampling of the signal and a further condition on the size of the array must hold for the signal to be considered narrowband. This condition is given by requiring the modulation envelope of the signal, $m(t)$, to be the same across the array at any given time, i.e.,

$$m(t) \cong m(t+\Delta t) \tag{2-11}$$

where Δt is of the order of the time difference between the signal passing the element closest to the source and passing the element farthest from the source.

Using the narrowband assumption, (2-10) can be re-written as

$$x_i(t) = m_I(t)\cos\omega_0\left(t+\tau_i(\theta,\phi)\right) - m_Q(t)\sin\omega_0\left(t+\tau_i(\theta,\phi)\right) + n_i(t). \tag{2-12}$$

2.2 *Narrowband Beamforming*

In narrowband, time-domain antenna array processing, beamforming can be accomplished by combining weighted signals derived from the array elements. A common technique is to derive in-phase and quadrature signals from each of the element signals. These are then weighted and combined in the beamforming process.

Providing certain conditions hold, this process can occur at the element stage, at an intermediate frequency (IF) stage (see Fig. 2.2) or at baseband.

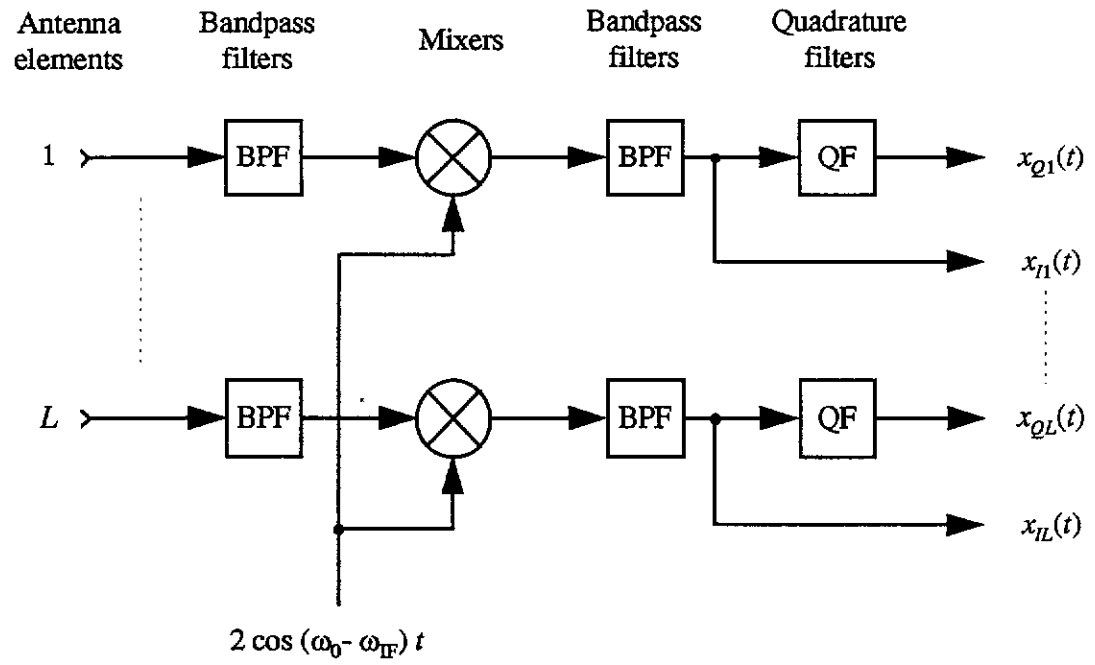


Figure 2.2 Derivation of In-phase and Quadrature Signals

From (2-12), the in-phase signal shown in Fig. 2.2 for the ith element is given by

$$\begin{aligned} x_{Ii}(t) = m_I(t)\cos\left(\omega_{IF}t + \omega_0\tau_i(\theta,\phi)\right) \\ -m_Q(t)\sin\left(\omega_{IF}t + \omega_0\tau_i(\theta,\phi)\right) + n_{Ii}(t) \end{aligned} \tag{2-13}$$

If the quadrature signals are generated at a stage where the highest frequency components of $m_I(t)$ and $m_Q(t)$ are lower than the mean frequency of the signal at that stage **or** if the quadrature signals are generated at baseband **and** $m_I(t)$ and $m_Q(t)$ are related by the Hilbert transform, i.e., $m_I(t) = \check{m}_Q(t)$, then using (2-5), (2-8) and (2-11), the quadrature signal for the ith element at the beamforming stage is given by

$$\begin{aligned} x_{Qi}(t) = m_I(t)\sin\left(\omega_{IF}t + \omega_0\tau_i(\theta,\phi)\right) \\ +m_Q(t)\cos\left(\omega_{IF}t + \omega_0\tau_i(\theta,\phi)\right) + n_{Qi}(t). \end{aligned} \tag{2-14}$$

Let the set of L in-phase signals at the beamformer stage, $\boldsymbol{x}_I(t) \in \mathcal{R}^L$, be defined as

$$\boldsymbol{x}_I(t) = \left[x_{I1}(t), x_{I2}(t), \ldots, x_{IL}(t)\right]^T, \tag{2-15}$$

and the set of L quadrature signals, $\boldsymbol{x}_Q(t) \in \mathcal{R}^L$, be defined as

$$\boldsymbol{x}_Q(t) = \left[x_{Q1}(t), x_{Q2}(t), \ldots, x_{QL}(t)\right]^T. \tag{2-16}$$

Two beamforming systems can be used to represent the signals within the processor and even to actually realise the array processor: the *real* beamforming system where the inphase and quadrature signals derived from the element signals are combined to produce a single real output waveform; and, the IQ (In-phase and Quadrature) system which produces either an analytic or a complex output. An example of an element space real beamformer is shown in Fig. 2.3.

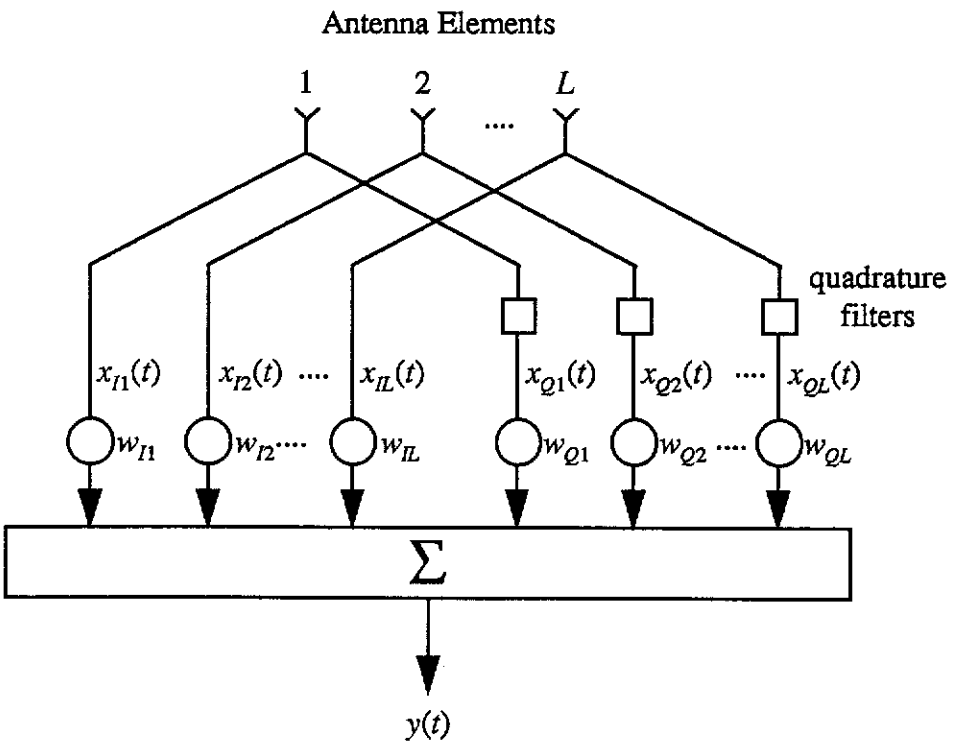


Figure 2.3 An Element Stage Real Beamforming System

The real output of this beamformer is given by

$$y(t) = \boldsymbol{w}^T \boldsymbol{x}(t) \tag{2-17}$$

where $\boldsymbol{w} \in \mathcal{R}^{2L}$ is a vector of the processor weights given by

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_I \\ \boldsymbol{w}_Q \end{bmatrix} \tag{2-18}$$

and $\boldsymbol{x}(t) \in \mathcal{R}^{2L}$ is a vector of the inphase and quadrature signals given by

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{x}_I(t) \\ \boldsymbol{x}_Q(t) \end{bmatrix}. \tag{2-19}$$

Alternatively, an IQ system might be realised as shown Fig. 2.4.

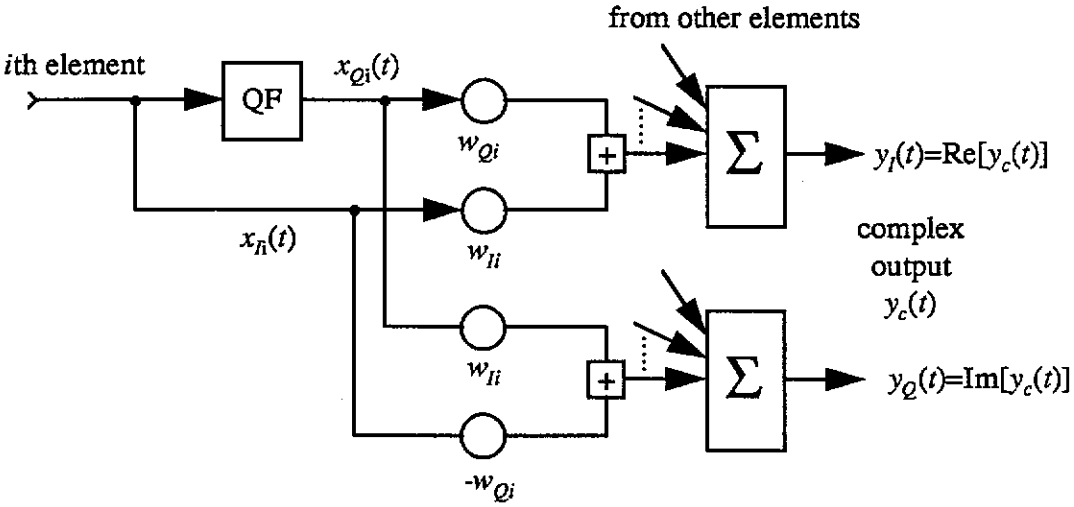


Figure 2.4 An Element Stage IQ Beamforming System

Note from Fig. 2.4 that the processor produces two outputs, $y_I(t)$ and $y_Q(t)$, corresponding to the in-phase and quadrature components of the complex output signal $y_c(t)$. It is important to note too that $y_I(t)$ and $y_Q(t)$ are in general not related by the Hilbert transform [2].

The complex output of this beamformer is given by

$$y_c(t) = \boldsymbol{w}_c^H \boldsymbol{x}_c(t) \tag{2-20}$$

where $\boldsymbol{w}_c \in C^L$ is a vector of complex weights given by

$$\boldsymbol{w}_c = \boldsymbol{w}_I + j\boldsymbol{w}_Q \tag{2-21}$$

and $\boldsymbol{x}_c(t) \in C^L$ is a complex vector given by

$$\boldsymbol{x}_c(t) = \boldsymbol{x}_I(t) + j\boldsymbol{x}_Q(t). \tag{2-22}$$

From (2-14) and (2-21), one can see that the $2L$-dimensional weight vector for the real beamformer is related to the L-dimensional weight vector for the complex

beamformer as follows

$$\boldsymbol{w} = \begin{bmatrix} \mathrm{Re}[\boldsymbol{w}_c] \\ \mathrm{Im}[\boldsymbol{w}_c] \end{bmatrix} \tag{2-23}$$

and the output of the real beamformer can be expressed in complex notation as

$$y(t) = \mathrm{Re}\left[\boldsymbol{w}_c^H \boldsymbol{x}_c(t)\right] \tag{2-24}$$

There are various other forms a narrowband adaptive antenna array processor structure can take, for example, the Generalised Sidelobe Canceller (GSC) [4]. A number of narrowband processors may also be incorporated within a structure as shown in Fig 2.5 to process broadband signals derived from antenna elements. Here the broadband signals from each of the N elements are divided into L frequency bins using FFT blocks. The L narrowband signals for each frequency bin are then processed separately using narrowband processors.

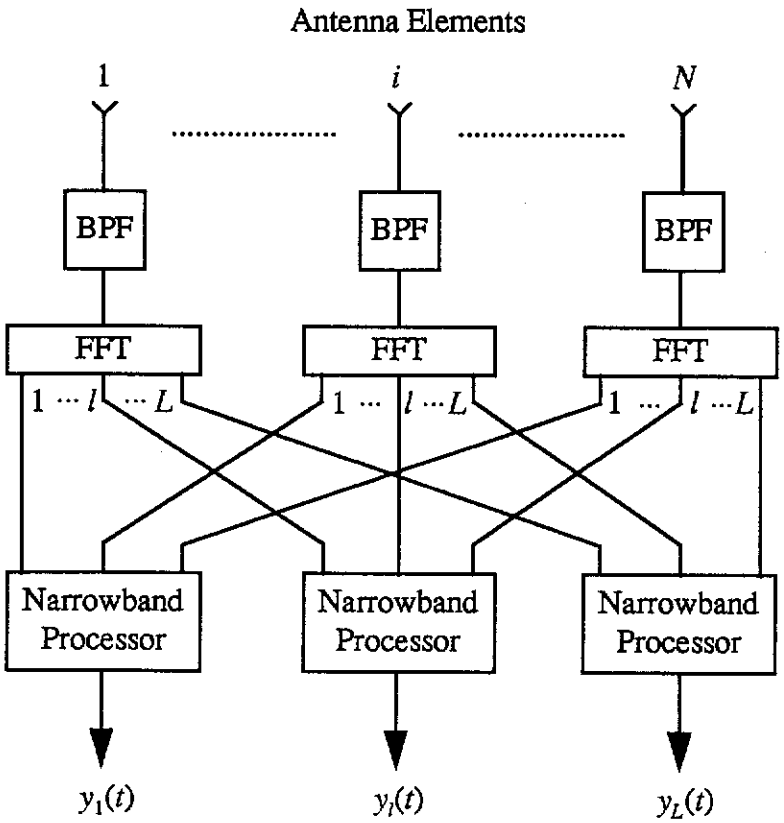


Figure 2.5 A Frequency Domain Element Space Antenna Array Processor

In this thesis we will consider only the real beamforming system of Fig. 2.3, although,

in the derivations that follow, we will be using both the real and the complex notations as discussed above. In addition, we will consider only the direct form processor shown in Fig. 2.3.

It can be shown [2, 4] that, provided the sources and noise can be modelled as stationary, independent, Gaussian zero mean random processes, the real array input signal correlation matrix, $\boldsymbol{R} \in \mathcal{R}^{2L \times 2L}$, is given by

$$\boldsymbol{R} = \mathrm{E}\left[\boldsymbol{x}(t)\boldsymbol{x}^T(t)\right] = \begin{bmatrix} \boldsymbol{S} & \boldsymbol{T} \\ -\boldsymbol{T} & \boldsymbol{S} \end{bmatrix} \tag{2-25}$$

where

$$\boldsymbol{S} = \mathrm{E}\left[\boldsymbol{x}_I(t)\boldsymbol{x}_I^T(t)\right], \tag{2-26}$$

and

$$\boldsymbol{T} = \mathrm{E}\left[\boldsymbol{x}_I(t)\boldsymbol{x}_Q^T(t)\right]. \tag{2-27}$$

The complex input signal correlation matrix, $\boldsymbol{R}_c \in C^{L \times L}$, is given by

$$\boldsymbol{R}_c = \mathrm{E}\left[\boldsymbol{x}_c(t)\boldsymbol{x}_c^H(t)\right] = 2\boldsymbol{S} - j2\boldsymbol{T}. \tag{2-28}$$

2.3 *LCMV and MVDR Beamforming and Derivative Constraints*

2.3.1 The LCMV and MVDR Beamformers

In designing a LCMV array processor, we minimise the array mean output power while maintaining a fixed response (gain) in the array look direction. The frequency response of the array to a plane wavefront of unity amplitude arriving from the direction (θ, ϕ) is given, in complex notation, by the expression

$$H(\theta, \phi) = \boldsymbol{s}^H(\theta, \phi)\boldsymbol{w}_c \tag{2-29}$$

where $H(\theta, \phi) \in C$, $\boldsymbol{w}_c \in C^L$ is a vector of complex weights and $\boldsymbol{s}(\theta, \phi) \in C^L$ is the array steering vector whose ith element is given by

$$\left[\boldsymbol{s}(\theta, \phi)\right]_i = e^{j\omega_0 \tau_i(\theta, \phi)}. \tag{2-30}$$

The superscript, H, in (2-29) represents the conjugate transpose operation.

For a given look direction, (θ_0, ϕ_0), the optimum set of weights for a LCMV beamformer is found from the following linearly constrained optimisation problem

$$\begin{aligned} &\min_{w_c} P(w_c) \\ &\text{Subject to} \quad s^H(\theta_0, \phi_0) w_c = e^{j\gamma} \end{aligned} \tag{2-31}$$

where $P(w_c) \in \mathcal{R}$ is the mean output power of the processor given by

$$P(w_c) = \mathrm{E}\left[|y_c|^2\right] = \mathrm{E}\left[w_c^H x_c x_c^H w_c\right] = w_c^H R_c w_c \tag{2-32}$$

A special case of LCMV beamforming is obtained if we set $\gamma = 0$ in (2-31) yeilding the *Minimum Variance Distortionless Response* or MVDR Beamformer [7, p.61]

$$\begin{aligned} &\min_{w_c} P(w_c) \\ &\text{Subject to} \quad s^H(\theta_0, \phi_0) w_c = 1 \end{aligned} \tag{2-33}$$

Note that in Equations (2-29) to (2-33) we have used the complex notation defined in the previous section.

Using the method of Lagrange Multipliers, the optimum weight vector which minimises the constrained optimisation problem of (2-33) can be shown to be [8]

$$w_{lcmv_{opt}} = R^{-1} s(\theta_0, \phi_0) \left[s^H(\theta_0, \phi_0) R^{-1} s(\theta_0, \phi_0) \right]^{-1}. \tag{2-34}$$

Fig. 2.6 shows a typical bearing response of an MVDR array whose weights are computed in this manner.

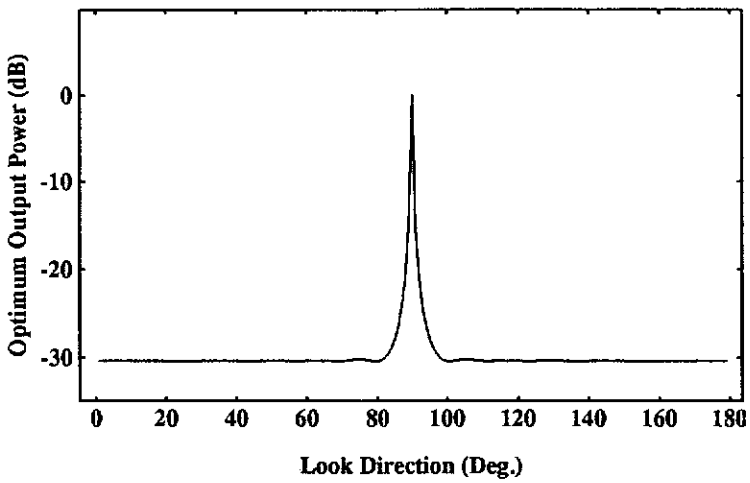


Figure 2.6 MVDR Narrowband Array Processor Optimum Output Power as the Look Direction is Stepped form 0° to 360° in 1° Increments

For simplicity of presentation, only a 2D scenario was considered in this illustration. The plot shows the optimum output power of an 11-element linear array processor as the look direction θ_0 is swept from 0 to 180 degrees. The element self noise is -30 dB and a 0 dB source is located at $\theta_s = 90^\circ$. Note from the plot of Fig. 2.6 that the response of the optimum processor for a look direction of $\theta_0 = 90^\circ$ (i.e., at the source) is a single point at approximately 0 dB and that as the processor is optimised for look directions in the near vicinity of the source, significant attenuation of the source signal results. In signal reception applications, an array with a response such as this may actually suppress the target signal if there are errors in the assumed direction of arrival of the signal. In spatial estimation the number of separate bins that the bearing space must be divided into would have to be very large to reduce the risk of missing a signal source in a particular direction.

2.3.2 Derivative Constraints

A number of methods have been suggested to increase the spatial acceptance angle of the LCMV and MVDR processors in order to make them more robust* to this type of error. The method we considered in this thesis uses spatial derivative constraints.

For an nth order maximally flat spatial power response in the array look direction, (θ_0, ϕ_0), the following must hold

$$\left.\frac{\partial^r \rho(\theta,\phi)}{\partial\theta^m \partial\phi^{r-m}}\right|_{(\theta_0,\phi_0)} = 0 \;,\; \begin{cases} r = 1,2,\ldots,n \\ m = 0,1,\ldots,r \end{cases} \tag{2-35}$$

where $\rho(\theta,\phi) \in \mathcal{R}$, the array power response, is given by

$$\rho(\theta) = H^*(\theta)H(\theta) \tag{2-36}$$

and $H(\theta)$ is defined by (2-29). The superscript, *, represents the complex conjugate.

In practice only the first and second order derivatives are considered as imposing higher order constraints not only results in a very complex non-linear optimisation problem but reduces the noise rejection capabilities of the processor [12]. The first and second order derivatives can be easily obtained from (2-35) above:

---

* The term *robust*, which we will be using throughout this thesis, is used to describe the sensitivity of an algorithm or technique to factors which interfere with its performance. For example, in the context of the current discussion, an array processor which is relatively unaffected by slight mismatches between the source direction and the array look direction is said to be robust to directional mismatch.

First Order

$$\left.\frac{\partial \rho(\theta,\phi)}{\partial \theta}\right|_{(\theta_0,\phi_0)} = 0 \tag{2-37}$$

$$\left.\frac{\partial \rho(\theta,\phi)}{\partial \phi}\right|_{(\theta_0,\phi_0)} = 0 \tag{2-38}$$

It follows from (2-29) and (2-36) that

$$\frac{\partial \rho}{\partial \theta} = H\frac{\partial H^*}{\partial \theta} + H^*\frac{\partial H}{\partial \theta} = \boldsymbol{w}_c^H \dot{\boldsymbol{s}}_\theta \boldsymbol{s}^H \boldsymbol{w}_c + \boldsymbol{w}_c^H \boldsymbol{s}\dot{\boldsymbol{s}}_\theta^H \boldsymbol{w}_c \tag{2-39}$$

and

$$\frac{\partial \rho}{\partial \phi} = H\frac{\partial H^*}{\partial \phi} + H^*\frac{\partial H}{\partial \phi} = \boldsymbol{w}_c^H \dot{\boldsymbol{s}}_\phi \boldsymbol{s}^H \boldsymbol{w}_c + \boldsymbol{w}_c^H \boldsymbol{s}\dot{\boldsymbol{s}}_\phi^H \boldsymbol{w}_c \tag{2-40}$$

where $\boldsymbol{s} = \boldsymbol{s}(\theta,\phi)$ and

$$\begin{aligned} \dot{\boldsymbol{s}}_\theta &= \frac{\partial \boldsymbol{s}(\theta,\phi)}{\partial \theta}, \\ \dot{\boldsymbol{s}}_\phi &= \frac{\partial \boldsymbol{s}(\theta,\phi)}{\partial \phi} \end{aligned} \tag{2-41}$$

Second Order

$$\left.\frac{\partial^2 \rho(\theta,\phi)}{\partial \theta^2}\right|_{(\theta_0,\phi_0)} = 0 \tag{2-42}$$

$$\left.\frac{\partial^2 \rho(\theta,\phi)}{\partial \phi^2}\right|_{(\theta_0,\phi_0)} = 0 \tag{2-43}$$

$$\left.\frac{\partial^2 \rho(\theta,\phi)}{\partial \theta \partial \phi}\right|_{(\theta_0,\phi_0)} = 0 \tag{2-44}$$

It follows from (2-29) and (2-36) that

$$\begin{aligned} \frac{\partial^2 \rho}{\partial \theta^2} &= H\frac{\partial^2 H^*}{\partial \theta^2} + 2\frac{\partial H}{\partial \theta}\frac{\partial H^*}{\partial \theta} + H^*\frac{\partial^2 H}{\partial \theta^2} \\ &= \boldsymbol{w}_c^H \ddot{\boldsymbol{s}}_\theta \boldsymbol{s}^H \boldsymbol{w}_c + 2\boldsymbol{w}_c^H \dot{\boldsymbol{s}}_\theta \dot{\boldsymbol{s}}_\theta^H \boldsymbol{w}_c + \boldsymbol{w}_c^H \boldsymbol{s}\ddot{\boldsymbol{s}}_\theta^H \boldsymbol{w}_c \end{aligned} \tag{2-45}$$

$$\frac{\partial^2 \rho}{\partial \phi^2} = H\frac{\partial^2 H^*}{\partial \phi^2} + 2\frac{\partial H}{\partial \phi}\frac{\partial H^*}{\partial \phi} + H^*\frac{\partial^2 H}{\partial \phi^2}$$
$$= \boldsymbol{w}_c^H \ddot{\boldsymbol{s}}_\phi \boldsymbol{s}^H \boldsymbol{w}_c + 2\boldsymbol{w}_c^H \dot{\boldsymbol{s}}_\phi \dot{\boldsymbol{s}}_\phi^H \boldsymbol{w}_c + \boldsymbol{w}_c^H \boldsymbol{s}\ddot{\boldsymbol{s}}_\phi^H \boldsymbol{w}_c \tag{2-46}$$

and

$$\frac{\partial^2 \rho}{\partial \theta \partial \phi} = H\frac{\partial^2 H^*}{\partial \theta \partial \phi} + \frac{\partial H}{\partial \theta}\frac{\partial H^*}{\partial \phi} + \frac{\partial H}{\partial \phi}\frac{\partial H^*}{\partial \theta} + H^*\frac{\partial^2 H}{\partial \theta \partial \phi}$$
$$= \boldsymbol{w}_c^H \ddot{\boldsymbol{s}}_{\theta\phi} \boldsymbol{s}^H \boldsymbol{w}_c + \boldsymbol{w}_c^H \dot{\boldsymbol{s}}_\theta \dot{\boldsymbol{s}}_\phi^H \boldsymbol{w}_c + \boldsymbol{w}_c^H \dot{\boldsymbol{s}}_\phi \dot{\boldsymbol{s}}_\theta^H \boldsymbol{w}_c + \boldsymbol{w}_c^H \boldsymbol{s}\ddot{\boldsymbol{s}}_{\theta\phi}^H \boldsymbol{w}_c \tag{2-47}$$

where

$$\ddot{\boldsymbol{s}}_\theta = \frac{\partial^2 \boldsymbol{s}(\theta,\phi)}{\partial \theta^2},$$
$$\ddot{\boldsymbol{s}}_\phi = \frac{\partial^2 \boldsymbol{s}(\theta,\phi)}{\partial \phi^2}, \tag{2-48}$$
$$\ddot{\boldsymbol{s}}_{\theta\phi} = \frac{\partial^2 \boldsymbol{s}(\theta,\phi)}{\partial \theta\, \partial \phi}.$$

From (2-31), (2-39), (2-40) and (2-45) to (2-47), the optimisation problem resulting from the use of additional first and second order NS derivative constraints can be summarised as follows:

$$\min_{\boldsymbol{w}_c} \quad \boldsymbol{w}_c^H \boldsymbol{R} \boldsymbol{w}_c$$
$$\text{subject to} \quad \boldsymbol{s}^H \boldsymbol{w}_c = e^{j\gamma} \tag{2-49}$$
$$\boldsymbol{w}_c^H \boldsymbol{A}_i \boldsymbol{w}_c = 0, \; i = 1,2,\ldots,5$$

where

$$\boldsymbol{A}_1 = \left(\dot{\boldsymbol{s}}_\theta \boldsymbol{s}^H + \boldsymbol{s}\dot{\boldsymbol{s}}_\theta^H\right)\Big|_{(\theta=\theta_0,\phi=\phi_0)},$$
$$\boldsymbol{A}_2 = \left(\dot{\boldsymbol{s}}_\phi \boldsymbol{s}^H + \boldsymbol{s}\dot{\boldsymbol{s}}_\phi^H\right)\Big|_{(\theta=\theta_0,\phi=\phi_0)},$$
$$\boldsymbol{A}_2 = \left(\ddot{\boldsymbol{s}}_\theta \boldsymbol{s}^H + 2\dot{\boldsymbol{s}}_\theta \dot{\boldsymbol{s}}_\theta^H + \boldsymbol{s}\ddot{\boldsymbol{s}}_\theta^H\right)\Big|_{(\theta=\theta_0,\phi=\phi_0)}, \tag{2-50}$$
$$\boldsymbol{A}_4 = \left(\ddot{\boldsymbol{s}}_\phi \boldsymbol{s}^H + 2\dot{\boldsymbol{s}}_\phi \dot{\boldsymbol{s}}_\phi^H + \boldsymbol{s}\ddot{\boldsymbol{s}}_\phi^H\right)\Big|_{(\theta=\theta_0,\phi=\phi_0)},$$
$$\boldsymbol{A}_5 = \left(\ddot{\boldsymbol{s}}_{\theta\phi} \boldsymbol{s}^H + \dot{\boldsymbol{s}}_\theta \dot{\boldsymbol{s}}_\phi^H + \dot{\boldsymbol{s}}_\phi \dot{\boldsymbol{s}}_\theta^H + \boldsymbol{s}\ddot{\boldsymbol{s}}_{\theta\phi}^H\right)\Big|_{(\theta=\theta_0,\phi=\phi_0)}.$$

In Appendix F we show that the optimum output power of (2-49) is independent of γ and that the optimum weight vector, for different choices of γ, are related by

$$\boldsymbol{w}_{c(opt)}(\gamma_1) = \boldsymbol{w}_{c(opt)}(\gamma_2)e^{j(\gamma_1-\gamma_2)} \tag{2-51}$$

where $\boldsymbol{w}_{c(opt)}(\gamma)$ is the optimum weight vector for a given choice of γ. Without loss of generality we choose $\gamma = 0$ and reformulate the optimisation problem of (2-49) as follows

$$\begin{aligned} \min_{\boldsymbol{w}_c} \quad & \boldsymbol{w}_c^H \boldsymbol{R} \boldsymbol{w}_c \\ \text{subject to} \quad & \boldsymbol{s}^H \boldsymbol{w}_c = 1 \\ & \boldsymbol{w}_c^H \boldsymbol{A}_i \boldsymbol{w}_c = 0, \; i = 1,2,\ldots,5 \end{aligned} \tag{2-52}$$

which is the MVDR problem of (2-33) but with additional quadratic constraints. In [13] and [14] it is shown that by substituting the look direction constraint into the quadratic constraints, the constraint structure in (2-52) can be simplified somewhat. The conditions that are both necessary and sufficient (NS) for the first order derivatives in (2-37) and (2-38) to be zero now become

$$\left.\frac{\partial \rho(\theta,\phi)}{\partial \theta}\right|_{(\theta_0,\phi_0)} = 0 \Leftrightarrow \mathrm{Re}\left(\dot{\boldsymbol{s}}_{\theta_0}^H \boldsymbol{w}_c\right) = 0, \tag{2-53}$$

and

$$\left.\frac{\partial \rho(\theta,\phi)}{\partial \phi}\right|_{(\theta_0,\phi_0)} = 0 \Leftrightarrow \mathrm{Re}\left(\dot{\boldsymbol{s}}_{\phi_0}^H \boldsymbol{w}_c\right) = 0, \tag{2-54}$$

and the NS conditions for the second order derivatives of (2-42) to (2-44) to be zero are

$$\left.\frac{\partial^2 \rho(\theta,\phi)}{\partial \theta^2}\right|_{(\theta_0,\phi_0)} = 0 \Leftrightarrow \mathrm{Re}\left(\ddot{\boldsymbol{s}}_{\theta_0}^H \boldsymbol{w}_c\right) + \left[\mathrm{Im}\left(\dot{\boldsymbol{s}}_{\theta_0}^H \boldsymbol{w}_c\right)\right]^2 = 0, \tag{2-55}$$

$$\left.\frac{\partial^2 \rho(\theta,\phi)}{\partial \phi^2}\right|_{(\theta_0,\phi_0)} = 0 \Leftrightarrow \mathrm{Re}\left(\ddot{\boldsymbol{s}}_{\phi_0}^H \boldsymbol{w}_c\right) + \left[\mathrm{Im}\left(\dot{\boldsymbol{s}}_{\phi_0}^H \boldsymbol{w}_c\right)\right]^2 = 0, \tag{2-56}$$

$$\left.\frac{\partial^2 \rho(\theta,\phi)}{\partial \theta \, \partial \phi}\right|_{(\theta_0,\phi_0)} = 0 \Leftrightarrow \mathrm{Re}\left(\ddot{\boldsymbol{s}}_{\theta_0\phi_0}^H \boldsymbol{w}_c\right) + \mathrm{Im}\left(\dot{\boldsymbol{s}}_{\theta_0}^H \boldsymbol{w}_c\right)\mathrm{Im}\left(\dot{\boldsymbol{s}}_{\phi_0}^H \boldsymbol{w}_c\right) = 0, \tag{2-57}$$

The constraint expressions in (2-53), (2-54) and (2-55) to (2-57) constitute NS conditions on the processor weights for a second order maximally flat response in the look direction and are often referred to as the first and second order NS derivative constraints respectively.

On inspection of the second order NS constraints of (2-55) to (2-57), one can see that a complex, non-linear optimisation problem results. To avoid this difficulty a number of linear approximations to the constraints have been suggested in the literature [13, 15]. While these approximations allow a much simpler linear optimisation problem to be solved, they represent only sufficient conditions for a maximally flat response. In practice they have been found to depend on the choice of array phase centre [15] and, on occasion, these constraints may not even be consistent [16].

It is necessary, therefore, to reduce the original optimum problem to a simpler form so that the solution might be found more efficiently. In [1], Thng, Cantoni and Leung present such a method and the next section summarises this method as it relates to the narrowband array processing problem.

In the remainder of this thesis we will consider only a 2D scenario where the array elements and signal sources are assumed to lie on a plane. Not only does this simplify the notation but it allows the concepts being introduced to be seen more clearly. The results, however, can be readily extended to the 3D scenario.

2.4 Optimum, NS Derivative Constrained Array Processor

In the 2D case the optimisation problem is found from (2-52) but using only the first and second order derivative constraints of (2-53) and (2-55). The result is the quadratically constrained optimisation problem summarised below

$$\min_{\boldsymbol{w}_c} P(\boldsymbol{w}_c) \tag{2-58}$$

subject to

$$\boldsymbol{s}^H(\theta_0)\boldsymbol{w}_c = 1, \tag{2-59}$$

$$\mathrm{Re}\left(\dot{\boldsymbol{s}}^H(\theta_0)\boldsymbol{w}_c\right) = 0, \tag{2-60}$$

and

$$\mathrm{Re}\left(\ddot{\boldsymbol{s}}^H(\theta_0)\boldsymbol{w}_c\right) + \left[\mathrm{Im}\left(\dot{\boldsymbol{s}}^H(\theta_0)\boldsymbol{w}_c\right)\right]^2 = 0. \tag{2-61}$$

Noting that the terms $\left[\mathrm{Re}\left(\ddot{\boldsymbol{s}}^H(\theta_0)\boldsymbol{w}_c\right)\right]$ and $\left[\mathrm{Im}\left(\dot{\boldsymbol{s}}^H(\theta_0)\boldsymbol{w}_c\right)\right]$ in (2-61) are scalars and making the assignment

$$\left[\mathrm{Im}\left(\dot{\boldsymbol{s}}^{H}(\theta_0)\boldsymbol{w}_c\right)\right]=\alpha, \tag{2-62}$$

such that

$$\left[\mathrm{Re}\left(\ddot{\boldsymbol{s}}^{H}(\theta_0)\boldsymbol{w}_c\right)\right]=-\alpha^2, \tag{2-63}$$

where $\alpha \in \mathcal{R}$ is some unspecified scalar, it can be shown [1] that the quadratically constrained optimisation problem of (2-58) to (2-61) can be re-formulated as an equivalent two-stage minimisation problem.

Before the problem can be reformulated, however, we note that the first and second order derivative constraints of (2-60) and (2-61) are not linear in the complex field since they involve taking the real and imaginary parts. In [13] Tseng notes that one can *linearise* the constraints in this regard by simply expressing them in real notation with double dimension as discussed in Section 2.2. In [2] it is shown that, for the real beamforming system, the mean output power of the array processor is equivalent for real notation and complex notation. Since we are minimising the array mean output power, the optimum beamformers resulting from the use of real and complex notations are equivalent. In real notation, then, the first stage of the two-stage minimisation is a linearly constrained optimisation problem, parameterised by α, and the second stage is an unconstrained optimisation of α:

$$\min_{\alpha}\left[\begin{array}{ll} \min\limits_{\boldsymbol{w}} P(\boldsymbol{w}) \\ \text{subject to} \quad \boldsymbol{C}^T\boldsymbol{w}=\boldsymbol{h}(\alpha) \end{array}\right] \tag{2-64}$$

where $\boldsymbol{C} \in \mathcal{R}^{2L\times M}$ is a stacked constraint matrix consisting of M columns of linear constraints formed from (2-59), (2-60) and (2-61) and $\boldsymbol{h}(\alpha) \in \mathcal{R}^{M}$ is the corresponding constraint vector. The constraint matrix $\boldsymbol{C}$ and constraint vector $\boldsymbol{h}(\alpha)$ are given, respectively, in expanded form below.

$$\boldsymbol{C}^T=\left[\begin{array}{ll} \left[\mathrm{Re}\left(\boldsymbol{s}_{\theta_0}\right)\right]^T, & \left[\mathrm{Im}\left(\boldsymbol{s}_{\theta_0}\right)\right]^T \\ -\left[\mathrm{Im}\left(\boldsymbol{s}_{\theta_0}\right)\right]^T, & \left[\mathrm{Re}\left(\boldsymbol{s}_{\theta_0}\right)\right]^T \\ \left[\mathrm{Re}\left(\dot{\boldsymbol{s}}_{\theta_0}\right)\right]^T, & \left[\mathrm{Im}\left(\dot{\boldsymbol{s}}_{\theta_0}\right)\right]^T \\ -\left[\mathrm{Im}\left(\dot{\boldsymbol{s}}_{\theta_0}\right)\right]^T, & \left[\mathrm{Re}\left(\dot{\boldsymbol{s}}_{\theta_0}\right)\right]^T \\ \left[\mathrm{Re}\left(\ddot{\boldsymbol{s}}_{\theta_0}\right)\right]^T, & \left[\mathrm{Im}\left(\ddot{\boldsymbol{s}}_{\theta_0}\right)\right]^T \end{array}\right] \tag{2-65}$$

and

$$\boldsymbol{h}(\alpha)=\begin{bmatrix}1\\0\\0\\\alpha\\-\alpha^2\end{bmatrix}. \tag{2-66}$$

Using the method of Lagrange multipliers, the inner stage of the two-stage minimisation problem of (2-64) has the optimum solution

$$\boldsymbol{w}_{opt}(\alpha)=\boldsymbol{R}^{-1}\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{h}(\alpha). \tag{2-67}$$

Note that the optimum solution depends on α and the optimum α is yet to be determined. By substituting $\boldsymbol{w}_{opt}(\alpha)$ into the expression for $P(\boldsymbol{w})$ in (2-64) the outer stage of the two-stage minimisation problem is now given by

$$\min_{\alpha} P(\alpha)=\boldsymbol{h}^T(\alpha)\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{h}(\alpha). \tag{2-68}$$

It can be verified that $P(\alpha)$ is a quartic polynomial in α [13]. Thus given $\boldsymbol{R}$ and $\boldsymbol{C}$ the optimum α can be found by computing the roots of a cubic polynomial. Substituting the optimum α into (2-67) gives, finally, the optimum weight vector $\boldsymbol{w}_{opt}$.

It is important to note at this point that since the optimum weight vector is found from the minimisation of a quartic polynomial, it is possible to have two optimum solutions, i.e., the feasible region of the constraints is not convex [17] in general and local minima and multiple optimal solutions can occur. In the case where two optimum weight vectors do occur one can simply be chosen arbitrarily over the other since they are both equally valid solutions.

As a comparison with the MVDR beamformer shown in Fig. 2.6, the plot in Fig 2.7 shows the optimum output power of the array processor with the additional first and second order NS derivative constraints as the look direction is swept from 0° to 360°. For ease of reference the plot of Fig. 2.6 is also shown in Fig. 2.7 by the dashed line.

The array structure and source scenarios are the same in both examples. Note the flat section in the solid plot for look directions around 90° (i.e. at the 0 dB source) resulting from the use of the derivative constraints. One can see that suppression of the desired signal will not occur for small differences between the look direction and the actual direction of arrival of the desired signal. Note also the reduction in noise rejection at look directions away from the source.

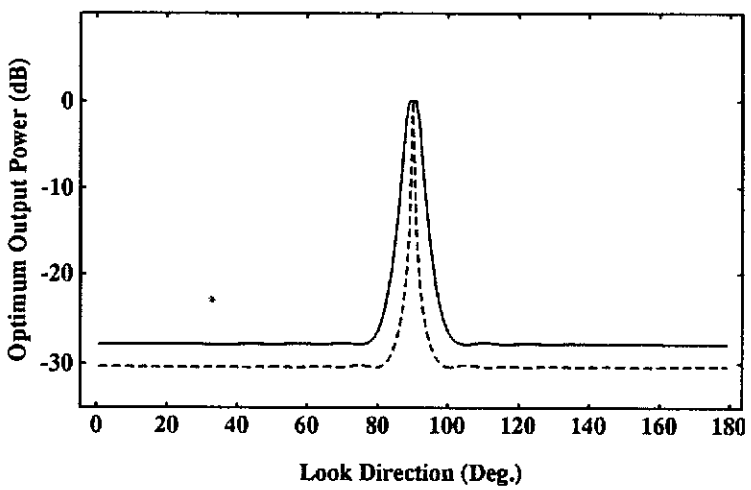


Figure 2.7 Derivative Constrained Array Processor Optimum Output Power as the Look Direction is Stepped form 0° to 360° in 1° Increments
Solid Line: Array Processor with 2nd order NS Derivative Constraints
Dashed Line: MVDR Array Processor

2.5 References

[1] I. Thng, A. Cantoni and Y. H. Leung, "Constraints for Maximally Flat Optimum Broadband Antenna Arrays", *IEEE Trans. Sig. Proc.*, vol. SP-43, no. 6, pp. 1334-1347, June 1995.

[2] L. C. Godara and A. Cantoni, "Signal Representation for Array Processing", Tech. Report EE8203, Dept. of Electrical and Computer Engineering, University of Newcastle, NSW, 2308, Australia, August 1982

[3] S. Haykin, *Communication Systems*, 2nd ed., Wiley, New York, 1983.

[4] J. E. Hudson, *Adaptive Array Principles*, Peter Peregrinus, London, 1981.

[5] E. Nicolau and D. Zaharia, *Adaptive Arrays*, Elsevier, Amsterdam, 1989.

[6] R. T. Crompton, Jr, *Adaptive Antennas: Concepts and Performance*, Prentice-Hall, Englewood Cliffs, 1988.

[7] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1991.

[8] O. L. Frost, III, "An Adaptive Algorithm for Linearly Constrained Adaptive Array Processing", *IEEE Proceedings*, vol. 60, no. 8, August 1972.

[9] D.Nunn, "Performance Assessments of a Time-Domain Adaptive Processor in a Broadband Environment", *IEE Proc.*, vol. 130, pts. F and H, no. 1, Feb. 1983, pp. 139-146.

[10] D. Nunn, "Suboptimal Frequency-Domain Adaptive Antenna Processing Algorithm for Broadband Environments", *IEE Proc.*, vol. 134, pt. F, no. 4, pp. 341-351, July 1987.

[11] A. K. Steele, "Comparison of Directional and Derivative Constraints for Beamformers Subject to Multiple Linear Constraints", *IEE Proc. F, Commun. Radar Signal Processing.*, and *IEE Proc. H, Microwaves, Opt. and Antennas*, (joint issue), vol. 130, pts. F and H, (1), pp. 41-45, 1983.

[12] M. H. Er and A. Cantoni, "Derivative Constraints for Broad-Band Element Space Antenna Array Processors", *IEEE Trans. Acoustics, Speech and Signal Proc.*, vol. 31, no. 6, pp 1378-1393, December 1983.

[13] C. Y. Tseng, "Minimum Variance Beamforming with Phase-Independent Derivative Constraints", *IEEE Trans. Antennas Propagat.*, vol. 40, no. 3, pp. 285-294, March 1992.

[14] M. H. Er, "Adaptive Antenna Array Under Directional and Spatial Derivative Constraints", *IEE Proc.*, vol. 135, pt. H, no. 6, pp. 414-419, December 1988.

[15] K. M. Buckley and L. J. Griffiths, "An Adaptive Sidelobe Canceller with Derivative Constraints", *IEEE Trans. Antennas Propagat.*, vol. 34, no. 3, pp. 311-319, March 1986.

[16] I. Thng, A. Cantoni and Y. H. Leung, "A New Set of Constraints for Maximally Flat Optimum Broadband Antenna Arrays", *Proc. IEEE ICASSP*, vol. 4, pp 533-536, Adelaide, Australia, April 1994.

[17] P. E. Gill, W. Murray and M. H. Wright, *Numerical Linear Algebra and Optimisation*, vol. 1, Addison-Wesley Pub. Co., Redwood City, Calif., 1991.

3. *The Adaptive Problem and Some Preliminary Approaches*

In the previous chapter, we found that an array processor can be made less susceptible to pointing errors by imposing constraints on the processor weights that force some of the spatial derivatives of the power response in the look direction to be zero.

In the first part of this chapter, we focus on the problem central to this thesis, i.e., can we find efficient and numerically robust methods to implement the look direction plus first and second order NS derivative constraints in an adaptive array processor. To begin with, we look at estimating functions for the input signal correlation matrix and its inverse and how these might be applied to our problem. In the final part of this section, we state more clearly the objective of the thesis by identifying the key components that a solution to the problem must contain.

The next two sections outline two different approaches to the problem. The first involves breaking the two-stage minimisation problem down further into a number of concurrent algorithms based on Frost's computationally simple constrained LMS algorithm. The second method employs the matrix inversion lemma to adaptively estimate the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$, a key to solving the optimum problem.

Computer studies have, however, highlighted some shortcomings in these methods which prohibit their use as practical algorithms. Nevertheless, they still give some insights to the adaptive, quadratically constrained array processing problem and provide a starting point for the development of better methods.

3.1 The Adaptive Array Problem Statement

3.1.1 Adaptive Narrowband Array Processing

In Section 2.4, we found that to design an optimum second order NS derivative constrained narrowband antenna array processor we are faced with the two-stage optimisation problem defined by (2-56) and (2-57) which we present again below for convenience:

$$\min_{\alpha}\left[\begin{array}{c}\min_{\boldsymbol{w}} P(\boldsymbol{w}) \\ \text{subject to} \quad \boldsymbol{C}^T\boldsymbol{w}=\boldsymbol{h}(\alpha)\end{array}\right]. \tag{3-1}$$

It was shown that the optimum solution to this problem is given by

$$\boldsymbol{w}_{opt}(\alpha)=\boldsymbol{R}^{-1}\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{h}(\alpha) \tag{3-2}$$

where the optimum α is found by locating the global minimum of the quartic polynomial

$$\min_{\alpha} P(\alpha)=\boldsymbol{h}^T(\alpha)\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{h}(\alpha) \tag{3-3}$$

Equations (3-2) and (3-3) allow us to solve the optimum problem where the input signal statistics and hence the input correlation matrix are assumed to be known *a priori*. In practice, however, the input signal statistics are rarely known and the correlation matrix must be estimated from the input data. Therefore, we wish to find some adaptive algorithm for estimating the optimum weight vector for the two-stage minimisation problem of (3-1) given the input data available. Recall in the Introduction that by the term *adaptive algorithm* we generally imply: (i) a recursive algorithm that starts from pre-defined initial conditions with usually little or no knowledge of the signal environment; (ii) that in a stationary input signal environment, the weight vector produced by the algorithm converges toward the optimum solution; and (iii) in a non-stationary environment, that the algorithm will respond to slow changes in the input signal statistics, i.e. track slow changes in the input signal environment. Essentially then, we want to design an antenna array processor which maintains unity gain and second order flatness in the power response in the look

direction of the array to ensure that the target signal is received but also to *learn* the signal environment and adaptively null any interference signals from other directions.

There are numerous ways of estimating the correlation matrix. Estimating functions may produce either a biased or an unbiased estimate [1, 2] depending on how they use the data available. For narrowband array processing, an unbiased estimate of the correlation matrix can be found simply by averaging a number of instantaneous values, or snapshots, of the correlation matrix, i.e.,

$$\boldsymbol{R}(N) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}(i)\boldsymbol{x}^T(i). \tag{3-4}$$

Note that in (3-4) above we are now considering sampled versions of the input vector given by (2-19).

In a non-stationary environment, the estimated correlation matrix computed as above will soon become outdated and no longer represent a good estimate of the true matrix. One might, therefore, discard the correlation matrix at regular intervals and repeat equation (3-4) using new blocks of input data. Alternatively, a method known as *exponential weighting* [9] can be used which provides an unbiased estimate of the correlation matrix at each sample instant according to the recursive relation

$$\boldsymbol{R}(n+1) = \lambda \boldsymbol{R}(n) + \eta \boldsymbol{x}(n+1)\boldsymbol{x}^T(n+1) \tag{3-5}$$

where $0 << \lambda < 1$ and $\eta = 1 - \lambda$. The weighting factor λ is known as the *forgetting factor* and ensures that data in the distant past are forgotten thus allowing the algorithm to adapt to slow changes in the input signal statistics. The value of $1/\eta$ is a measure of the memory of the algorithm. For a value of $\lambda = 1$ the algorithm has infinite memory and will not respond to any new input data for n large, while for $\lambda = 0$ the algorithm has no memory and the estimated correlation matrix becomes the instantaneous value, i.e.,

$$\boldsymbol{R}(n+1) = \boldsymbol{x}(n+1)\boldsymbol{x}^T(n+1). \tag{3-6}$$

(This is the input signal correlation matrix estimator used in the well known Least-Mean-Square (LMS) algorithm [9, ch. 9].)

On inspection of the expression for the optimum weight vector in (3-2) one can see that we are not really interested in the correlation matrix itself but rather its inverse. If the time average of (3-4) is used to generate a new estimate of the correlation matrix every N samples then the inverse must also be computed every N samples. This forms the basis of techniques such as *Sample Matrix Inversion (SMI)* [3] and *Estimate and Plug* [4].

An alternative method of estimating the inverse correlation matrix can be derived using (3-5) and the matrix inverse identity known as *Woodbury's identity* or the *Matrix Inversion Lemma* [9, p. 480]. This Lemma states that for an $M \times M$ positive definite matrix $\boldsymbol{A}$, an $N \times N$ positive definite matrix $\boldsymbol{C}$ and two $M \times N$ matrices $\boldsymbol{B}$ and $\boldsymbol{D}$ the following holds

$$\left(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{C}\boldsymbol{D}^T\right)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{C}^{-1}+\boldsymbol{D}^T\boldsymbol{A}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{D}^T\boldsymbol{A}^{-1}. \tag{3-7}$$

Substituting (3-5) into (3-7), we obtain a recursive update for the estimated inverse correlation matrix

$$\boldsymbol{R}^{-1}(n+1) = \frac{1}{\lambda}\left[\boldsymbol{R}^{-1}(n) - \frac{\boldsymbol{R}^{-1}(n)\boldsymbol{x}(n+1)\boldsymbol{x}^T(n+1)\boldsymbol{R}^{-1}(n)}{\frac{\lambda}{\eta}+\boldsymbol{x}^T(n+1)\boldsymbol{R}^{-1}(n)\boldsymbol{x}(n+1)}\right] \tag{3-8}$$

which forms the basis of the well-known recursive least squares (RLS) algorithm [9].

An adaptive algorithm, then, might be found if we use one of the above estimators in place of the true inverse correlation matrix in equations (3-2) and (3-3), i.e.,

$$\alpha_{opt}(n) = \min_{\alpha}\left[\boldsymbol{h}^T(\alpha)\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}\boldsymbol{h}(\alpha)\right] \tag{3-9}$$

$$\boldsymbol{w}_{opt}(n) = \boldsymbol{R}^{-1}(n)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}\boldsymbol{h}\left(\alpha_{opt}(n)\right). \tag{3-10}$$

In Section 2.4, we pointed out that the optimisation problem arising from the use of second order NS derivative constraints (equation (3-1)) is multimodal, i.e., the feasible region of the constraints is non-convex in general. In the case of an optimum processor this does not pose a problem as, in the worst case of two optimum solutions, one can simply be chosen arbitrarily over the other. In an adaptive application, however, the question of multiple optimum solutions becomes a more complex matter.

Consider the scenario where one optimum set of weights minimises the array mean output power while satisfying the constraint equations and a second set of weights results in the same or nearly the same minimum mean output power while also satisfying the constraints. An adaptive method which is trying to locate an optimum weight vector based only on an estimate of the array mean output power has no way of deciding *a priori* which optimum weight vector to move towards. Additionally, in this case of two optimum weight vectors, noise in the correlation matrix may cause the adaptive algorithm to switch between optimum weight vectors as it converges.

3.1.2 Problem Statement

In the previous section, we introduced the idea of using an estimate of the input signal correlation matrix based on recent samples of the input vector to estimate the optimum weight vector thus allowing the array to adapt to a non-stationary input environment.

In essence, the problem addressed in this thesis is to find an adaptive algorithm which implements the two-stage minimisation of (3-1). In addition we are interested in some of the conventional performance measures of an adaptive algorithm such as rate of convergence, tracking, misadjustment, computational complexity and numerical robustness. In the case of multiple optimum solutions, as discussed in the last part of the previous section, we need to investigate the effect, if any, of switching between optimum weight vector solutions.

Now, from (3-2) and (3-3), we see that an adaptive algorithm for estimating the optimum weight vector, $\boldsymbol{w}_{opt}$, must contain three major components:

1. A method of estimating the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ so that the coefficients of the quartic polynomial $P(\alpha)$ can be found;
2. A method to find the roots of a cubic polynomial so that the global minimum of $P(\alpha)$ can be located; and
3. A method to estimate the inverse correlation matrix $\boldsymbol{R}^{-1}$ in order that the new weight vector may be determined.

3.2 The Method of Elementary Weights

In this section, a method based on Frost's linearly constrained LMS algorithm [5] is proposed for implementing the adaptive version of the two-stage minimisation problem. For completeness, a summary of Frost's algorithm is provided in Appendix A.

3.2.1 Derivation

We begin the derivation of the method of elementary weights by orthonormalising the $2L \times M$ constraint matrix $\boldsymbol{C}$ in (3-1) thus producing an orthonormal set of M column vectors. This can be accomplished by using the modified Gram-Schmidt process as described in [6] and [7]. For the constraint equation to remain unchanged, the column operations performed on the constraint matrix $\boldsymbol{C}$ must also be performed on the elements of the constraint vector $\boldsymbol{h}(\alpha)$. If the original constraint vector have K non-zero elements (see Equation 2-58) then the resulting constraint vector will also have K non-zero elements which are functions of the parameter α. We now re-arrange the elements of the constraint vector so that all of the non-zero elements are at the top. The columns of the constraint matrix are also re-ordered accordingly and we arrive at the new constraint equation

$$\boldsymbol{C}'^{T}\boldsymbol{w} = \boldsymbol{h}'(\alpha) \tag{3-11}$$

where $\boldsymbol{h}'(\alpha)$ has the structure

$$\boldsymbol{h}'(\alpha) = [g_1(\alpha), \quad g_2(\alpha), \quad g_3(\alpha), \quad 0, \quad 0]^T. \tag{3-12}$$

The functions $g_i(\alpha)$ result from the orthonormalisation of the constraint matrix. The orthonormal matrix $\boldsymbol{C}'$ now has the property

$$\boldsymbol{C}'^{T}\boldsymbol{C}' = \boldsymbol{I}_M. \tag{3-13}$$

Note from (3-12) that for a 2-D narrowband array, there are three non-zero elements, i.e., $K = 3$.

Substituting $\boldsymbol{C}'$ for $\boldsymbol{C}$ in (3-2), we obtain the following expression for the optimum weight vector

$$\boldsymbol{w}_{opt}(\alpha) = \boldsymbol{R}^{-1}\boldsymbol{C}'\left(\boldsymbol{C}'^{T}\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}\boldsymbol{h}'(\alpha). \tag{3-14}$$

and substituting $\boldsymbol{C}'$ for $\boldsymbol{C}$ in (3-3) the optimum value of α is found from

$$\min_{\alpha} P(\alpha) = \boldsymbol{h}'^{T}(\alpha)\left(\boldsymbol{C}'^{T}\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}\boldsymbol{h}'(\alpha). \tag{3-15}$$

Since the last $M-K$ elements of $\boldsymbol{h}'(\alpha)$ are zero, the optimum weight vector in (3-14) is completely defined by the first K columns of the matrix $\boldsymbol{R}^{-1}\boldsymbol{C}'\left(\boldsymbol{C}'^{T}\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}$ Let the first K columns of this matrix be represented by $\boldsymbol{W}_e$, i.e.,

$$\boldsymbol{W}_e = \left[\boldsymbol{R}^{-1}\boldsymbol{C}'\left(\boldsymbol{C}'^{T}\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}\right]_K. \tag{3-16}$$

In addition, the quartic polynomial in α, $P(\alpha)$ in (3-15), can be found from the first $K\times K$ submatrix of $\left(\boldsymbol{C}'^{T}\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}$. Let this matrix be represented by

$$\boldsymbol{\Psi} = \left[\left(\boldsymbol{C}'^{T}\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}\right]_{K\times K}. \tag{3-17}$$

Consider now the K linearly constrained optimisation problems

$$\begin{aligned} &\min_{\boldsymbol{w}_i} \boldsymbol{w}_i^{T}\boldsymbol{R}\boldsymbol{w}_i \\ \text{subject to}\quad &\boldsymbol{C}'^{T}\boldsymbol{w}_i = \boldsymbol{h}_i \\ \text{where}\quad &\left[\boldsymbol{h}_i\right]_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}, \\ \text{and}\quad &i \in Z^K, j \in Z^M. \end{aligned} \tag{3-18}$$

We define a set of K vectors $\boldsymbol{w}_{ei} \in \mathcal{R}^{2L}, i \in Z^K$, which we term vectors of *elementary weights*, such that

$$\boldsymbol{w}_{ei} = \boldsymbol{R}^{-1}\boldsymbol{C}'\left(\boldsymbol{C}'^{T}\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}\boldsymbol{h}_i. \tag{3-19}$$

It is clear from optimisation theory that $\boldsymbol{w}_{ei}$ optimises (3-18), and from (3-16) it is also clear that

$$\boldsymbol{W}_e = \left[\boldsymbol{w}_{e1},\ \boldsymbol{w}_{e2},\ \ldots\ \boldsymbol{w}_{eK}\right]. \tag{3-20}$$

To find the elementary weights adaptively, we start K concurrent Frost constrained LMS algorithms. The algorithms produce estimates of the elementary weights, and hence the columns of the $\boldsymbol{W}_e$ matrix, according to the following K update equations

$$\begin{aligned} \boldsymbol{w}_{ei}(n+1) &= \boldsymbol{P}\left[\boldsymbol{w}_{ei}(n) - \mu y_i(n)\boldsymbol{x}(n)\right] + \boldsymbol{w}_{0i} \\ & i \in Z^K \end{aligned} \tag{3-21}$$

where $\boldsymbol{P} \in \mathcal{R}^{2L\times 2L}$ is a projection matrix given by

$$\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{C}'\left(\boldsymbol{C}'^T\boldsymbol{C}'\right)^{-1}\boldsymbol{C}'^T = \boldsymbol{I} - \boldsymbol{C}'\boldsymbol{C}'^T, \tag{3-22}$$

$y_i(n)$ is given by

$$y_i(n) = \boldsymbol{w}_{ei}^T(n)\boldsymbol{x}(n), \tag{3-23}$$

and $\boldsymbol{w}_{0i} \in \mathcal{R}^{2L}$ is given by

$$\boldsymbol{w}_{0i} = \boldsymbol{C}'\left(\boldsymbol{C}'^T\boldsymbol{C}'\right)^{-1}\boldsymbol{h}_i = \boldsymbol{C}'\boldsymbol{h}_i. \tag{3-24}$$

Note that $\boldsymbol{P}$ and the $\boldsymbol{w}_{0i}$ can be pre-computed before the algorithms are started. The parameter μ in (3-21) controls the step size of the Frost algorithm and is normally chosen to satisfy

$$0 < \mu < \frac{2}{3\ \mathrm{tr}\ (\boldsymbol{R})}. \tag{3-25}$$

For a more detailed discussion on the choice of μ, the reader is referred to [5].

We can now get an estimate of the matrix $\boldsymbol{W}_e$ at time n by using the elementary weight vector estimates, $\boldsymbol{w}_{ei}(n)$, computed above (see (3-20)):

$$\boldsymbol{W}_e \cong \boldsymbol{W}_e(n) = \left[\boldsymbol{w}_{e1}(n),\quad \boldsymbol{w}_{e2}(n),\quad \cdots\quad \boldsymbol{w}_{eK}(n)\right] \tag{3-26}$$

If an estimate of the correlation matrix is also made, using, for example the time average estimate of (3-4) or the exponentially weighted estimate of (3-5), then an estimate, $\Psi(n)$, of the first $K\times K$ sub-matrix of $\left(\boldsymbol{C}'^T\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}$ can be computed as follows

$$\Psi(n) = \boldsymbol{C}_K'^T\boldsymbol{R}(n)\boldsymbol{W}_e(n) \cong \Psi \tag{3-27}$$

where $\boldsymbol{R}(n)$ is the estimated correlation matrix and $\boldsymbol{C}'_K$ is the first K columns of the constraint matrix $\boldsymbol{C}'$.

The matrix computed in (3-27) can now be used in (3-15) to find the coefficients of the quartic polynomial in α. To find the optimum α at time n, $\alpha_{opt}(n)$, the zeros of the first derivative of $P(\alpha)$ with respect to α need to be found. This can be done using conventional polynomial factorisation techniques. Once $\alpha_{opt}(n)$ has been determined it can be used along with the matrix $\boldsymbol{W}_e(n)$ in equation (3-14) to obtain an estimate of the optimum weight vector.

The method of elementary weights is summarised in Table 3.1.

Initialise:

1. Orthonormalise the constraint matrix $\boldsymbol{C}$ and re-order the constraint vector $\boldsymbol{h}(\alpha)$ so that the non-zero elements are at the top. This results in the new constraint matrix $\boldsymbol{C}'$ and constraint vector $\boldsymbol{h}'(\alpha)$ defined in (3-11).
2. Compute the projection matrix $\boldsymbol{P}$ in (3-22).
3. Compute the set of vectors vector $\boldsymbol{w}_{0i}$ in (3-24)
4. Initialise the correlation matrix and the elementary weight vectors:

$$\begin{aligned} \boldsymbol{R}(0) &= \delta \times \boldsymbol{I}_{2L} \\ \boldsymbol{w}_{ei}(0) &= \boldsymbol{w}_{0i}, i = 1,2,\ldots,K \end{aligned} \tag{3-28}$$

For each new sample $x(n)$ at time n:

1. Update the K elementary weights

For $i = 1$ to K do:

$$y_i(n) = \boldsymbol{w}_{ei}^T(n)\boldsymbol{x}(n) \tag{3-29}$$

$$\boldsymbol{w}_{ei}(n+1) = \boldsymbol{P}\left[\boldsymbol{w}_{ei}(n) - \mu y_i(n)\boldsymbol{x}(n)\right] + \boldsymbol{w}_{0i}$$

2. Update the estimated correlation matrix using, for example, the exponentially weighted update

$$\boldsymbol{R}(n) = \lambda \boldsymbol{R}(n-1) + \eta \boldsymbol{x}(n)\boldsymbol{x}^T(n) \tag{3-30}$$

3. Compute an estimate of the $K \times K$ sub-matrix of $\left(\boldsymbol{C}'^T \boldsymbol{R}^{-1} \boldsymbol{C}'\right)^{-1}$

$$\Psi(n) = \boldsymbol{C}_K'^T \boldsymbol{R}(n) \boldsymbol{W}_e(n) \tag{3-31}$$

4. Compute the coefficients of the polynomial $P'(\alpha) = \dfrac{d}{d\alpha} P(\alpha)$ from the elements of the matrix $\Psi(n)$ using (3-15).

5. Compute the roots of the polynomial $P'(\alpha)$ and identify the α that minimises $P(\alpha)$. Denote this α as $\alpha_{opt}(n)$.

6. Compute the new weight vector

$$w(n) = \boldsymbol{W}_e(n) \boldsymbol{h}'\left(\alpha_{opt}(n)\right) \tag{3-32}$$

Table 3.1 The Method of Elementary Weights.

3.2.2 Discussion

The key to the method of elementary weights lies in finding an estimate of the optimum weight vector using only a sub-matrix of $\left(\boldsymbol{C}'^T \boldsymbol{R}^{-1} \boldsymbol{C}'\right)^{-1}$. The method is computationally very efficient as this sub-matrix is estimated without requiring any matrix inversions explicitly. The $\boldsymbol{W}_e$ matrix is estimated at each iteration of the algorithm using K parallel computations which require order $(2L)^2$ floating point operations each. The correlation matrix can be updated in a number of different ways

as discussed earlier and the exponentially weighted update given in (3-30) is just one example.

Even though the method of elementary weights appears to be computationally efficient and theoretically sound, it did not prove to be a useful adaptive algorithm. The problem lies in the differences between the two correlation matrix estimates used in the algorithm. These differences result in a poor estimate of the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ (represented by $\boldsymbol{\Psi}(n)$ in Table 3.1) severely effecting the second stage of the two-stage minimisation.

Recall from the above derivation that the $\boldsymbol{\Psi}(n)$ matrix is obtained from the matrix of K elementary weights by pre-multiplying $\boldsymbol{W}_e(n)$ by an estimate of the correlation matrix and the orthonormalised constraint matrix, equation (3-31). The $\boldsymbol{W}_e(n)$ matrix is an estimate of the first K columns of the true $\boldsymbol{W}_e$ matrix and is updated by the K concurrent Frost algorithms. If we represent the effective inverse correlation matrix implicitly computed by the K Frost algorithms by $\hat{\boldsymbol{R}}^{-1}(n)$, then we have

$$\boldsymbol{W}_e(n) = \left[\hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{C}'\left(\boldsymbol{C}'^T\hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{C}'\right)^{-1}\right]_K. \tag{3-33}$$

Pre-multiplying $\boldsymbol{W}_e(n)$ by the estimated correlation matrix updated in Step 2 (equation (3-30)) we get

$$\boldsymbol{C}'^T\boldsymbol{R}(n)\boldsymbol{W}_e(n) = \left[\boldsymbol{C}'^T\boldsymbol{R}(n)\hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{C}'\left(\boldsymbol{C}'^T\hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{C}'\right)^{-1}\right]_{K\times K}. \tag{3-34}$$

If $\boldsymbol{R}(n)$ and $\hat{\boldsymbol{R}}(n)$ are computed in the same manner and are good estimates of the true correlation matrix then (3-34) should result in a good estimate of the first $K\times K$ block matrix of $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$. However, in the method of elementary weights the matrices $\boldsymbol{R}(n)$ and $\hat{\boldsymbol{R}}(n)$ are computed in different ways resulting in a very poor estimate of the true $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ matrix. In computer studies it has been found that the matrix $\boldsymbol{\Psi}(n)$ may not even be positive definite, a property which is essential to the numerical stability of the algorithm.

A method is therefore required which uses uniformly the same estimates of the correlation matrix throughout the algorithm. This leads us to the *Double Inversion* algorithm presented in the next section.

3.2.3 Computer Studies

The following simulation results demonstrate the characteristics of the method of elementary weights.

The simulation results in Figs. 3.2 and 3.3 show the instantaneous output power of the array at each sample instant as the processor adapts to the stationary signal environment. The calculation of the instantaneous output power is analogous to the *a priori* estimation error squared in a standard RLS algorithm since it is computed from the current input vector and the previous weight vector, i.e.,

$$P(n) = \left[y(n)\right]^2 = \left[\boldsymbol{w}^T(n-1)\boldsymbol{x}(n)\right]^2 \tag{3-35}$$

The simulation scenario assumes a narrowband, 5-element circular array as shown in Fig. 3.1.

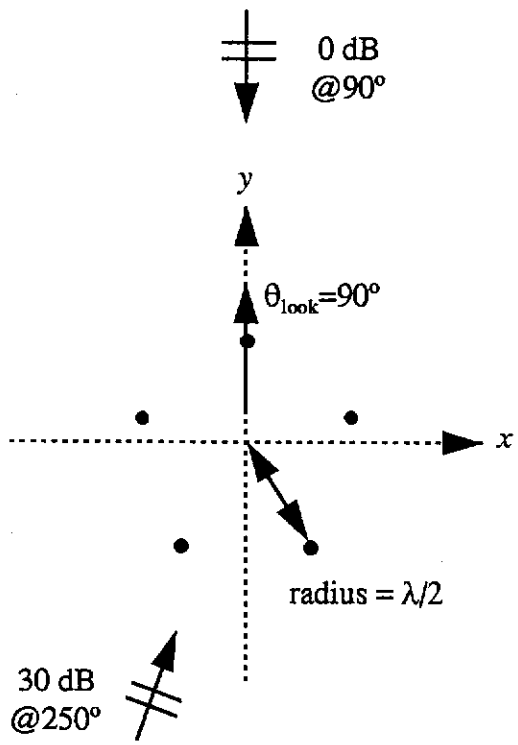


Figure 3.1 Array Structure and Source Scenario

Additional simulation parameters include:

- Forgetting factor, $\lambda = 0.98$
- Frost algorithm step size $\mu = 1.36 \times 10^{-5}$
- Initialisation constant, $\delta = 100$
- Source and interference signals modelled as zero mean, white Gaussian distributed sequences
- Element self noise = -20 dB (also modelled as zero-mean, whit Gaussian distributed signal)
- Array phase centre located at the centre of the circle

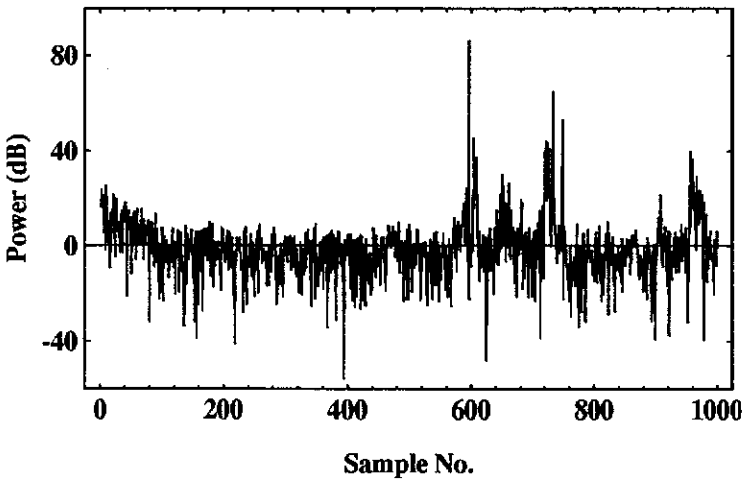
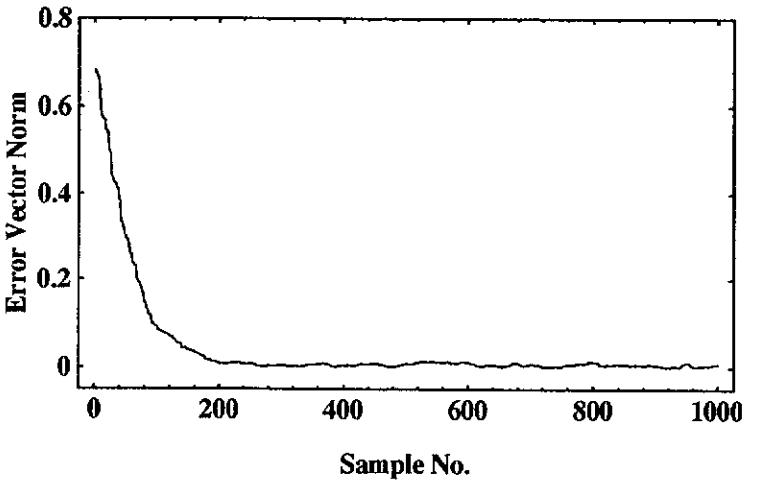


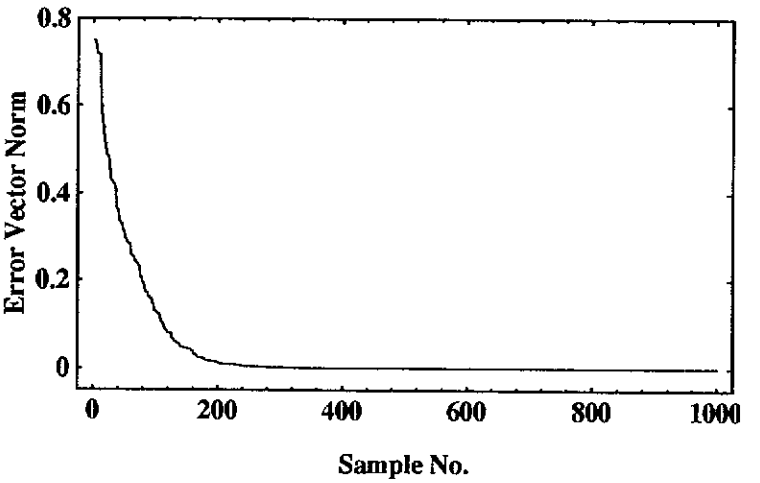
Figure 3.2 Instantaneous Array Output Power
Optimum output power ≅ 0 dB (horizontal line)

Note from Fig. 3.2 that after about 580 iterations the algorithm starts to display numerical instability. There are three possible areas in the elementary weights algorithm given in Table 3.1 which may lead to numerical problems, namely: (i) the matrix of elementary weights computed in Step 1 is a poor estimate of the true matrix; (ii) the correlation matrix updated in Step 2 is a poor estimate of the true matrix; and (iii) the $\Psi(n)$ matrix computed in Step 3 is a poor estimate of the first $K \times K$ block of the true $\left(\boldsymbol{C}^T \boldsymbol{R}^{-1} \boldsymbol{C}\right)^{-1}$ matrix. We can check the elementary weights by pre-computing the true elementary weight vectors from the optimum problem prior to starting the algorithm then measuring the norm of the error between the true vectors and the

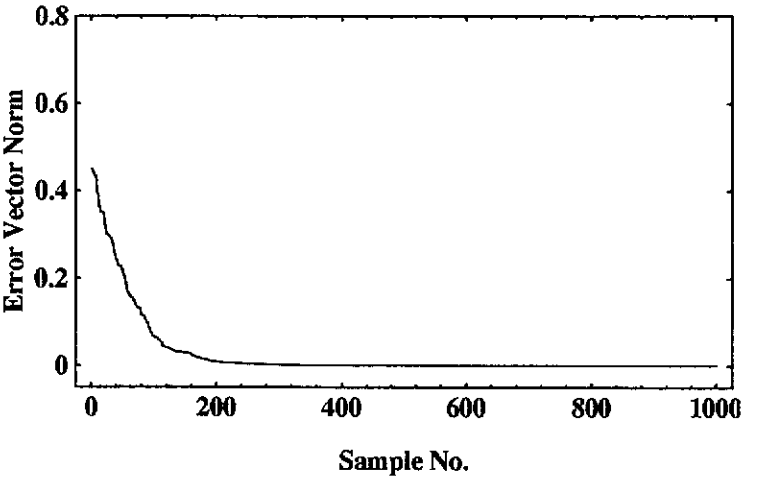
estimates computed at each sample instant. The three error vector norms are shown in Figs. 3.3(a) to 3.3(c).



(a) Norm of the Error in the First Elementary Weight, $\|W_{e1}(n) - W_{opt1}\|$



(b) Norm of the Error in the Second Elementary Weight, $\|W_{e2}(n) - W_{opt2}\|$



(c) Norm of the Error in the Third Elementary Weight, $\|W_{e3}(n) - W_{opt3}\|$

Figure 3.3 Error Vector Norms for the Three Elementary Weights

From Figs. 3.3(a) to (c) one can see that the elementary weights converge to the optimum vectors after about 300 iterations and don't appear to have any influence on the numerical instability problems shown in Fig. 3.2. The correlation matrix update in Step 2 of the algorithm can be checked in a similar manner. In this case we pre-compute the true matrix from the optimum problem and measure the normalised *spectral norm* of the error matrix formed from the difference between the true and estimated correlation matrices, i.e.,

$$\frac{\sqrt{\text{largest eigenvalue of } \left(\boldsymbol{R}_{opt} - \boldsymbol{R}(n)\right)^2}}{\sqrt{\text{largest eigenvalue of } \left(\boldsymbol{R}_{opt}\right)^2}}. \tag{3-36}$$

The normalised spectral norm of the error matrix at each sample instant is shown in Fig. 3.4.

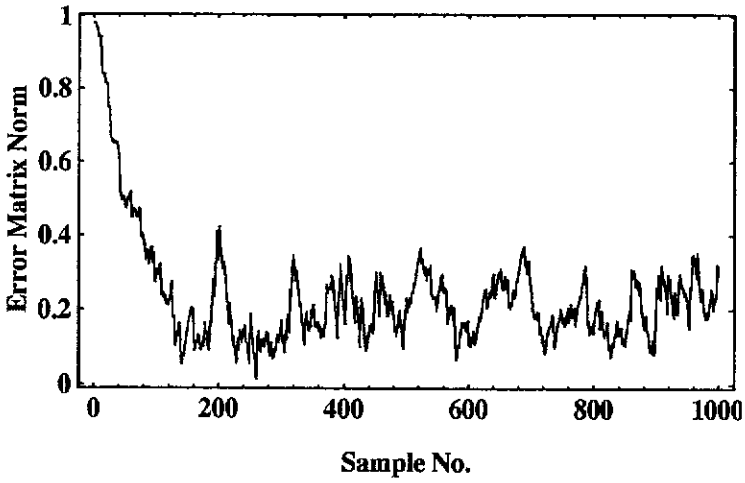


Figure 3.4 Spectral Norm of the Correlation Error Matrix

From Fig. 3.4, we see that the correlation matrix converges within 200 iterations and comparing Figs. 3.4 and 3.2 it appears that the correlation matrix update also has no influence on the numerical problems. Finally, examine at the matrix $\Psi(n)$ computed in Step 3 of Table 3.1 and see that the regions of instability shown in Fig. 3.2 correspond to sample instants when $\Psi(n)$ becomes negative definite. This is seen most clearly by looking at the quartic polynomial $P(\alpha)$ whose coefficients are computed from the elements of the $\Psi(n)$ matrix in Step 4 of Table 3.1. Since $P(\alpha)$ is the output power of the array, it must always be a positive function. However, if the $\Psi(n)$ becomes negative definite, $P(\alpha)$ becomes negative and the minimisation in Step 5 fails. Fig. 3.5 shows the quartic polynomial $P(\alpha)$ for 100 sample instants from sample numbers 500 to 600. Note how the polynomial becomes negative at the region corresponding to the numerical instability (between iteration numbers 580 and 600).

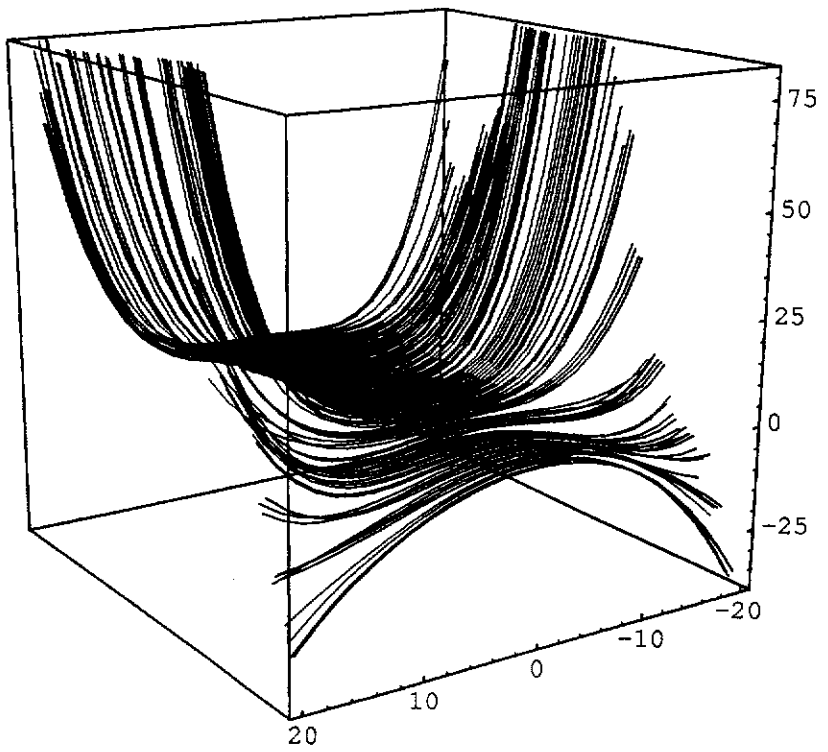


Figure 3.5 Evolution of the Quartic Polynomial $P(\alpha)$ from Sample Numbers 500 to 600 (Sample number increasing from 500 to 600 out of the page, value for α ranging from -20 to 20)

From the computer studies presented here, we have shown that even though the estimates of the elementary weight vectors and the correlation matrix converge to their optimum values and remain numerically stable, the algorithm, nevertheless, exhibits numerical instability problems. The instability appears to stem from errors resulting from the use of two different estimates of the correlation matrix within the algorithm and is marked by the estimate of the first $K \times K$ block of the $\left(\boldsymbol{C}'^{T}\boldsymbol{R}^{-1}\boldsymbol{C}'\right)^{-1}$ matrix becoming negative definite.

3.3 The Method of Double Inversion

In this section, we present an algorithm based on the matrix inversion lemma summarised in equation (3-7).

The section is organised in a similar manner to Section 3.2 and is divided into three parts. In the first part, the method is derived and a summary of the algorithm is presented. The second part gives a discussion of the method's characteristics and, finally, in the third part, results of some computer studies are presented which highlight some issues from the discussion.

3.3.1 Derivation

Using the matrix inversion lemma, the estimated inverse correlation matrix can be updated as new input samples become available according to the relation

$$\boldsymbol{R}^{-1}(n)=\frac{1}{\lambda}\left[\boldsymbol{R}^{-1}(n-1)-\frac{\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{T}(n)\boldsymbol{R}^{-1}(n-1)}{\frac{\lambda}{\eta}+\boldsymbol{x}^{T}(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}\right]. \tag{3-37}$$

Post-multiplying both sides of (3-37) by the constraint matrix $\boldsymbol{C}$ and pre-multiplying both sides by $\boldsymbol{C}^{T}$ we get

$$\boldsymbol{C}^{T}\boldsymbol{R}^{-1}(n)\boldsymbol{C}=\frac{1}{\lambda}\boldsymbol{C}^{T}\left[\boldsymbol{R}^{-1}(n-1)-\frac{\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{T}(n)\boldsymbol{R}^{-1}(n-1)}{\frac{\lambda}{\eta}+\boldsymbol{x}^{T}(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}\right]\boldsymbol{C}. \tag{3-38}$$

Using the following definitions

$$\beta(n)=\lambda/\eta+\boldsymbol{x}^{T}(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n) \tag{3-39}$$

$$\boldsymbol{b}(k)=\boldsymbol{C}^{T}\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n) \tag{3-40}$$

$$\Gamma(n)=\boldsymbol{C}^{T}\boldsymbol{R}^{-1}(n)\boldsymbol{C} \tag{3-41}$$

equation (3-38) can be written as

$$\Gamma(n)=\frac{1}{\lambda}\Gamma(n-1)+\left(\frac{-1}{\lambda\beta(n)}\right)\boldsymbol{b}(n)\boldsymbol{b}^{T}(n) \tag{3-42}$$

which is now in the form of equation (3-5). Applying the matrix inversion lemma a second time to (3-42) we obtain the following recursive update for the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}$

$$\Gamma^{-1}(n)=\lambda\left[\Gamma^{-1}(n-1)+\frac{\Gamma^{-1}(n-1)\boldsymbol{b}(n)\boldsymbol{b}^T(n)\Gamma^{-1}(n-1)}{\beta(n)-\boldsymbol{b}^T(n)\Gamma^{-1}(n-1)\boldsymbol{b}(n)}\right]. \tag{3-43}$$

The $\Gamma^{-1}(n)$ matrix can now be used to find the coefficients of the quartic polynomial in (3-3) thus allowing the optimum α, $\alpha_{opt}(n)$ to be determined. The inverse correlation matrix estimated in (3-37), together with $\alpha_{opt}(n)$, can then be used in (3-2) to find an estimate of the optimum weight vector.

The algorithm is summarised in Table 3.2.

Initialise:

$$\begin{aligned}\boldsymbol{R}^{-1}(0)&=\delta^{-1}\boldsymbol{I}\quad\delta>0\\ \Gamma^{-1}(0)&=\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(0)\boldsymbol{C}\right)^{-1}\end{aligned} \tag{3-44}$$

For each new sample $x(n)$ at time n:

1. Update $\beta(n)$ and $\boldsymbol{b}(n)$:

$$\beta(n)=\lambda/\eta+\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n) \tag{3-45}$$

$$\boldsymbol{b}(n)=\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n) \tag{3-46}$$

2. Update the $\Gamma^{-1}(n)$ matrix:

$$\Gamma^{-1}(n)=\lambda\left[\Gamma^{-1}(n-1)+\frac{\Gamma^{-1}(n-1)\boldsymbol{b}(n)\boldsymbol{b}^T(n)\Gamma^{-1}(n-1)}{\beta(n)-\boldsymbol{b}^T(n)\Gamma^{-1}(n-1)\boldsymbol{b}(n)}\right] \tag{3-47}$$

3. Update the inverse correlation matrix $\boldsymbol{R}^{-1}(n)$:

$$\boldsymbol{R}^{-1}(n)=\frac{1}{\lambda}\left[\boldsymbol{R}^{-1}(n-1)-\frac{\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)}{\frac{\lambda}{\eta}+\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}\right] \tag{3-48}$$

4. Compute the coefficients of the polynomial $P'(\alpha) = \frac{d}{d\alpha} P(\alpha)$ from the elements of the matrix $\Gamma^{-1}(n)$ using (3-3).

5. Compute the roots of the polynomial $P'(\alpha)$ and identify the α that minimises $P(\alpha)$. Denote this α as $\alpha_{opt}(n)$.

6. Compute the new weight vector

$$\boldsymbol{w}(n) = \boldsymbol{R}^{-1}(n)\boldsymbol{C}\Gamma^{-1}(n)\boldsymbol{h}(\alpha_{opt}) \tag{3-49}$$

Table 3.2 The Method of Double Inversion

3.3.2 Discussion

The Double Inversion algorithm presented above requires order L^2 ($O(L^2)$) floating point operations (FLOPS) per iteration to update the new weight vector. This represents a dramatic improvement in computational load over direct inversion of the estimated correlation matrix which requires $O(L^3)$ FLOPS. (Recall that L is the number of array elements). Unfortunately, despite the large reduction in complexity, the algorithm can suffer from some numerical problems.

If the algorithm is implemented exactly as it is presented above, it may suffer from *explosive divergence* (well documented for RLS techniques [8, 9, 10]). Explosive divergence is a result of finite precision effects, for example rounding and truncation, in the algorithm calculations which act to destroy the symmetric and positive definite properties of the estimated inverse correlation matrix, $\boldsymbol{R}^{-1}(n)$, and the matrix $\Gamma^{-1}(n)$. Bottomley and Alexander in [8] and Haykin in [9, p 485] give versions of the RLS algorithm that preserve the symmetry of the inverse correlation matrix estimate when the algorithm is used in a finite precision environment. Even though these methods improve the stability of the algorithm in the short term and reduce the chance of explosive divergence, the phenomenon can still occur. Further measures to avoid this problem are discussed in the next chapter.

Another numerical problem arises when the algorithm is implemented in a finite precision environment. On inspection of the algorithm in Table 3.2 one can see that there is no facility to re-introduce the constraints at each step in the iteration or to correct for numerical errors which bias the weight vector away from the constraint space. Consequently, finite precision effects will eventually allow the weight vector to drift out of the constraint space. The problem is demonstrated numerically in the computer studies that follow in the next section.

3.3.3 Computer Studies

The simulation results presented below demonstrate the numerical problem discussed whereby the weight vector solution drifts out of the constraint space due to finite precision effects in the algorithm.

The simulation scenario involving a narrowband, 5-element circular array is shown in Fig. 3.6.

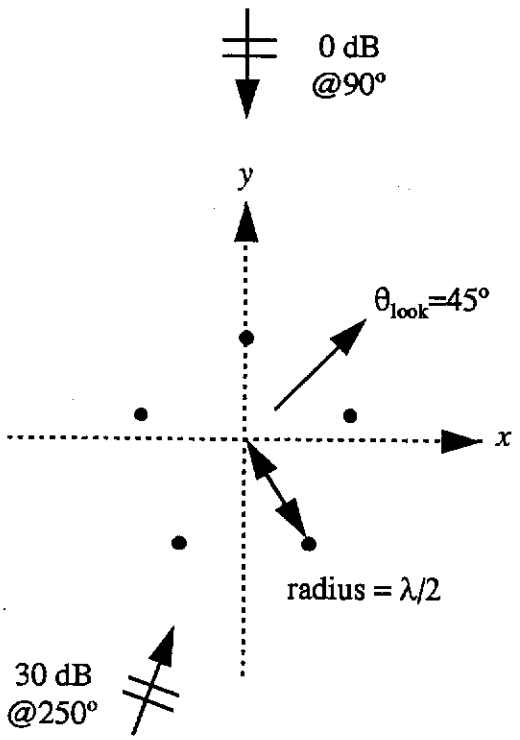


Figure 3.6 Array Structure and Source Scenario

In addition, the correlation matrix was initialised with $\delta = 100$ in equation (3-44) and the forgetting factor was set to $\lambda = 0.95$.

Fig. 3.7 shows the instantaneous output power of the array at each sample instant as the algorithm adapts to the signal environment.

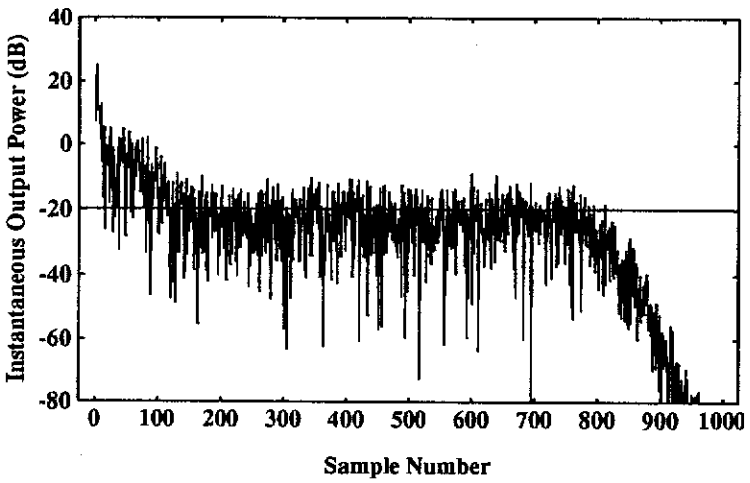


Figure 3.7 Adaptive Array Processor Output Power Learning Curve
Computed optimum output power is -19.83 dB
(indicated by the horizontal line)

Note from Fig. 3.7 that the algorithm displays the characteristic fast convergence (about 150 to 200 samples), typical of RLS algorithms. Note also that after about 750 samples the power starts to drop away. At this stage the constraints are no longer being met and the weight vector has moved out of the constraint space and is approaching the zero vector, effectively suppressing both the noise/interference and the desired signal. This can be seen clearly in Fig. 3.8 which shows the weight vector norm starting to decrease rapidly after about 750 iterations reaching a value of zero at about 900 iterations.

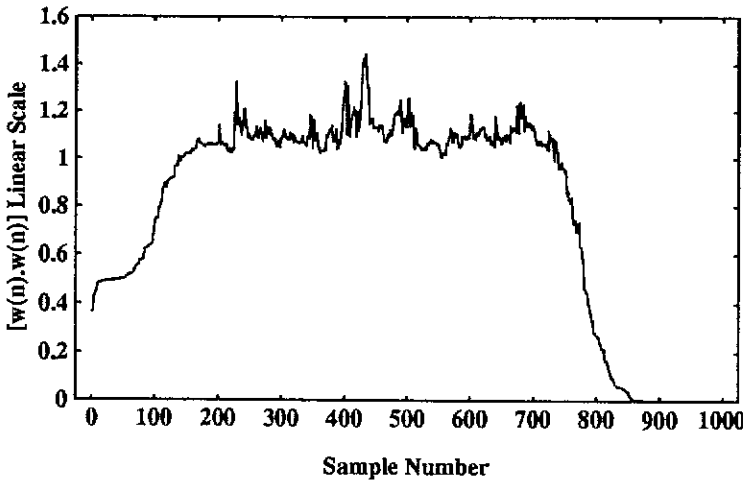


Figure 3.8 Weight Vector Norm

3.4 Conclusions

In this chapter we have investigated two methods for adaptively estimating the $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ matrix, a key to solving the two-stage optimisation problem. From computer studies, the first method (the method of elementary weights) was found to be unsuitable due to the significant amount of noise produced by the algorithm itself. The second method (double inversion) seems more promising. It is fast converging, computationally efficient and converges to the correct solution. Problems, however, are encountered in a finite precision environment. Since the constraints are introduced only once when the algorithm is started, the weight vector can drift out of the constraint space due to rounding errors in the algorithm calculations. What is required then, is some way of re-introducing the constraints at regular intervals in the algorithm to correct for any drift of the solution. This is the problem addressed in the next chapter.

3.5 References

[1] S. Lawrence Marple, Jr., *Digital Spectral Analysis with Applications*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1987.

[2] Maurice G. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker Inc., New York, 1987.

[3] I. S. Reed, J. D. Mallett, L. E. Brennan, "Rapid Convergence Rate in Adaptive Arrays", *IEEE Trans. Aerospace and Electronic Systems*, vol. 10, pp 853 - 863, November 1974.

[4] W. S. Hodgkiss and L. W. Nolte, "Bayes Optimal versus Estimate and Plug Array Processor Performance when there is Directional Uncertainty", *G. Tacconi (Editor), Aspects of Signal Processing*, Pt. 1, NATO Advanced Study Institute Series, D. Reidel Publishing Co., Dordrecht, Holland, pp. 203-212, 1977.

[5] O. L. Frost, III, "An Algorithm for Linearly Constrained Adaptive Array Processing", *Proc. IEEE*, vol. 60, no. 8, pp. 926-935, August 1972.

[6] G. Dahlquist and A. Björck, *Numerical Methods*, Prentice-Hall Series in Automatic Computation, Englewood Cliffs, N.J. 1974.

[7] R. Bronson, *Matrix Operations*, Schaum's Outline Series, McGraw-Hill, New York, 1989.

[8] G. E. Bottomley, S. T. Alexander, "A Novel Approach for Stabilising Recursive Least Squares Filters", *IEEE Trans. Signal Processing*, vol. 39, no. 8, pp. 1770-1779, August 1991.

[9] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1991.

[10] M. H. Verhaegen, "Round-off Error Propagation in Four Generally Applicable, Recursive, Least-squares Estimation Schemes", *Automatica*, vol. 25, no. 3, pp. 437-444, 1989.

4. *The Adaptive Antenna Array Processor with NS Derivative Constraints*

In the previous chapter it was shown that the method of Double Inversion for adaptively estimating the $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ matrix from the input data has fast convergence. However, since the constraints are imposed only at the start of the algorithm the weight vector will eventually drift out of the constraint space due to finite precision effects. This chapter presents a technique that repeatedly imposes the constraints at each iteration of the algorithm thus correcting for any errors resulting from fixed precision effects.

We begin by summarising a method proposed by Resende, Romano and Bellanger [1] which allows linear constraints to be incorporated within a recursive least squares framework. The method ensures the constraints are met at each iteration in a manner similar to Frost's linearly constrained LMS algorithm [6].

In the discussion of the Double Inversion algorithm we also mentioned the issue of explosive divergence. In the second part of this chapter we will investigate this issue further and look at a method developed by Bottomley and Alexander [2] for stabilising RLS filters.

These two methods will then be brought together to form an efficient, numerically stable, fast converging adaptive algorithm for estimating the $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ matrix; the first stage of the two-stage optimisation process.

4.1 A Numerically Robust RLS Filter Algorithm with Quadratic Constraints

4.1.1 The Algorithm of Resende, Romano and Bellanger

The algorithm presented in the paper by Resende et al. [1] gives a recursive scheme for estimating the matrix $\boldsymbol{R}^{-1}\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ which, in their procedure, is represented by the matrix $\boldsymbol{Q}(n)$, i.e.,

$$\boldsymbol{Q}(n) = \boldsymbol{R}^{-1}(n)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1} \tag{4-1}$$

The update is found from the product of two separate recursions, namely a recursive procedure for updating the matrix $\boldsymbol{R}^{-1}(n)\boldsymbol{C}$ and a recursive update for the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}$.

The recursive update of the matrix $\boldsymbol{R}^{-1}(n)\boldsymbol{C}$ is derived from the inverse correlation matrix update in the RLS procedure:

$$\boldsymbol{R}^{-1}(n) = \frac{1}{\lambda}\left[\boldsymbol{R}^{-1}(n-1) - \boldsymbol{g}(n)\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\right] \tag{4-2}$$

where $\boldsymbol{g}(n) \in \mathcal{R}^{2L}$ is the RLS gain vector [3].

Post-multiplying (4-2) by $\boldsymbol{C}$ and defining $K(n) = \boldsymbol{R}^{-1}(n)\boldsymbol{C}$ gives the required recursion

$$K(n) = \frac{1}{\lambda}\left[K(n-1) - \boldsymbol{g}(n)\boldsymbol{x}^T(n)K(n-1)\right]. \tag{4-3}$$

The recursive update of the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}$ is found in the same manner as equation (3-34) in the Double Inversion algorithm in the last chapter, i.e., via the double application of the matrix inversion lemma. Resende et al update the $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}$ matrix as follows:

Defining $\Gamma^{-1}(n) = \left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}$, the recursion for $\Gamma^{-1}(n)$ is given by

$$\Gamma^{-1}(n) = \lambda\left[\Gamma^{-1}(n-1) + \boldsymbol{l}(n)\boldsymbol{x}^T(n)K(n-1)\Gamma^{-1}(n-1)\right] \tag{4-4}$$

where

$$l(n) = \frac{\Gamma^{-1}(n-1)C^T g(n)}{1 - x^T(n)K(n-1)\Gamma^{-1}(n-1)C^T g(n)}. \tag{4-5}$$

Combining (4-3) and (4-4) it can be shown [5] that the matrix $Q(n)$ can be updated by

$$Q(n) = \left[Q(n-1) - g(n)v^T(n)\right]\left[I_M + \frac{u(n)v^T(n)}{1 - v^T(n)u(n)}\right] \tag{4-6}$$

where $u(n) \in \mathcal{R}^M$ is given by

$$u(n) = C^T g(n), \tag{4-7}$$

and $v(n) \in \mathcal{R}^M$ is given by

$$v^T(n) = x^T(n)Q(n-1). \tag{4-8}$$

The update given in (4-6) suffers, however, from the same problem encountered by the Double Inversion algorithm. Since the constraints are only introduced at the initialisation stage of the procedure, the weight vector computed using $Q(n)$ will not satisfy the constraints after a number of iterations due to the accumulation of round-off errors.

To resolve this problem, Resende et al introduced an additional term in the update of (4-6) which corrects for round-off errors by ensuring a specific property of the $Q(n)$ matrix holds at each iteration. The correction term essentially works as a least squares version of Frost's linearly constrained LMS algorithm [6]. To understand how the linearly constrained RLS algorithm maintains numerical robustness, it is helpful to look briefly at how Frost's LMS algorithm achieves this.

The weight vector update for the Frost LMS algorithm is given (in expanded form) below

$$w(n) = \underbrace{w(n-1) - \mu P y(n-1)x(n-1)}_{\text{stochastic gradient descent term}} + \underbrace{C\left(C^T C\right)^{-1}\left[f - C^T w(n-1)\right]}_{\text{correction term}} \tag{4-9}$$

where

$$P = I - C\left(C^T C\right)^{-1} C^T \tag{4-10}$$

and f is the constraint vector which, in the absence of round-off errors is given in the equality

$$C^T w(n) = f \tag{4-11}$$

The update of (4-9) is written in expanded form to emphasise that the factor $\left[f - C^T w(n-1)\right]$ is not assumed to be zero as would be the case if the constraints were exactly satisfied. The presence of this factor permits the algorithm to correct for small deviations from the constraints due to finite precision effects.

In the constrained RLS algorithm a property similar to (4-11) for the weight vector in the Frost algorithm is required for the matrix $Q(n)$. According to the definition (4-1), in an exact implementation the following equality holds

$$C^T Q(n) = I_M \tag{4-12}$$

Therefore, if we denote $Q'(n)$ as the matrix containing round-off errors, we may introduce a correcting term in a manner similar to Frost's algorithm of (4-9) above

$$Q(n) = Q'(n) + C\left(C^T C\right)^{-1}\left[I_M - C^T Q'(n)\right] \tag{4-13}$$

Equations (4-6), (4-7), (4-8) and (4-13) constitute a major part of the linearly constrained RLS algorithm of Resende, Romano and Bellanger.

4.1.2 Summary of the New Algorithm

We are now ready to apply the above procedure to the problem we are concerned with in this thesis. Recall that the two-stage minimisation problem is given by

$$\min_{\alpha}\left[\begin{array}{c} \min_{w} P(w) \\ \text{subject to} \quad C^T w = h(\alpha) \end{array}\right] \tag{4-14}$$

and that inner stage of this problem is optimised by the weight vector

$$w_{opt}(\alpha) = R^{-1}C\left(C^T R^{-1} C\right)^{-1} h(\alpha). \tag{4-15}$$

It is clear from (4-1) that $Q(n)$ is an estimate of the matrix $R^{-1}C\left(C^T R^{-1}C\right)^{-1}$. However, we still need to find the optimum α before the global optimum weight

vector can be determined. In Chapter 2 it was shown that the optimum α could be found by locating the global minimum of the quartic polynomial given by

$$\min_{\alpha} P(\alpha) = \boldsymbol{h}^T(\alpha)\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{h}(\alpha). \tag{4-16}$$

To obtain the coefficients of this polynomial we must still find an estimate of the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$.

If an estimate of the correlation matrix, $\boldsymbol{R}(n)$, is made at the same time as the $\boldsymbol{Q}(n)$ matrix is being updated then an estimate of the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ can be found by pre-multiplying $\boldsymbol{Q}(n)$ by $\left(\boldsymbol{C}^T\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}(n)$, i.e.,

$$\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1} = \left(\boldsymbol{C}^T\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}(n)\boldsymbol{Q}(n). \tag{4-17}$$

Thus, we can now summarise the new algorithm which adaptively estimates the global optimum weight vector for the two-stage minimisation problem of (4-14). The algorithm, summarised in Table 4.1, incorporates the numerical robust properties of the linearly constrained RLS procedure of Resende, Romano and Bellanger.

Initialise:

$$\left.\begin{aligned} \boldsymbol{R}(0) &= \delta \boldsymbol{I} \\ \boldsymbol{R}^{-1}(0) &= \delta^{-1}\boldsymbol{I} \end{aligned}\right\} \delta > 0$$
$$\boldsymbol{Q}(0) = \boldsymbol{R}^{-1}(0)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(0)\boldsymbol{C}\right)^{-1} \tag{4-18}$$

For each new input sample, $\boldsymbol{x}(n)$, at time n:

1. Compute the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}$:

1.1 Update the correlation matrix $\boldsymbol{R}(n)$:

$$\boldsymbol{R}(n) = \lambda\boldsymbol{R}(n-1) + \boldsymbol{x}(n)\boldsymbol{x}^T(n) \tag{4-19}$$

1.2. Update the RLS gain vector $\boldsymbol{g}(n)$ (using the standard RLS algorithm).

1.3. Update $\boldsymbol{Q}(n)$:

$$\boldsymbol{u}(n) = \boldsymbol{C}^T\boldsymbol{g}(n) \tag{4-20}$$

$$\boldsymbol{v}^T(n) = \boldsymbol{x}^T(n)\boldsymbol{Q}(n-1) \tag{4-21}$$

$$\boldsymbol{Q}'(n) = \left[\boldsymbol{Q}(n-1) - \boldsymbol{g}(n)\boldsymbol{v}^T(n)\right]\left[\boldsymbol{I}_M + \frac{\boldsymbol{u}(n)\boldsymbol{v}^T(n)}{1-\boldsymbol{v}^T(n)\boldsymbol{u}(n)}\right] \tag{4-22}$$

1.4. Correct $\boldsymbol{Q}(n)$:

$$\boldsymbol{Q}(n) = \boldsymbol{Q}'(n) + \boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{C}\right)^{-1}\left[\boldsymbol{I}_M - \boldsymbol{C}^T\boldsymbol{Q}'(n)\right] \tag{4-23}$$

1.5. Compute the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}$:

$$\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1} = \left(\boldsymbol{C}^T\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}(n)\boldsymbol{Q}(n) \tag{4-24}$$

2. Find the optimum α and the global optimum weight vector:

2.1 Compute the coefficients of the polynomial $P'(\alpha) = \frac{d}{d\alpha} P(\alpha)$ from the elements of $\left(\boldsymbol{C}^T \boldsymbol{R}^{-1}(n)\boldsymbol{C}\right)^{-1}$.

2.2 Compute the roots of $P'(\alpha)$ and identify the α that minimises $P(\alpha)$. Denote this α as $\alpha_{opt}(n)$.

2.3 Compute the weight vector:

$$\boldsymbol{w}(n) = \boldsymbol{Q}(n)\boldsymbol{h}\left(\alpha_{opt}(n)\right) \tag{4-25}$$

Table 4.1 Quadratically Constrained Adaptive Filter Algorithm incorporating the Technique of Resende, Romano and Bellanger

4.2 Discussion

4.2.1 Numerical Stability of the New Algorithm

In Step 1.2 of Table 4.1, we update the RLS gain vector, $\boldsymbol{g}(n)$, via the standard RLS procedure. The recursion for the gain vector is given by

$$\boldsymbol{g}(n) = \frac{\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}{\lambda + \boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}. \tag{4-26}$$

In Section 3.3.2, we indicated that some versions of the RLS procedure are more numerically robust than others. It has been found [7] that symmetry in the inverse correlation matrix is critical to the numerical stability of the procedure. In [2] and [8] two versions of the RLS algorithm which preserve the symmetry of the inverse correlation matrix are presented. The procedure in Table 4.2 for updating the gain vector in step 1.2 of Table 4.1 is a part of the symmetry preserving RLS algorithm presented in [8]

$$\boldsymbol{p}(n) = \boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n) \tag{4-27}$$

$$\mu(n) = \boldsymbol{x}^T(n)\boldsymbol{p}(n) \tag{4-28}$$

$$\beta(n) = \frac{1}{(\lambda + \mu(n))} \tag{4-29}$$

$$\boldsymbol{g}(n) = \beta(n)\boldsymbol{p}(n) \tag{4-30}$$

$$\boldsymbol{R}^{-1}(n) = \frac{1}{\lambda}\left[\boldsymbol{R}^{-1}(n-1) - \beta(n)\left(\boldsymbol{p}(n)\boldsymbol{p}^T(n)\right)\right] \tag{4-31}$$

Table 4.2 Symmetry Preserving RLS Algorithm

Although the formulation given in Table 4.2 significantly improves the numerical robustness of the gain vector update, in a fixed precision environment a numerical effect known as *explosive divergence* can still occur. Explosive divergence is a problem of numerical instability which is well documented for the standard RLS algorithm, [2, 4, 7, 8]. It occurs when the estimated inverse correlation matrix, $\boldsymbol{R}^{-1}(n)$, loses its property of positive definiteness due to finite precision effects in the time update equations.

An effective technique for preventing explosive divergence in the symmetry preserving formulation of the standard RLS algorithm (a major part of which is given in Table 4.2) was developed by Bottomley and Alexander in [2]. They show that the complex error propagation mechanism which tends to bias the inverse correlation matrix, $\boldsymbol{R}^{-1}(n)$, toward being negative definite can be counteracted by applying certain rounding rules when performing the calculations in Table 4.2. The rounding rules are summarised in Table 4.3 and apply to the quantities defined in the update equations in Table 4.2.

1. Round down or truncate when forming $\beta(n)$ in (4-29).
2. Round up when forming $\mu(n)$ in (4-28).
3. When forming $\boldsymbol{p}_{\mu}(n)$ in (4-27) to compute $\mu(n)$ in (4-28), round up if $x_i(n) > 0$, and round down if $x_i(n) < 0$.
4. When forming $\boldsymbol{p}_{\varphi}(n)$ in (4-27) to compute the diagonal elements of $\boldsymbol{p}(n)\boldsymbol{p}^T(n)$ in (4-31), round down in magnitude.
5. When forming the diagonal elements of $\boldsymbol{p}(n)\boldsymbol{p}^T(n)$ in (4-31), subtract "1" from the accumulator if it is non-zero.
6. When forming $\beta(n)\left[\boldsymbol{p}(n)\boldsymbol{p}^T(n)\right]$ in (4-31), round down on the diagonal.
7. When forming $\boldsymbol{R}^{-1}(n)$ in (4-31), round up on the diagonal.

Table 4.3 Rounding Rules for the conventional RLS Algorithm

Note that in rules 3, 4 and 5, three different $\boldsymbol{p}(n)$ vectors need to be generated, namely, $\boldsymbol{p}_{\mu}(n)$ for the calculation of $\mu(n)$, $\boldsymbol{p}_{\varphi}(n)$ for the calculation of the diagonal elements of $\boldsymbol{p}(n)\boldsymbol{p}^T(n)$, and $\boldsymbol{p}(n)$ which is created using conventional rounding and is used to calculate the off-diagonal elements of $\boldsymbol{p}(n)\boldsymbol{p}^T(n)$.

A detailed analysis of the complex error propagation mechanism in the standard (unconstrained) RLS algorithm is presented in [2]. A summary of this analysis is beyond the scope of this thesis. However, in Appendix B we look at the primary mechanism which brings about explosive divergence in the conventional RLS (CRLS) algorithm and demonstrate the effect of the rounding rules given in Table 4.3 by simulating a one dimensional, unconstrained adaptive filter in a fixed wordlength environment.

By applying the rounding rules in Table 4.3 to the inverse correlation matrix update in Table 4.1, we ensure that the gain vector update in Table 4.2 remains numerically stable in a fixed precision environment. The algorithms in Tables 4.1, 4.2 and 4.4, therefore constitute a numerically robust, recursive least-squares method for quadratically constrained adaptive filtering. The quadratically constrained adaptive

filter of interest in this thesis is, of course, the array processor with second order NS derivative constraints.

It should be noted that since the update for the matrix $\boldsymbol{Q}(n)$ is also essentially a variant of the RLS algorithm it can suffer from similar numerical instability problems. On close inspection of the update equation for $\boldsymbol{Q}(n)$, equation (4-22), a parameter similar to the angle parameter for the standard RLS algorithm discussed in Appendix B can be identified. This second angle parameter, $\gamma(n) \in \mathcal{R}$, is defined as

$$\gamma(n) = 1 - \boldsymbol{v}^T(n)\boldsymbol{u}(n). \tag{4-32}$$

In Appendix C it is shown that this parameter is bounded as follows

$$0 \leq \gamma(n) \leq 1. \tag{4-33}$$

Although a detailed analysis of the error propagation mechanism in the $\boldsymbol{Q}(n)$ matrix update has not yet been done, the results of numerous computer studies suggest that this second angle parameter is a critical indicator of the onset of numerical instability.

The difference between the symmetry preserving version of the standard RLS algorithm given in Table 4.2 and other versions which do not preserve symmetry in the inverse correlation matrix is in the exact way the quantities in the update equations are calculated, (see for example [4, p. 485]). In [10], Low shows that, in a similar way to the standard RLS algorithm, the numerical stability of the $\boldsymbol{Q}(n)$ matrix update can be improved significantly in a fixed precision environment if the numerator, $\boldsymbol{u}(n)\boldsymbol{v}^T(n)$, in the update equation (4-22) is evaluated in expanded form rather than by computing the vector outer product $\boldsymbol{u}(n)\boldsymbol{v}^T(n)$. Two expansions have been found to improve the numerical stability of the algorithm and are given by:

$$\boldsymbol{u}(n)\boldsymbol{v}^T(n) = \frac{\left\{\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)\right\}\left\{\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right\}\left[\boldsymbol{C}^T\boldsymbol{C}\right]^{-1}\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{Q}(n-1)}{\lambda + \boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)} \tag{4-34}$$

$$\boldsymbol{u}(n)\boldsymbol{v}^T(n) = \frac{\boldsymbol{C}^T\left\{\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)\right\}\left\{\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\right\}\left\{\boldsymbol{C}\left[\boldsymbol{C}^T\boldsymbol{C}\right]^{-1}\boldsymbol{C}^T\right\}\boldsymbol{R}^{-1}(n-1)\boldsymbol{Q}(n-1)}{\lambda + \boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)} \tag{4-35}$$

Note that the matrices $\left[C^T C\right]^{-1} C^T$ and $C\left[C^T C\right]^{-1} C^T$ can be pre-computed and that the order of the computation in (4-34) and (4-35) must proceed with the bracketed quantities first. Note too that the stabilising effect of this technique has only been demonstrated through computer studies. An analysis of the error propagation mechanism within the $Q(n)$ matrix recursion equations is left as an area for further research.

4.2.2 Computational Complexity

The number of floating point operations required for each step of the algorithm in Tables 4.1 and 4.2 for an array processor with M constraints and L antenna elements is given in Table 4.4.

Step	Additions	Multiplications	Divisions
1.1	$4L^2$	$8L^2$	0
1.2	$8L^2$	$4L(4L+1)$	1
1.3	$M(6LM+4L+M-1)$	$M(6LM+6L+M+1)$	1
1.4	$(2L-1)(2ML+M^2)$	$2L(2ML+M^2)$	0
2.1 (see note 1 below)	$2M(M-1)$	$M(M+1)$	0
2.2 (see note 2 below)	-	-	-
2.3	$2L(M-1)$	$2LM$	0
Totals	$2L(6L+2LM+4M^2+2M-1)$ $+M(2M-3)$	$4L(6L+LM+2M^2+2M+1)$ $+2M(M+1)$	2

Table 4.4 Computational Complexity

Notes:

1. The number of floating point operations shown for Step 2.1 for the computation of the coefficients of the polynomial $P'(\alpha)$ are absolute maximum values only. Since the vector $\boldsymbol{h}(\alpha)$ generally has a number of its elements equal to zero, a large reduction in the number of floating point operations required to find the coefficients of the polynomial $P'(\alpha)$ can be achieved by exploiting the structure of $\boldsymbol{h}(\alpha)$. For example, for a narrowband, 2D array processing scenario the vector $\boldsymbol{h}(\alpha)$ has two (out of five) elements equal to zero and consequently the coefficients of $P'(\alpha)$ can be found with only 5 multiplications and 1 addition compared with the 20 multiplications and 40 additions indicated in the table entries.

2. The computational complexity involved in finding the roots of the polynomial $P'(\alpha)$ in Step 2.2 depends on a number of factors, for example, the polynomial factorisation method used and the accuracy required of the root estimates. This step of the algorithm is discussed in more detail in the next chapter. Regardless, it is sufficient to say at this point that the polynomial factorisation algorithm used requires $O(r^2)$ floating point operations per iteration, where r is the order of the polynomial, and offered a quadratic rate of convergence toward the polynomial roots.

From the entries given in Table 4.6 it can be seen that the computational complexity of the algorithm is of the order L^2M. Since the number of constraints M, is often small compared with the number of array elements L, the computational complexity becomes $O(L^2)$. This is a significant improvement over sample matrix inversion methods which require $O(L^3)$ floating point operations to invert the correlation matrix.

Another important point worth noting in the algorithm presented in Table 4.1 is that the constraint correction of (4-23) need not be performed at each iteration of the algorithm. Depending on the machine precision and the algorithm parameters, the

correction may only need to be applied at regular intervals rather than every iteration thus resulting in further reductions in the computational load.

4.2.3 Stage 2 of the Two-stage Minimisation

Note that the algorithm described above has been presented in two parts corresponding to the inner and outer stages of the two stage minimisation problem (Equation 4-14). In the second stage of this algorithm, where the optimum value for α is determined, we have not specified the method to compute the roots of the cubic polynomial. In the 2D antenna array processing scenario we have been considering so far, conventional root finding techniques, for example the direct form solution or Newton's method, could be applied. If we were working with a 3D antenna array processing scenario we would be faced with finding the roots of a 9th order polynomial [9]. Fortunately, more efficient root finding methods that take advantages of additional properties of the polynomial $P'(\alpha)$ are available. These are discussed in the next chapter.

4.3 Computer Studies

The computer studies that follow illustrate the application of the new algorithm in the adaptive narrowband array processing problem. The first set of simulation results highlight the self-correcting feature of the new algorithm resulting from the application of the technique of Resende, Romano and Bellanger. For ease of comparison with the Double Inversion algorithm, the simulation scenario is set identical to that used in Section 3.3.3.

In the second set of studies, we compare simulation results obtained using the new processor with those obtained using two other adaptive array processors, namely, the LCMV array processor and an LCMV array processor with additional linear constraints which approximate the quadratic, second order NS derivative constraints.

In the third and final set of simulation results the numerical characteristics of the new algorithm in a fixed precision environment are investigated. In particular, we examine the behaviour of the angle parameter $\gamma(n)$ which was discussed in Section 4.2.1.

4.3.1 Constraint Correction Feature

The array structure and source scenario is repeated in Fig. 4.1.

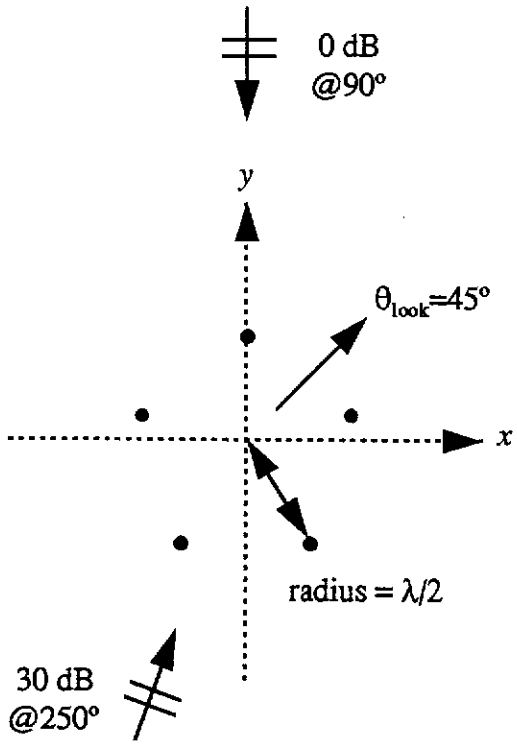


Figure 4.1 Array Structure and Source Scenario

Additional simulation parameters include:

- Forgetting factor, $\lambda = 0.95$.
- Initialisation constant, $\delta = 100$ in (4-18).
- Source and interference signals modelled as zero mean, Gaussian distributed sequences.
- Element self noise = -20 dB (also modelled as zero-mean, Gaussian distributed signal).
- Array phase centre located at the centre of the circle.

Figs. 4.2 and 4.3 show the instantaneous output power of the array and the weight vector norm, respectively, at each sample instant as the array processor adapts to the signal environment.

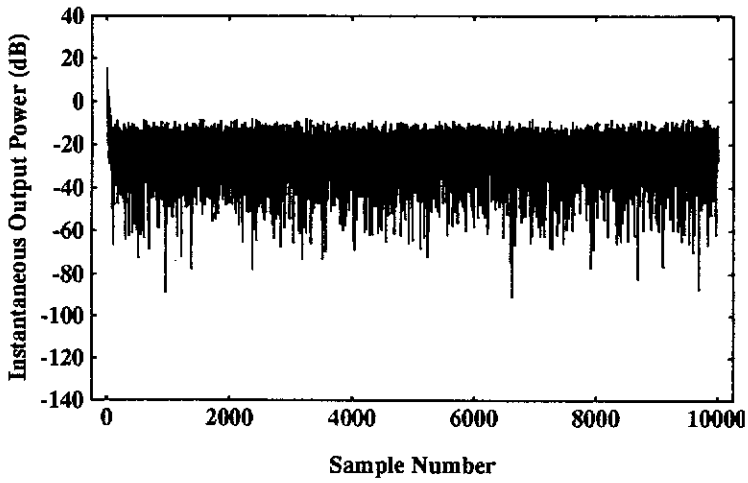


Figure 4.2 Adaptive Array Processor Output Power Learning Curve

Computed optimum output power is -19.83 dB

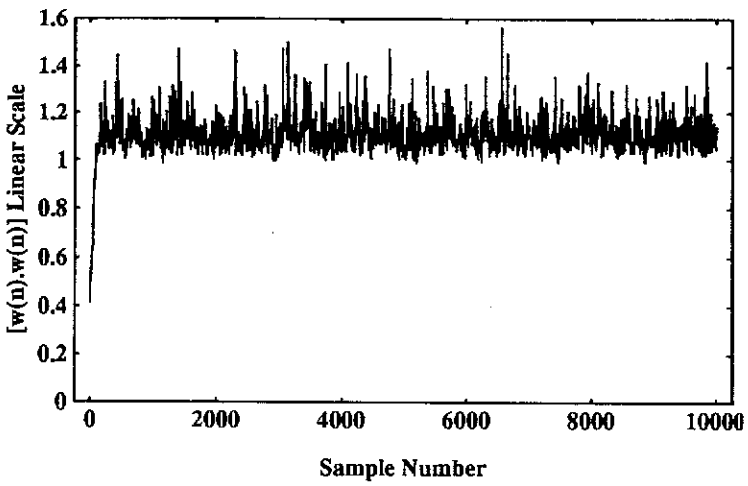


Figure 4.3 Weight Vector Norm $\left[w^T(n)w(n)\right]$

Note that identical input data sets were used for the first 1000 samples of the above simulation and the Double Inversion algorithm simulation of Section 3.3.3. Comparing Figs. 4.2 and 4.3 with Figs. 3.5 and 3.6, one sees that, even though the algorithm in Table 4.1 has been run for ten times the number of iterations, it still has not demonstrated the problems encountered with the Double Inversion algorithm.

4.3.2 Comparison with Other Antenna Array Processors

The first set of studies in this section demonstrates the robustness of the new adaptive array processor to pointing errors. Here, the average output power of the new

processor is compared with that of the LCMV processor for two look directions that differ by only four degrees. The simulations assume a 5-element circular array with source and interference scenarios as shown in Fig.4.4.

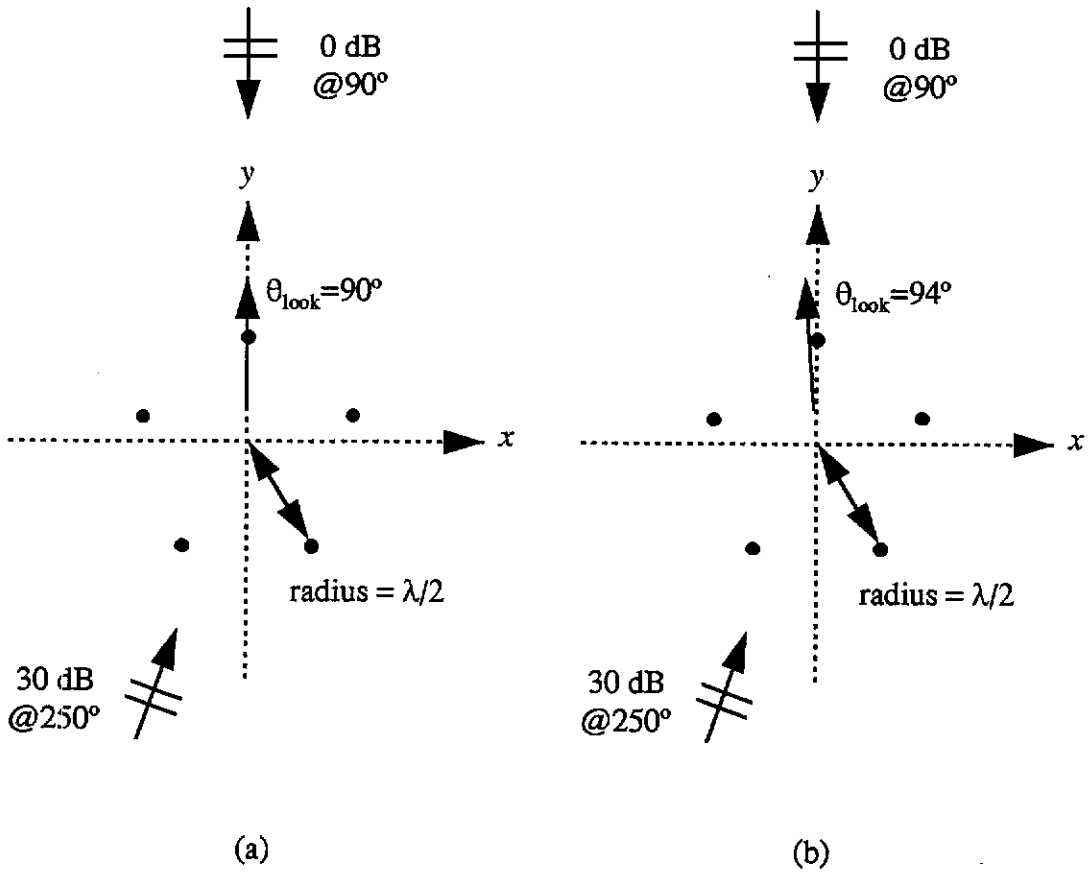
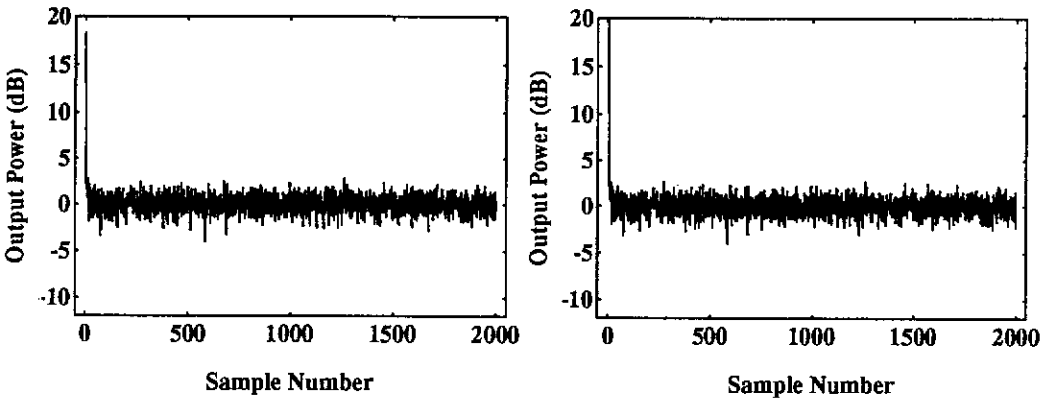


Figure 4.4 Array Structures and Source Scenarios: (a) look direction = 90°, (b) look direction = 94°

Additional simulation parameters include:

- Forgetting factor, $\lambda = 0.99$
- Initialisation constant, $\delta = 100$
- Source and interference signals modelled as zero mean, Gaussian distributed sequences.
- Element self noise = -20 dB (also modelled as zero-mean, Gaussian distributed signal).
- Array phase centre located at the centre of the circle.

Figs. 4.5 and 4.6 show the instantaneous output power of the two array processors ensemble averaged over 50 separate simulation runs. The plots in Figs. 4.5(a) and 4.6(a) correspond to the scenario in Fig. 4.4(a) and those in Figs. 4.5(b) and 4.6(b) correspond to the scenario in Fig. 4.4(b). The input data sequence in all simulations is identical.

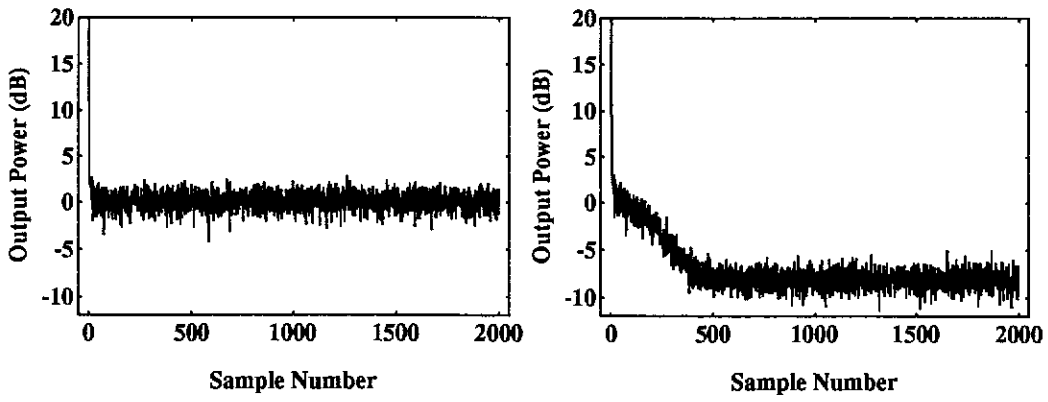


(a) (b)

Figure 4.5 Ensemble Averaged Output Power vs. Sample Number for the New Adaptive Array Processor with Quadratic, 2nd Order NS Derivative Constraints

(a): Looking at the source (scenario in Fig 4.4(a))

(b): 4° mismatch between the look and source directions (scenario in Fig 4.4(b))



(a) (b)

Figure 4.6 Ensemble Averaged Output Power vs. Sample Number for a LCMV Array Processor

(a): Looking at the source (scenario in Fig 4.4(a))

(b): 4° mismatch between the look and source directions (scenario in Fig 4.4(b))

In Fig. 4.5 we see that the new array processor is unaffected by the 4° pointing error. It produced an output of the desired signal in both cases and converging to the optimum output power of 0 dB. In Fig. 4.6, however, the 4° pointing error has caused the LCMV array processor to regard the desired signal at 90° as interference, suppressing it by about 8 dB.

In the second set of studies, we investigate the interference suppression properties of the new processor. The average output power of the new processor for a look direction away from any source is compared with that of a similar processor having a set of constraints that are linear approximations to the 2nd order NS quadratic constraints and as such are only sufficient for 2nd order flatness in the array power response. The set of linear, approximate constraints are found by setting $\alpha = 0$ in (4-14). The simulation scenario is shown in Fig. 4.7.

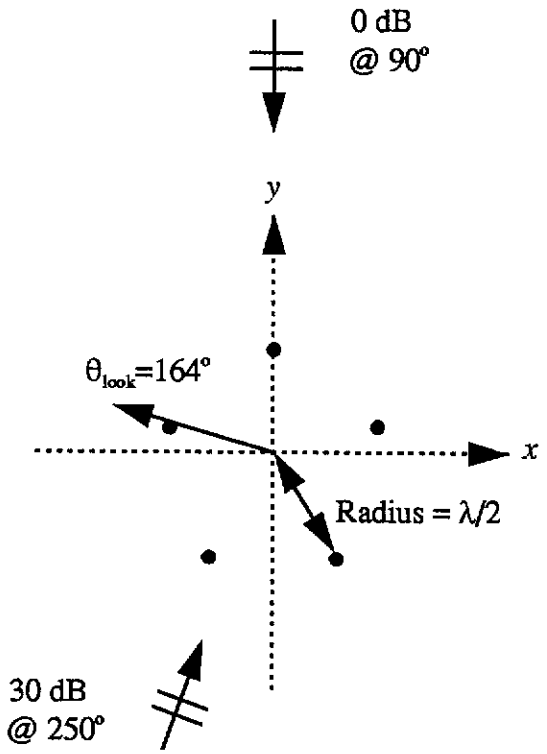


Figure 4.7 Array Structure and Source Scenario

Additional simulation parameters include:

- Forgetting factor, $\lambda = 0.99$
- Initialisation constant, $\delta = 100$
- Source and interference signals modelled as zero mean, Gaussian distributed sequences.
- Element self noise = -20 dB (also modelled as zero-mean, Gaussian distributed signal).
- Array phase centre located at the centre of the circle.

Fig. 4.8 shows the instantaneous output power of the two array processors ensemble averaged over 50 separate simulation runs.

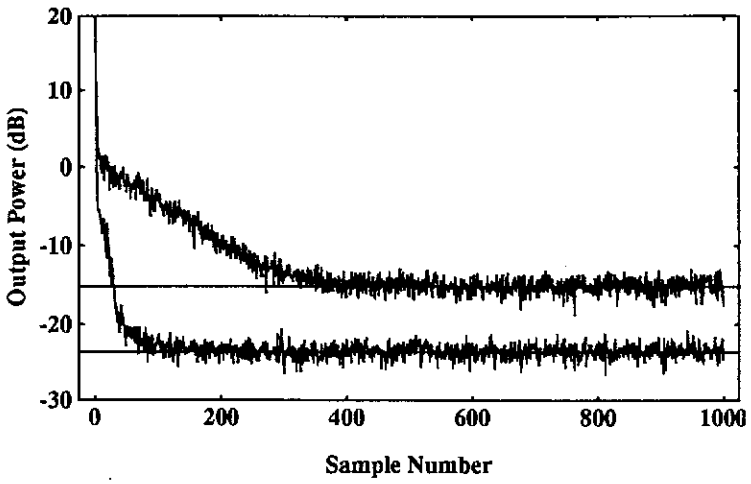


Figure 4.8 Ensemble Averaged Output Power for the New Array Processor (lower curve) and for the Array Processor with Approximate, Linear 2nd Order Derivative Constraints (upper curve)

Note from Fig. 4.8 that the new processor converges to an output power of -23.63 dB which can be found from the corresponding optimum processor. The processor with the approximate linear constraints, however, is not able to suppress the noise as effectively and converges to a value of -15.15 dB (about 8.5 dB higher). (The optimum value of -15.15 dB can also be found from the corresponding optimum processor.)

4.3.3 Numerical Stability

In the final set of studies, the effects of finite precision implementation on the algorithm are investigated. Here we simulate the effect of a fixed length mantissa in a floating point processor by rounding the mantissa results from all calculations to 3 decimal places. The simulation scenario is detailed in Fig.4.9.

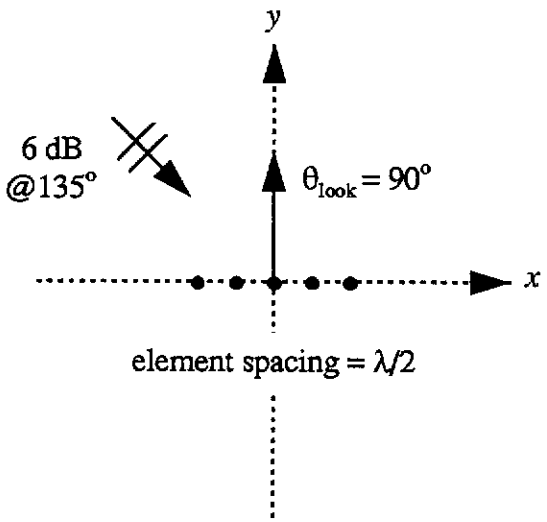


Figure 4.9 Array Structure and Source Scenario.

Additional simulation parameters include:

- Floating point accuracy fixed at 3 decimal places (mantissa).
- Forgetting factor, $\lambda = 0.95$
- Initialisation constant, $\delta = 10$
- Source and interference signals modelled as zero mean, Gaussian distributed sequences.
- Element self noise = -30 dB (also modelled as zero-mean, Gaussian distributed signal).
- Array phase centre located at the origin.

Fig. 4.10 shows the instantaneous output power of the array for a single run. Figs. 4.11 and 4.12 give the corresponding angle parameters for the inverse correlation matrix update and the $Q(n)$ matrix update respectively.

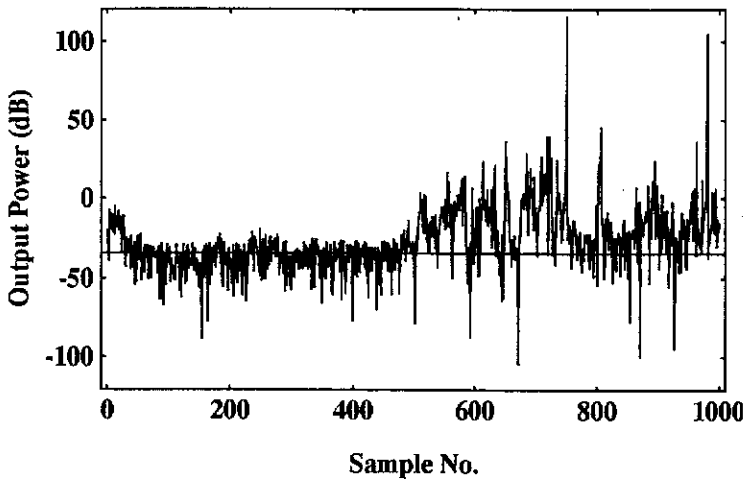


Figure 4.10 Instantaneous Array Output Power.

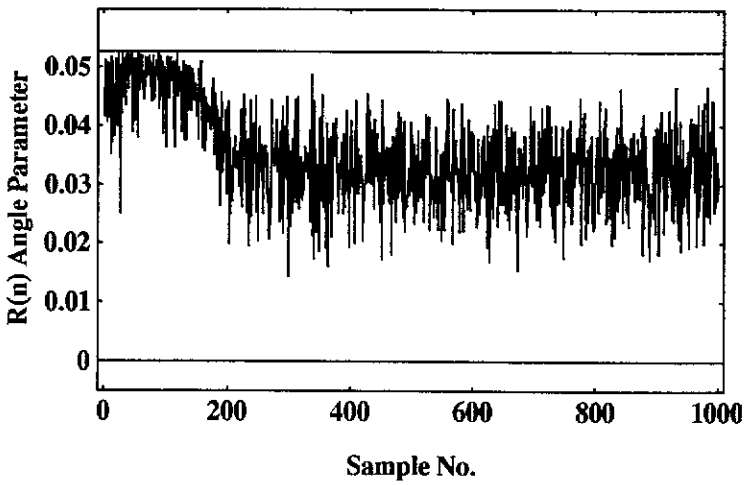


Figure 4.11 Angle Parameter $\beta(n)$ of the Inverse Correlation Matrix Update (Equation (4-29)).

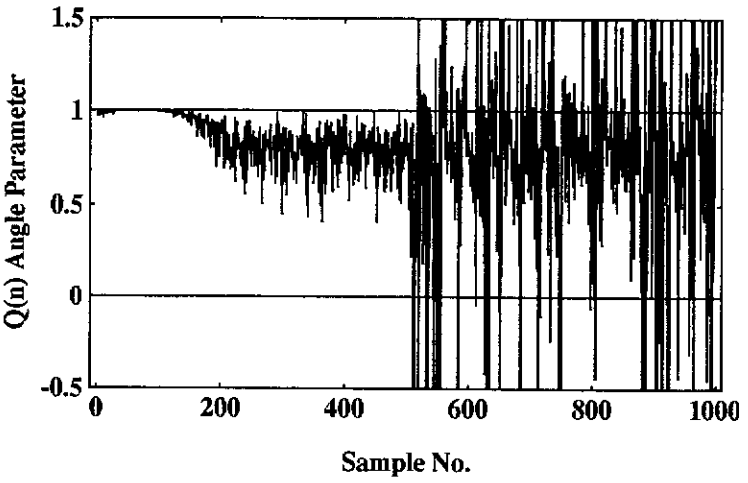


Figure 4.12 Angle Parameter $\gamma(n)$ of the $Q(n)$ Matrix Update (Equation (4-32)).

Note from the output power plot of Fig. 4.10, the new algorithm becomes unstable after about 500 iterations. The inverse correlation matrix update remains, however, stable as shown in Fig. 4.11. This results from the rounding rules discussed in Section 4.2.1. The instability in this simulation is a result of an error propagation mechanism within the $Q(n)$ matrix update as discussed in Section 4.2.1. In Fig. 4.12 one can see that the onset of instability is marked by the angle parameter $\gamma(n)$ exceeding its bound.

Note the severe numerical precision environment assumed in this simulation, i.e., 3 decimal places of floating point precision, is unlikely in a practical application. It was chosen only to illustrate the potential for numerical instability in a low number of algorithm iterations. In a higher precision environment, the onset of instability may simply be delayed if measures are not taken to stabilise the algorithm.

As indicated in the last part of Section 4.2.1, the numerical stability of the algorithm can be improved significantly by modifying the recursion equations for the $Q(n)$ matrix. Alternatively, an adaptive algorithm can monitor the angle parameter $\gamma(n)$ to detect the onset of instability and reset the algorithm.

4.4 References

[1] L. S. Resende, J. M. T. Romano and M. G. Bellanger, "A Robust FLS Algorithm For Linearly-Constrained Adaptive Filtering", *Proc. ICASSP*, Adelaide, Australia, pp. III 381- III 384, April 1994.

[2] G E Bottomley, S T Alexander, "A Novel Approach for Stabilising Recursive Least Squares Filters", *IEEE Trans. Signal Processing*, vol. 39, no. 8, pp. 1770-1779, August 1991.

[3] M. G. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker, Inc., New York, 1987.

[4] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1991.

[5] L. S. Resende, J. M. T. Romano and M. G. Bellanger, "A Fast Least-Squares Algorithm for Linearly -Constrained Adaptive Filtering", Submitted for publication in *IEEE Trans. Signal Processing*, August 1994.

[6] O. L. Frost, III, "An Algorithm for Linearly Constrained Adaptive Array Processing", *Proc. IEEE*, vol. 60, no. 8, pp. 926-935, August 1972.

[7] M. H. Verhaegen, "Round-off Error Propagation in Four Generally Applicable, Recursive, Least-squares Estimation Schemes", *Automatica*, vol. 25, no. 3, pp. 437-444, 1989.

[8] G. E. Bottomley and S. T. Alexander, "A Theoretical Basis for the Divergence of Conventional Recursive Least Squares Filters", *Proc. 1989 IEEE ICASSP* (Glasgow, Scotland), pp. 908-911, May 1989.

[9] I. Thng, A. Cantoni and Y. H. Leung, "Constraints for Maximally Flat Optimum Broadband Antenna Arrays", *ASPL Report 1994-2*, Curtin University of Technology, Bentley WA 6021, Australia, 1994. (Accepted for publication IEEE Trans. Sig. Proc.)

[10] MK. Low, "Numerically Robust, Linearly Constrained Recursive Least Squares (RLS) Filters", *Honours Thesis submission*, Department of Electrical and Electronic Engineering, University of Western Australia, WA, Australia, 1995

5. *Polynomial Root Solving*

In this chapter, we investigate a class of techniques, known as *Global methods*, for estimating all of the roots of a polynomial simultaneously. These methods are of particular interest here since, in step 2 of Table 4.1, the roots of a polynomial, whose coefficients are being continually updated, must be computed or tracked at each data sampling instant.

The chapter is organised into five sections. Firstly, the general concepts behind global methods are introduced and are contrasted against the better known conventional or *local* methods for root finding. The next two sections briefly summarise two distinctly different global methods which have proved to be particularly useful in the array processing problems studied here. In Section 4, we consider some issues relating to the practical implementation of these methods. The discussion in the fifth section centres on the application of the global methods to the particular polynomial factorisation problem we are faced with in this thesis. Finally, computer studies highlight some practical aspects of these methods with emphasis on their application in array processing problems.

5.1 Introduction

There appears to be two distinct classes of methods which may be applied to the problem of finding the roots of univariate polynomials. The first, often called conventional or local methods, locate just one root at a time in the iteration process. The second, known as global methods, provide successively better approximations to all of the roots simultaneously as the iteration procedure progresses.

Local methods generally involve making an initial guess at a root then iterating an algorithm which makes successively better approximations to the root until a desired accuracy is reached. If more than one root is required then the root already obtained is used to *deflate* or reduce the order of the polynomial and the iteration process is re-started using the deflated polynomial to find another root. Iteration functions which estimate one root at a time include the well known Secant algorithm [1, p.5] and Newton's method [1, p.5]. In the case of complex roots, Muller's method [2, p.74] is applicable.

The process of deflation is prone, however, to numerical problems since any inaccuracies in the estimate of a root will accumulate as the deflation process continues. For this reason, individual roots must be estimated to the maximum accuracy one at a time. One must also be careful that the roots are estimated in an increasing order of magnitude to minimise round-off-error growth [2, p.79].

Global methods, on the other hand, begin with a set of initial estimates to all of the roots and an iteration procedure is entered which makes successively better approximations to all of these roots simultaneously. Generally speaking, global methods are computationally more complex than local methods but have a number of significant advantages over local methods in certain applications.

Consider the scenario we are concerned with in this thesis where the roots of a polynomial whose coefficients make small changes in time are required. This is a common situation in many signal processing applications [5]. If one were to use a local method to find the roots, then the entire factorisation process, including deflation, would need to be restarted each time the coefficients of the polynomial are

updated, even for slight changes in the coefficients. In contrast, global methods exhibit fast convergence in the vicinity of the roots. Therefore, if the coefficients do not change by large amounts from one update instant to the next, then a global root finding algorithm, supplied with a set of the previous root estimates as starting values, may only have to be iterated once or twice to converge to the new roots. This is assuming, of course, that the polynomial does not exhibit *pathological* characteristics where small changes in the coefficients correspond to large changes in the roots.

In the sections that follow, we will be considering the general, complex valued rth order polynomial in z

$$f(z) = z^r + a_1 z^{r-1} + \cdots + a_{r-1} z + a_r \tag{5-1}$$

with roots $\lambda_k \in C$, $k \in Z_r^+$ such that

$$f(z) = (z-\lambda_1)(z-\lambda_2)\ldots(z-\lambda_r). \tag{5-2}$$

In addition, we will consider only the class of polynomials having real coefficients, i.e., $a_i \in \mathcal{R}$, since in our array processing problem, the coefficients of the polynomials are derived from the elements of a real matrix (see Step 2 in Table 4.5). We shall denote the set of initial estimates to the roots of the polynomial by $\{\lambda_k(0) \in C, k \in Z_r^+\}$ and the set of root estimates at the nth iteration of the process by $\{\lambda_k(n) \in C, k \in Z_r^+\}$.

In the next two sections, we will consider the two distinct approaches to deriving global methods which have appeared in the literature. The first approach, based on the Taylor series expansion of $f(z)$ about $\lambda_k(n)$, results in a class of similar techniques offering different orders of convergence. The second approach uses a technique known as coefficient matching to derive the global algorithm.

5.2 *Global Methods based on Taylor Series Expansion*

A class of similar techniques offering different orders of convergence can be derived from the Taylor series expansion of $f(z)$ about $\lambda_k(n)$ [3]. A brief derivation of these methods is presented in Appendix D. Expressions for the second up to the sixth order methods based on this derivation are presented below. For an insight into the origins of these methods and an analysis of their characteristics, the reader is referred to [4].

To begin with we make the following definitions:

$$u = u(\lambda_k(n)) = \frac{f(\lambda_k(n))}{f'(\lambda_k(n))} \tag{5-3}$$

$$A_m = A_m(\lambda_k(n)) = \frac{f^{(m)}(\lambda_k(n))}{m!\, f'(\lambda_k(n))} \tag{5-4}$$

$$T_m = T_m(\lambda_k(n)) = \sum_{i=1;i\neq k}^{r} \frac{1}{(\lambda_k(n) - \lambda_i(n))^m} \tag{5-5}$$

where the prime, $f'(z)$, indicates the first derivative of $f(z)$ with respect to z. The definitions (5-3) and (5-4) were originally presented in Traub [1], while (5-5) appeared in Farmer and Loizou [3].

Second Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{f(\lambda_k(n))}{\prod_{i=1;i\neq k}^{r} (\lambda_k(n) - \lambda_i(n))} \tag{5-6}$$

Third Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{u}{1 - T_1 u} \tag{5-7}$$

Fourth Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{(1 - A_2 u)u}{1 - 2A_2 u + 0.5\left(A_2^2 - T_2\right)u^2} \tag{5-8}$$

Fifth Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{\left(1 - 2A_2u + A_3u^2\right)u}{1 - 3A_2u + \left(2A_3 + A_2^2\right)u^2 - \left(3A_2^2A_3 - A_2^3 + T_3\right)u^3/3} \quad (5\text{-}9)$$

Sixth Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{[NUMERATOR]}{[DENOMINATOR]}$$

where

$$[NUMERATOR] = \left(1 - 3A_2u + \left(2A_3 + A_2^2\right)u^2 - A_4u^3\right)u \quad (5\text{-}10)$$

$$[DENOMINATOR] = 1 - 4A_2u + \left(3A_3 + 3A_2^2\right)u^2 - \left(2A_2A_3 + 2A_4\right)u^3 + \left(A_2^4 - 4A_2^2A_3 + 4A_2A_4 + 2A_3^2 - T_4\right)u^4/4$$

All of the above iteration functions can be regarded as Jacobi like methods [6] whereby they are used to generate successive approximations to the roots in *parallel* and do not make use of the latest available root estimates computed in the current iteration. It is advantageous to modify the above methods slightly to make use of the better approximations as they become available in the evaluation of the next root estimate. The iteration functions then become more of a *serial* or Gauss-Seidel like [6] global method. As an example consider the second order method of (5-6). The modified, Gauss-Seidel like, iteration function is given by

$$\lambda_k(n+1) = \lambda_k(n) - \frac{f(\lambda_k(n))}{\prod_{i=1}^{k-1}(\lambda_k(n) - \lambda_i(n+1)) \prod_{i=k+1}^{r}(\lambda_k(n) - \lambda_i(n))}. \quad (5\text{-}11)$$

Not only does this improve the convergence rate of the methods, but it also avoids some problems that can occur if symmetry exists on the complex plane between the initial root estimates and the true polynomial roots [4].

Another point worth noting from the above algorithm descriptions is that the polynomial, given by (5-1), and some of its derivatives will have to be evaluated frequently. It is therefore desirable to do this as efficiently as possible. It is inefficient to evaluate each of the r terms in (5-1) separately and then to sum. This would take r

additions and $r(r+1)/2$ multiplications. Rather, noting that each term in (5-1) bar the first contains the factor z, (5-1) can be rewritten in *nested form* [2]:

$$f(z) = a_r + z\Big(a_{r-1} + z\big(a_{r-2} + \cdots + z(a_1 + z)\cdots\big)\Big). \tag{5-12}$$

Evaluating the polynomial in this way takes only r additions and r multiplications.

5.3 Global Method based on Coefficient Matching

An entirely different approach to the global polynomial factorisation problem was presented by Starer and Nehorai [5]. Their algorithm attempts to minimise the error between the actual polynomial coefficients and a corresponding set of coefficients derived from the current root estimates. The method exhibits a quadratic rate of convergence close to the true roots, and its convergence behaviour is comparable to the second order global method of the previous section.

We begin by constructing the r-dimensional real vector $\boldsymbol{a}$ from the polynomial coefficients $a_k, k \in Z_r^+$ given in (5-1)

$$\boldsymbol{a} = [a_1, \quad a_2, \quad \cdots \quad , a_r]^T. \tag{5-13}$$

Let the set of root estimates at iteration n be represented by the r-dimensional complex vector

$$\underset{\sim}{\lambda}(n) = [\lambda_1(n) \quad \lambda_2(n) \quad \cdots \quad \lambda_r(n)]^T \tag{5-14}$$

and define $\boldsymbol{a}\big(\underset{\sim}{\lambda}(n)\big) \in C^r$ to be the vector of polynomial coefficients derived from these root estimates.

The algorithm essentially minimises a cost function which measures the square of the difference between the actual polynomial coefficients and those derived from the current root estimates, i.e.,

$$V\big(\underset{\sim}{\lambda}\big) = \varepsilon^H\big(\underset{\sim}{\lambda}\big)\varepsilon\big(\underset{\sim}{\lambda}\big) \tag{5-15}$$

where H denotes the conjugate transpose and the error vector is given by

$$\underline{\varepsilon}(\underline{\lambda}) = \underline{a} - \underline{a}(\underline{\lambda}). \tag{5-16}$$

Note that the iteration index, n has been omitted for simplicity of presentation.

In [5] it is shown that successively better approximations to the roots of the polynomial can be made using the following update relation

$$\underline{\lambda}(n+1) = \underline{\lambda}(n) - [\boldsymbol{T}\boldsymbol{\Lambda}]^{-1}\underline{\varepsilon}(\underline{\lambda}(n)) \tag{5-17}$$

where $\boldsymbol{T} \in C^{r\times r}$ is a lower triangular Toeplitz matrix defined by

$$\boldsymbol{T} = \begin{bmatrix} 1 & & & 0 \\ a_1(\underline{\lambda}) & 1 & & \\ \vdots & \ddots & \ddots & \\ a_{r-1}(\underline{\lambda}) & \cdots & a_1(\underline{\lambda}) & 1 \end{bmatrix} \tag{5-18}$$

and $\boldsymbol{\Lambda} \in C^{r\times r}$ is the Vandermonde matrix given by

$$\boldsymbol{\Lambda} = \begin{bmatrix} 1 & \cdots & 1 \\ \lambda_1(n) & \cdots & \lambda_r(n) \\ \vdots & & \vdots \\ \lambda_1^{r-1}(n) & \cdots & \lambda_r^{r-1}(n) \end{bmatrix}. \tag{5-19}$$

Since the matrices $\boldsymbol{T}$ and Λ are highly structured, efficient methods are available to compute the matrix inversion in (5-17).

Let $\boldsymbol{x}$ represent the root vector update term in (5-17)

$$\boldsymbol{x} = [\boldsymbol{T}\boldsymbol{\Lambda}]^{-1}\boldsymbol{\varepsilon} = \boldsymbol{\Lambda}^{-1}\boldsymbol{T}^{-1}\boldsymbol{\varepsilon}. \tag{5-20}$$

The vector $\boldsymbol{x}$ can be found by solving for $\boldsymbol{y}$ in

$$\boldsymbol{T}\boldsymbol{y} = \boldsymbol{\varepsilon} \tag{5-21}$$

followed by solving for $\boldsymbol{x}$ in

$$\boldsymbol{\Lambda}\boldsymbol{x} = \boldsymbol{y}. \tag{5-22}$$

Recalling that $\boldsymbol{T}$ is lower triangular, $\boldsymbol{y}$ can be found by simple forward substitution. The solution of the Vandermonde system of (5-22) can be computed efficiently using

the method of Björck and Pereyra [6] which uses a technique equivalent to polynomial interpolation.

Therefore, the root update of (5-17) may proceed in the following manner: (i) with the root estimates available at iteration n compute the vector of corresponding polynomial coefficients $\boldsymbol{a}\left(\underset{\sim}{\lambda}(n)\right)$ and hence the error vector $\boldsymbol{\varepsilon}\left(\underset{\sim}{\lambda}\right)$ and the matrix $\boldsymbol{T}$; (ii) determine $\boldsymbol{y}$ in (5-21) using forward substitution; (iii) find the total update term $\boldsymbol{x}$ by solving the Vandermonde system of (5-22); and finally (iv) update the root estimate vector, $\underset{\sim}{\lambda}(n+1)$ using (5-17).

Starer and Nehorai [10] give the following simple recursive procedure for finding polynomial coefficients corresponding to the root estimates $\underset{\sim}{\lambda}(n)$ in Step (i) above.

Firstly an $(r+1)\times(r+1)$ matrix $\boldsymbol{A}_{i,j}$ must be initialised with the elements of the first row set to 1 and the rest to 0. The following recursion is then applied

$$\begin{aligned} &\text{for } i = 2,\ldots,j \\ &\quad \text{for } j = 2,\ldots,(r+1) \\ &\qquad \boldsymbol{A}_{i,j} = \boldsymbol{A}_{i,j-1} - \lambda_{j-1}(n)\boldsymbol{A}_{i-1,j-1} \end{aligned} \qquad (5\text{-}23)$$

On completion of this procedure the coefficients of the polynomial corresponding to the roots, $\underset{\sim}{\lambda}(n)$, appear in the last column of $\boldsymbol{A}$ as $a_i\left(\underset{\sim}{\lambda}(n)\right) = \boldsymbol{A}_{i+1,r+1},\ i \in Z_r^+$.

5.4 Notes on Implementation

In this section, we briefly consider three aspects relating to the practical implementation of the global methods just described. For a more detailed study of these issues the reader is referred to [4].

5.4.1 Complex Roots

On inspection of any of the global methods (5-6) to (5-10) and the coefficient matching method of (5-17), one can see that if the coefficients of the polynomial are all real and if the algorithm is started with initial guesses that are all real, then the algorithm can never resolve complex roots. This is because none of the algorithms

contain a square root term and, consequently, cannot create a complex component. In addition, it has been found in practice that the global methods of Section 5.2 (equations (5-6) to (5-10)) may fail to converge whenever symmetry exists between the actual roots and the initial guesses [4, 8, 9]. That is, if the actual roots of the polynomial are symmetrically distributed about some line L on the complex plane and the initial guesses are so chosen that they are also symmetrically placed with respect to this line. A characteristic of the algorithms presented in Section 5.2 is that successive approximations preserve their symmetry with respect to L. By modifying these Jacobi-like methods to their Gauss-Seidel equivalents as suggested in the last part of Section 5.2, the problems of symmetry are circumvented.

Therefore, if nothing is known about the nature of the polynomial roots, and if either the method of coefficient matching or the Gauss-Seidel global methods are implemented, then the algorithms should be initialised with guesses that contain at least a complex component to ensure convergence to complex roots if they exist. This can be done by simply adding a small complex component onto one of the root estimates prior to starting the root finding algorithm.

5.4.2 Multiple Roots

The presence of multiple roots can dramatically effect the convergence properties of the algorithms. It is clear on inspection of any of the methods above that the initial guesses to the roots must be distinct or the algorithms will fail. Even so, the algorithms are still capable of locating roots with multiplicity greater than one, albeit with a large reduction in the rate of convergence [7, 9].

5.4.3 Computational Complexity

Table 5.1 gives a comparison of the number of floating point operations (FLOPS) required by each algorithm to complete one update of the set of root estimates for a polynomial of order r.

Computation Type	Algorithm					
	2nd Order	3rd Order	4th Order	5th Order	6th Order	Coefficient Matching
$+/-$	$2r^2$	$4r^2-2r$	$5r^2-r$	$6r^2+11r$	$7r^2+4r$	-
$\times$	$2r^2-2r$	$2r^2$	$4r^2+11r$	$6r^2+13r$	$7r^2+25r$	-
$\div$	r	r^2	r^2+r	r^2+3r	r^2+4r	-
Total FLOPS	$4r^2-r$	$7r^2-2r$	$10r^2+11r$	$13r^2+27r$	$15r^2+33r$	$5.5r^2+3.5r$

Table 5.1. A Comparison of the Computational Complexity of the Algorithms

5.5 Discussion

From the many polynomials and simulation scenarios used in studying the global methods, it is the author's experience that the third order Taylor series method of (5-4) and the second order algorithm of Starer and Nehorai (5-17) provide the most reliable, efficient and stable techniques for simultaneously finding all the roots of a polynomial. The second order Taylor series method of (5-3) often exhibited instability for high order polynomials and appeared to be very sensitive to the choice of initial root estimates, while the fourth, fifth and sixth order Taylor series methods of (5-5) to (5-10) exhibited a problem described in [7] and [8] where, for a polynomial with distinct roots, adjacent root estimates can move towards the same root and so the method fails to find all the roots.

In the case of the array processing application of interest here, involving root finding in time-varying polynomials and where the initial root estimates are usually quite good we found, in addition, that the third order Taylor series method of (5-4) proved to be more applicable than Starer and Nehorai's algorithm of (5-17). The third order method gave the best results as it is faster converging and computationally less expensive than Starer and Nehorai's algorithm. For fixed coefficient polynomials*,

---

*By fixed coefficient we mean that the coefficients are not changing with time.

however, the algorithm of Starer and Nehorai appeared to be the best choice as the technique of coefficient matching makes the algorithm more robust to poor initial root estimates.

5.6 Computer Studies

The following computer studies illustrate the application of the two global methods discussed above in, firstly, root finding in fixed coefficient polynomials, and secondly, in root tracking where the coefficients of the polynomial are changing with time.

In the first of the computer studies we compare the convergence properties of the third order global method of (5-4) with the method of Starer and Nehorai, equation (5-17). This comparison is made using a *fixed coefficient*, fifth order polynomial given in (5-24).

$$f(z) = z^5 + z^4 - 15z^3 - 45z^2 + 74z + 104. \qquad (5\text{-}24)$$

This polynomial has the exact roots $\{-3 \pm 2i, 1, 2, 4\}$ shown in the plots in Fig. 5.1 by triangles. The initial estimates were chosen at regular intervals on a circle of radius 5.0 on the complex plane (large dots). The plots show the trajectories of the root approximations as the algorithms converge.

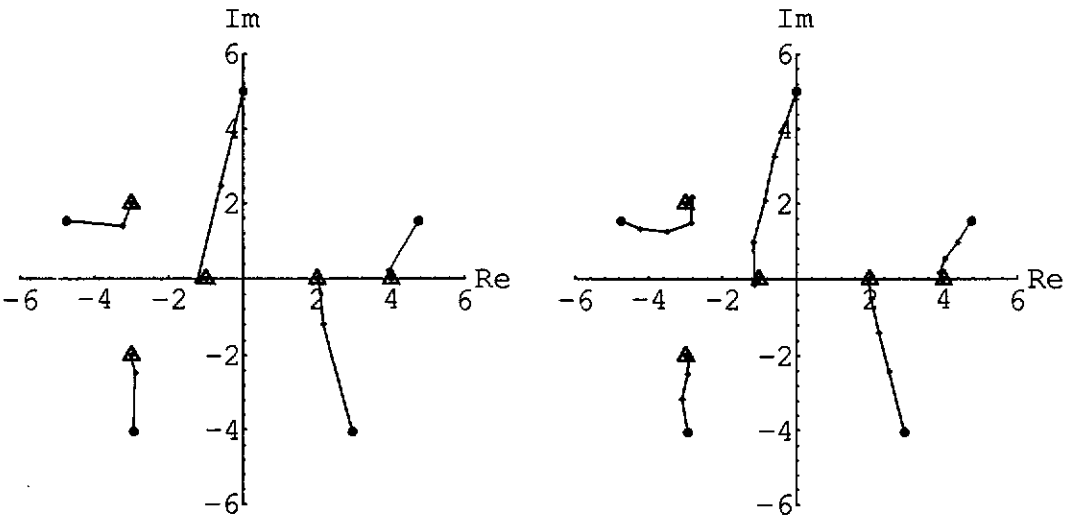


Figure 5.1 Root Estimate Trajectories on the Complex Plane
(a) Gauss-Seidel Third order Global Method
(b) Starer and Nehorai's Method of Coefficient Matching

Note that the third order global method was modified slightly from (5-4) to make use of the Gauss-Seidel approach as suggested at the end of Section 5.3.1. As one can see from these plots, the modified third order global method on the left converges much faster than Starer and Nehorai's method shown on the right. The convergence information is shown in Table 5.2.

Global Method	Iterations	Relative Calculation Time
Modified 3rd order	4	1.0
Starer and Nehorai	8	1.9

Table 5.2 Convergence information for Fig. 5.1

The "Iterations" column gives the number of algorithm cycles required to obtain eight decimal places of accuracy in all of the polynomial roots.

The next two computer studies illustrate the root tracking capabilities of global methods. In the first of the studies we use the tenth order time-varying polynomial with which Starer and Nehorai tested their root tracking algorithm. The polynomial has fixed roots at $\pm 0.6 \pm j0.4$ and $\pm 0.2 \pm j0.8$ and a pair of time varying roots at

$0.7e^{\pm\varphi(t)}$ where $\phi(t)$ is a parameter which varies sinusoidally with time and is given by $\phi(t) = 1 + 0.5\sin(28.8t)$.

In this example we compare the tracking performance of the Gauss-Seidel third order algorithm with that of the coefficient matching technique of Starer and Nehorai. In the first part of this computer study, both algorithms were initialised with the true roots at time $t = 0$ and were iterated only once for each new sample of the polynomial coefficients. The results are shown in Figs. 5.2 and 5.3. In Fig. 5.3, the solid line is the true value of $\phi(t)$ in radians, the triangles are the values of $\phi(t)$ computed from the root estimates provided by the Gauss-Seidel third order algorithm and the diamonds are the values of $\phi(t)$ computed from the root estimates provided by Starer and Nehorai's technique. To appreciate relative performance of the two techniques, Fig. 9 shows the square of the difference between the true values of $\phi(t)$ and those computed using the third order method (solid line) and those computed using the coefficient matching technique (dashed line).

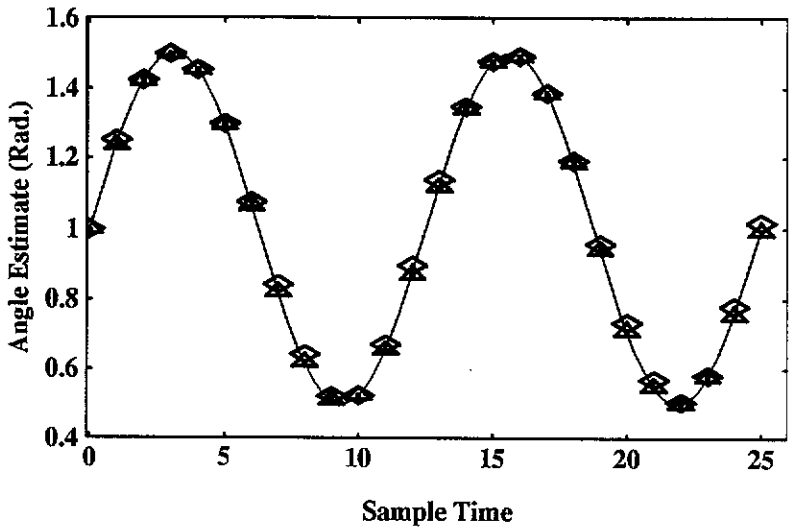


Figure 5.2 Root tracking example

Solid line - true value of the parameter $\phi(t)$

Triangles - estimated value of $\phi(t)$ from 3rd order method

Diamonds - estimated value of $\phi(t)$ from coefficient matching technique

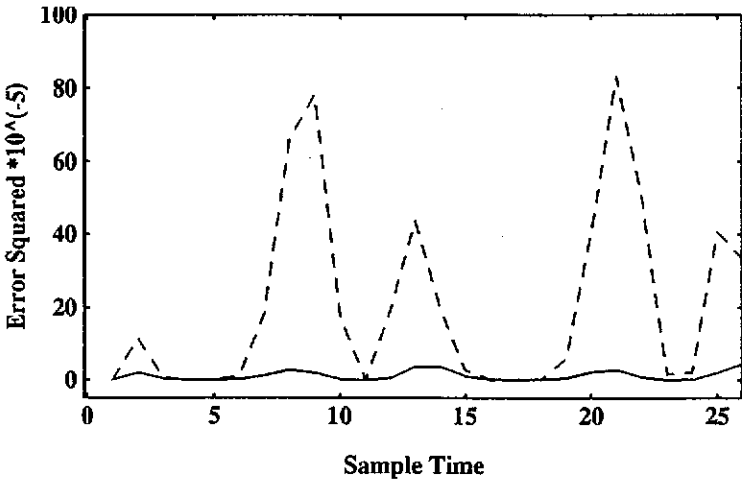


Figure 5.3 Tracking Error

Solid line - Square of the error between the true $\phi(t)$ and that computed using the 3rd order method

Dashed line - Square of the error between the true $\phi(t)$ and that computed using the coefficient matching technique

In the second part of this example, both algorithms were again initialised with the true roots at time $t = 0$ but for each new sample of the polynomial coefficients, the algorithms were iterated until an accuracy of eight decimal places was obtained in all of the polynomial roots. Fig. 5.4 shows the number of iterations required by the algorithms at each sample instant to find the new polynomial roots. The solid line corresponds to the Gauss-Seidel third order method and the dashed line to the method of coefficient matching.

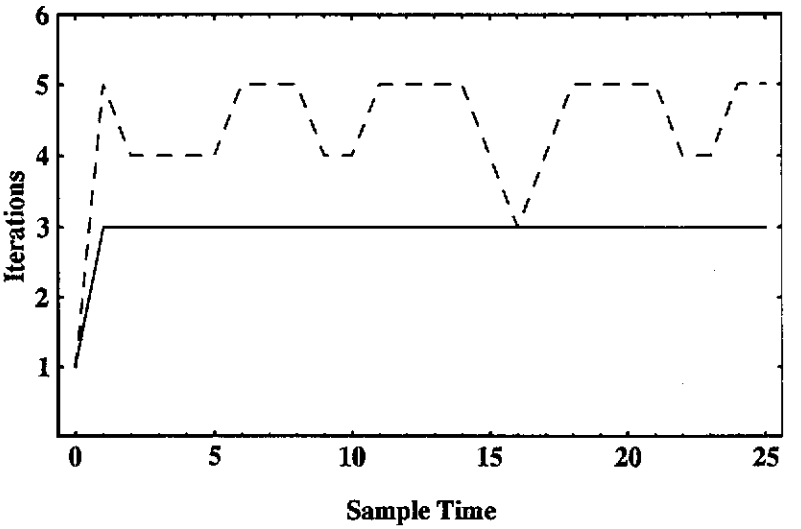


Figure 5.4 Iterations Required to Update the Polynomial Roots to an Accuracy of 8 Decimal Places

Solid line - Gauss-Seidel 3rd order method

Dashed line - Method of coefficient matching

It is apparent from Fig. 5.3 that, for both algorithms, a single iteration is sufficient to accurately track the roots. However, from Fig. 5.4 we see that the Gauss-Seidel third order method generally provides a significantly more accurate estimate of the changing polynomial roots. This is attributed to the faster convergence characteristic of this method as demonstrated in Fig. 5.4. Here we see that each time the polynomial coefficients change, the third order method generally takes one or two iterations less than the technique of coefficient matching to estimate all of the roots to eight decimal places. It is for these reasons that the Gauss-Seidel third order method was chosen as the preferred method for polynomial root tracking applications.

In the second of the root tracking computer studies, we demonstrate the application of the preferred third order method in tracking the roots of the third order polynomial $P'(\alpha)$, given in Step 2 of Table 4.1. The array structure and source scenario is shown in Fig. 5.5.

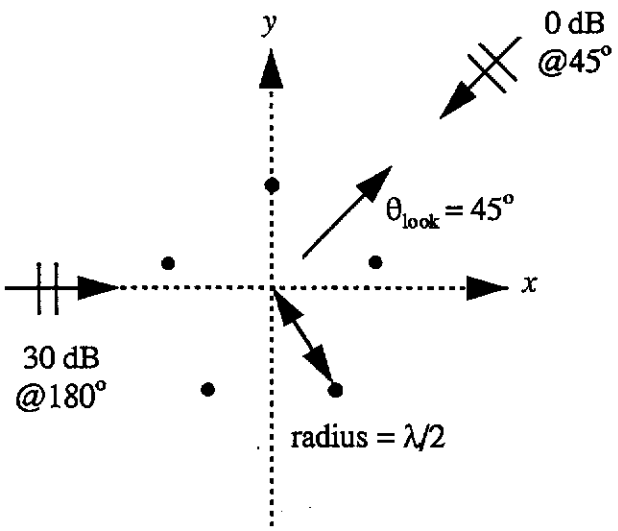


Figure 5.5 Array Structure and Source Scenario

Additional simulation parameters include:

- Forgetting factor, $\lambda = 0.99$.
- Initialisation constant, $\delta = 100$.
- Source and interference signals modelled as zero mean, Gaussian distributed sequences.
- Element self noise = -20 dB (also modelled as zero-mean, Gaussian distributed signal).
- Array phase centre located at the centre of the circle.

Figs. 5.6 and 5.7 show, respectively, the instantaneous output power of the array and the optimum value of the parameter α at each sample instant as the array processor adapts to the signal environment. In Fig. 5.8, the number of iterations at each coefficient update instant (i.e., at each input data sample instant) required by the modified third order global method to track the three polynomial roots of $P'(\alpha)$ to an accuracy of four decimal places is shown.

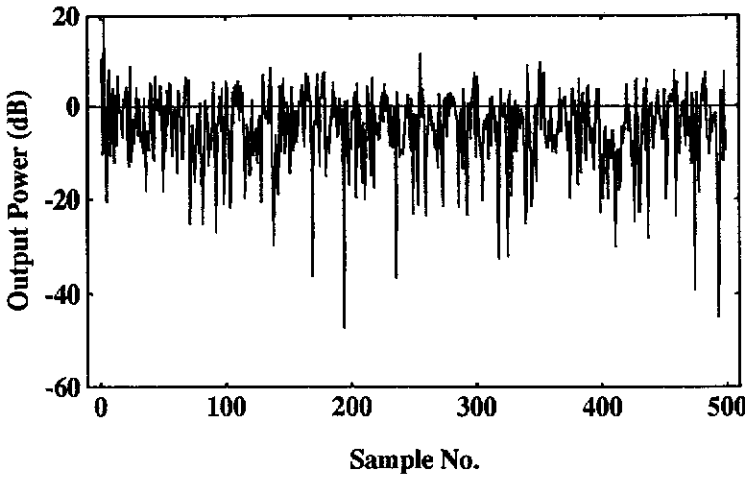


Figure 5.6 Instantaneous Array Output Power

(Computed optimum output power = 0 dB, shown as a horizontal line)

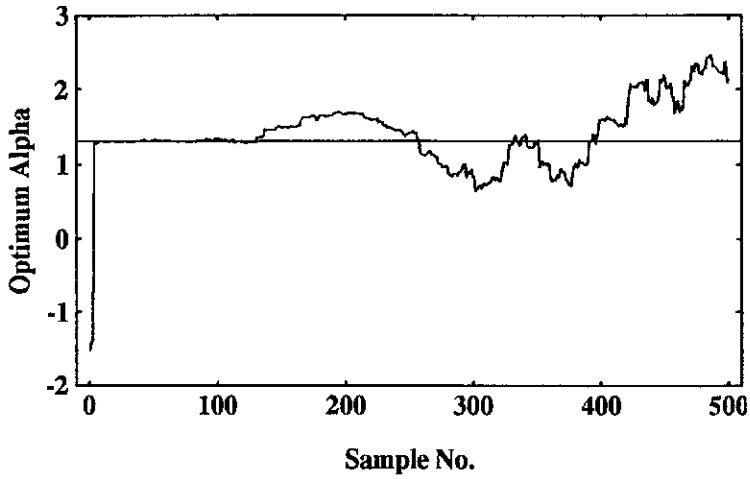


Figure 5.7 Instantaneous Optimum Value of the Parameter α

(Horizontal line shows the optimum value for α computed from the optimum array processor)

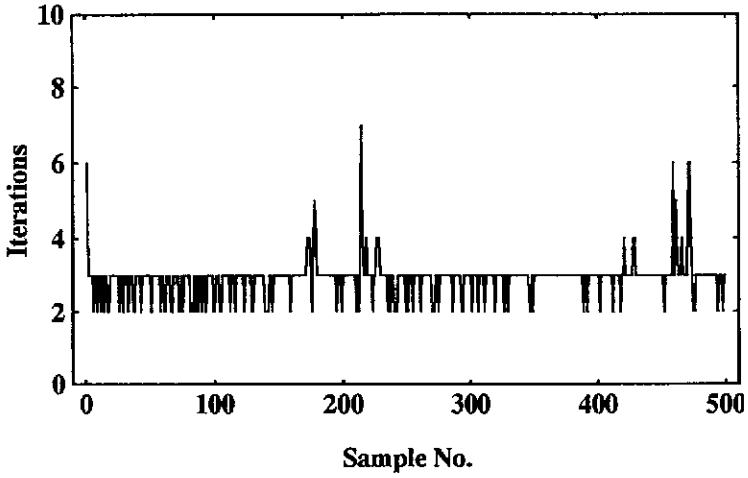


Figure 5.8 Number of Iterations Required to Update the Polynomial Roots to an Accuracy of 4 Decimal Places at Each Iteration

Note from Fig. 5.8 that the number of iterations required to update the roots to an accuracy of four decimal places is, on average, between 2 and 3 iterations. Moreover, in other similar computer studies conducted in the course of this research, it was found that for an accuracy of eight decimal places the number of iterations only rose to between 3 and 4 on average. The spikes that can be seen in Fig. 5.8 (near sample number 200 and between 400 and 500) correspond to all-real to real-plus-complex conjugate pair or real-plus-complex conjugate pair to all-real transitions in the roots. The additional iterations are required to either eliminate or develop the complex components within the algorithm. (Recall from the previous discussion that a small complex component is added into one of the initial estimates before the global method iteration is started to ensure convergence to complex roots, if they exist.)

5.7 References

[1] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice-Hall Series in Automatic Computation, Englewood Cliffs, 1964.

[2] S. D. Conte and C. de Boor, *Elementary Numerical Analysis : an algorithmic approach*, 2nd Edition, International Series in Pure and Applied Mathematics, McGraw Hill, New York, 1972.

[3] M. R. Farmer and G. Loizou, "A Class of Iteration Functions for Improving, Simultaneously, Approximations to the Zeros of a Polynomial", *BIT*, vol. 15, pp 250 - 258, 1975.

[4] J. Tuthill, "On Global Methods for Finding the Roots of a Polynomial Simultaneously", *Adaptive Signal Processing Laboratory Report SPL-TM-007*, ATRI, Curtin University of Technology, Bentley, Australia.

[5] D. Starer and A. Nehorai, "Adaptive Polynomial Factorisation by Coefficient Matching", *IEEE Trans. Signal Processing*, vol. 39, no. 2, pp. 527 - 530, February 1991.

[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Second Edition, Johns Hopkins, Baltimore, 1989.

[7] O. Aberth, "Iteration Methods for Finding all the Zeros of a Polynomial Simultaneously", *Mathematics of Computation*, vol. 27, no. 122, pp 339 - 344, April 1973.

[8] M. Igarashi, "Some Remarks for the Methods to Find All the Zeros of a Polynomial Simultaneously", *Topics in Polynomials of One and Several Variables and their Applications*, T. M. Rassias, H. M. Srivastava and A. Yanushauskas (Ed.), World Scientific Publ. Co., Singapore, pp. 273 - 285, 1993.

[9] R. D. Small, "Problem 75-14, Simultaneous Iteration Towards All Roots of a Complex Polynomial", Problems and Solutions section, *SIAM Review*, vol. 18, pp. 501-502, 1976.

[10] D. Starer and A. Nehorai, "Polynomial Factorisation Algorithms for Adaptive Root Estimation", *Proc. IEEE ICASSP*, D7.2, pp. 1158-1161, 1989.

6. The Multimodal Problem

In Chapters 1, 2 and 3, we have shown that the minimisation problem which results when look direction plus first and second order NS derivative constraints are used may have more than one "optimum" solution.

In this chapter we investigate the effects of this multimodal behaviour on both the optimum and adaptive problems. It must be said, though, that a full analysis of this complex issue is beyond the scope of this thesis. We simply present some preliminary findings based primarily on computer studies.

6.1 Summary of the Optimum Problem

In Chapter 2, we found that to minimise the mean output power of a narrowband array processor operating in a 2D scenario subject to look direction plus first and second order NS spatial derivative constraints, the following two-stage optimisation problem must be solved

$$\min_{\alpha}\left[\begin{array}{c}\min_{\boldsymbol{w}} P(\boldsymbol{w}) \\ \text{subject to} \quad \boldsymbol{C}^T\boldsymbol{w}=\boldsymbol{h}(\alpha)\end{array}\right] \tag{6-1}$$

where the first stage is a linearly constrained optimisation problem, parameterised by α, and the second stage is an unconstrained optimisation of α. We found that using the method of Lagrange multipliers the inner stage of the two-stage minimisation problem has solution

$$\boldsymbol{w}_{opt}(\alpha)=\boldsymbol{R}^{-1}\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{h}(\alpha) \tag{6-2}$$

and that the optimum value for α is found from the outer stage, given by

$$\min_{\alpha} P(\alpha)=\boldsymbol{h}^T(\alpha)\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{h}(\alpha). \tag{6-3}$$

We have shown that $P(\alpha)$ is a quartic polynomial in α, and so the optimum α, α_{opt}, can be found by simply computing the roots of the first derivative of $P(\alpha)$ with respect to α and identifying the root **or roots** which minimise $P(\alpha)$. We emphasise

here that since there may be more that one α_{opt}, there may also be more than one optimum weight vector $w_{opt}(\alpha_{opt})$.

6.2 *Effect on the Optimum Processor*

Consider a minimum variance narrowband antenna array processor with look direction plus first and second order NS derivative constraints as described above with array structure and source scenario as shown in Fig. 6.1.

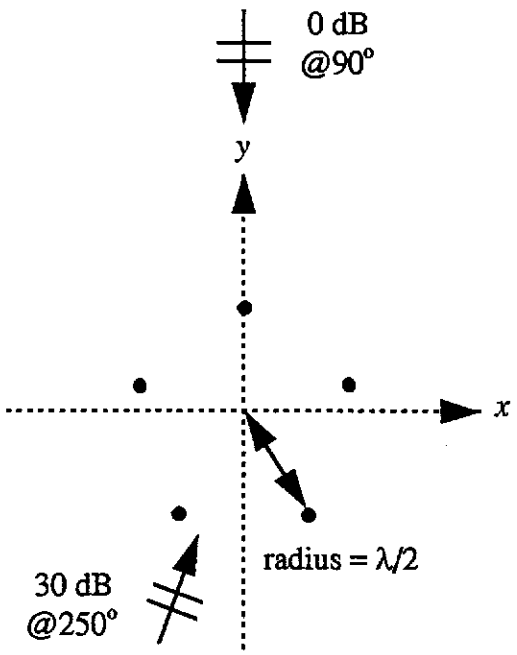


Figure 6.1 Array Structure and Source Scenario

The optimum output power of this processor as the look direction is swept from 0° to 360° is shown in Fig. 6.2.

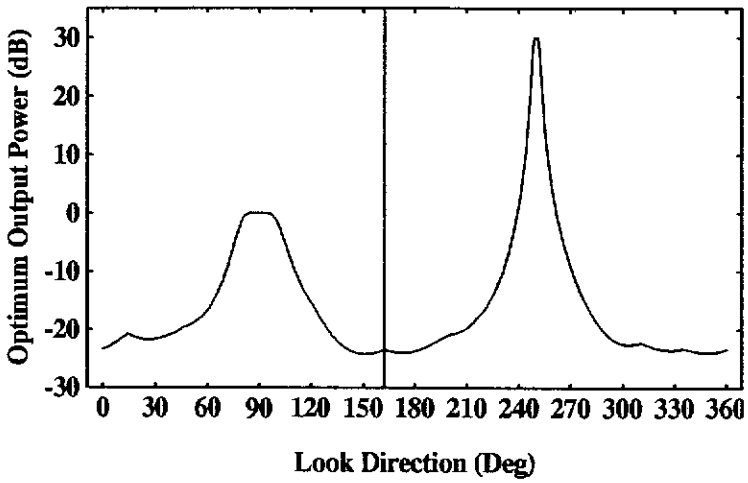


Figure 6.2 Optimum Output Power of the Array Processor

Evidence of the multimodal nature of the optimisation problem can be seen on close examination of Fig. 6.2. For example, at bearing angles of approximately 15°, 162° (marked by the vertical line) and 310°, cusps or discontinuities can be located.

The vertical line in Fig. 6.2 identifies the look direction $\theta_0 = 162.31^\circ$. At this look direction the quartic polynomial $P(\alpha)$ in (6-3) has two solutions for α which minimise $P(\alpha)$ equally. As a result, two equally optimum weight vectors can be computed using (6-2). The two values for α and the corresponding weight vectors and mean output power values are given below:

a) $\alpha_{opt1} = -2.104$, P_{opt} = -23.47 dB.

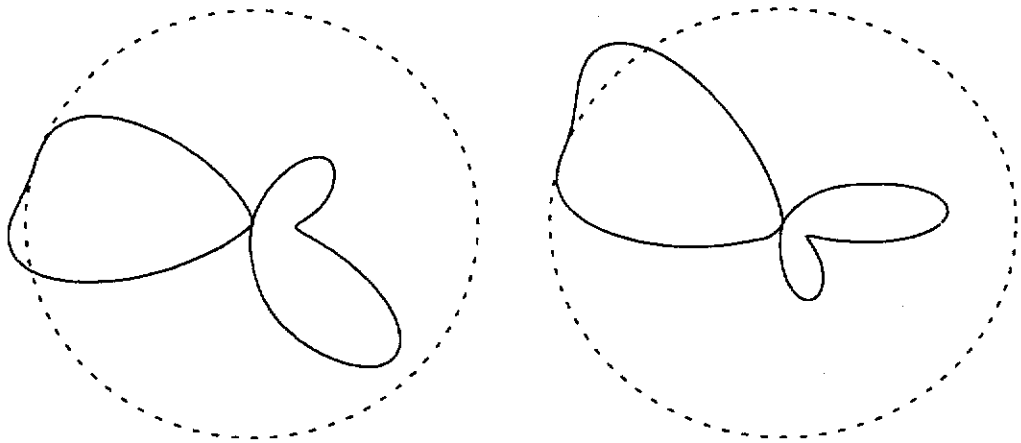
$$w_{opt}(\alpha_{opt1}) = \{-0.15, -0.18, 0.13, -0.36, -0.16, -0.09, 0.09, 0.42, -0.14, -0.09\}$$

b) $\alpha_{opt2} = 1.856$, P_{opt} = -23.47 dB.

$$w_{opt}(\alpha_{opt2}) = \{0.13, -0.22, -0.19, -0.21, -0.30, 0.35, 0.17, -0.09, -0.11, -0.20\}$$

One parameter of the array processor to be affected by multiple optimum solutions is its polar response or beam pattern. To appreciate the effect of the two optimum solutions on the beam pattern, we compare the two optimum processors designed using the two above choices of optimum weight vector. The polar response of the optimum array processor is obtained by plotting the magnitude response against

source direction as a unity magnitude signal source in the far field of the array is moved spatially around the array. The polar responses of optimum array processors corresponding to the two weight vectors in a) and b) above are shown in Fig. 6.3.



(a) (b)

Figure 6.3 Polar Response Plots (a) plot obtained using the weight vector in a) above. (b) plot obtained using the weight vector in b)

The dashed circles represent a magnitude response of 1.0, i.e., unity gain. Note from Fig. 6.3 that for both choices of optimum weight vector the array processor design criteria have been met, namely unity gain and second order flatness of the response in the look direction, $\theta_0 = 162.31^\circ$, and rejection of interference signals from all other directions, as can be seen from the deep nulls in both plots for the source directions of $\theta_{s1} = 90^\circ$ and $\theta_{s2} = 250^\circ$. Aside from these essential characteristics the two polar responses are quite different.

Notwithstanding, in terms of the original problem formulation, clearly both solutions are equally valid and the designer may arbitrarily choose one over the other or, alternatively, introduce some further constraint on the problem that identifies one as being the appropriate solution, for example, minimum weight vector norm or reduced side lobe levels in the response. (In this particular example, both weight vector norms have the same value of about 0.45 and so the former criterion could not be used here.)

When using a linear array structure with equally spaced antenna elements and operating in a 2D environment, the solutions to the two-stage optimisation problem of

(6-1) have some interesting properties. We found that regardless of the look direction, there are always two optimum solutions to (6-1). In simple cases of two, three and four element arrays aligned on the x-axis and centred about the origin, this result can be proved analytically with simple though cumbersome algebra. For the more general case of N elements, numerical studies have indicated that the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ in (6-3) has the form

$$\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}=\begin{bmatrix} c_1 & 0 & c_2 & 0 & c_3 \\ 0 & c_4 & 0 & c_5 & 0 \\ c_2 & 0 & c_6 & 0 & c_7 \\ 0 & c_5 & 0 & c_8 & 0 \\ c_3 & 0 & c_7 & 0 & c_9 \end{bmatrix} \tag{6-4}$$

In this situation, the polynomial $P(\alpha)$ in (6-3) becomes

$$P(\alpha)=c_1+(c_8-2c_3)\alpha^2+c_9\alpha^4. \tag{6-5}$$

Taking the first derivative of $P(\alpha)$ with respect to α gives the cubic polynomial

$$P'(\alpha)=2(c_8-2c_3)\alpha+4c_9\alpha^3 \tag{6-6}$$

which has the three roots

$$\alpha=0,\alpha=\pm\sqrt{\frac{c_8-2c_3}{2c_9}}. \tag{6-7}$$

Substituting each of these roots into (6-5) gives the following local minima of $P(\alpha)$

$$\alpha=0\rightarrow P_{\min}(\alpha)=c_1 \tag{6-8}$$

$$\alpha=\pm\sqrt{\frac{c_8-2c_3}{2c_9}}\rightarrow P_{\min}(\alpha)=\frac{-(c_8-2c_3)^2}{4c_9}+c_1 \tag{6-9}$$

Since the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}$ must be positive definite, the element c_9 on the main diagonal must be positive and from (6-8) and (6-9) we can see that $P(\alpha)$ will always have two equal global minimum values located at the two values of α given in (6-9).

This characteristic of the linear array is illustrated in the following numerical example. The 5-element linear array structure and source scenario is shown in Fig. 6.4.

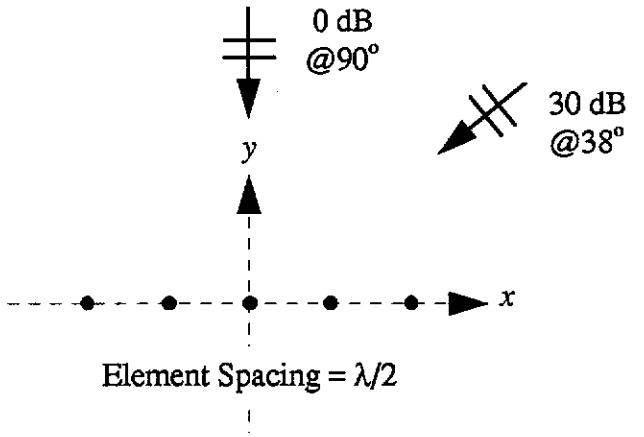


Figure 6.4 Array Structure and Source Scenario

The optimum output power of this processor as the look direction is swept from 0° to 180° is shown in Fig. 6.5.

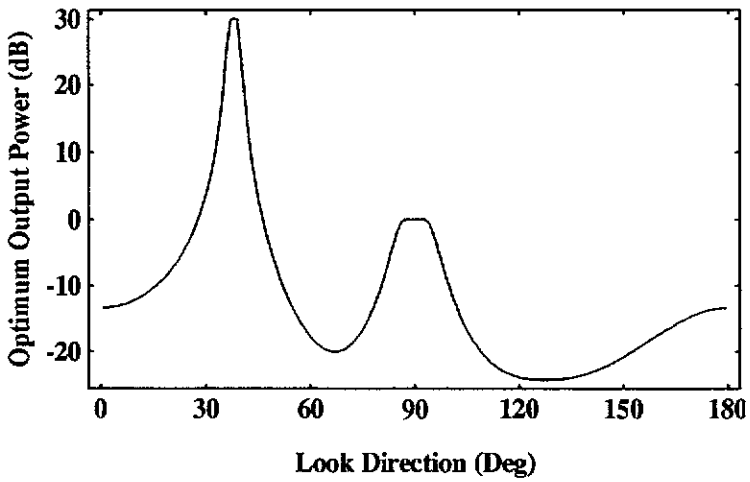


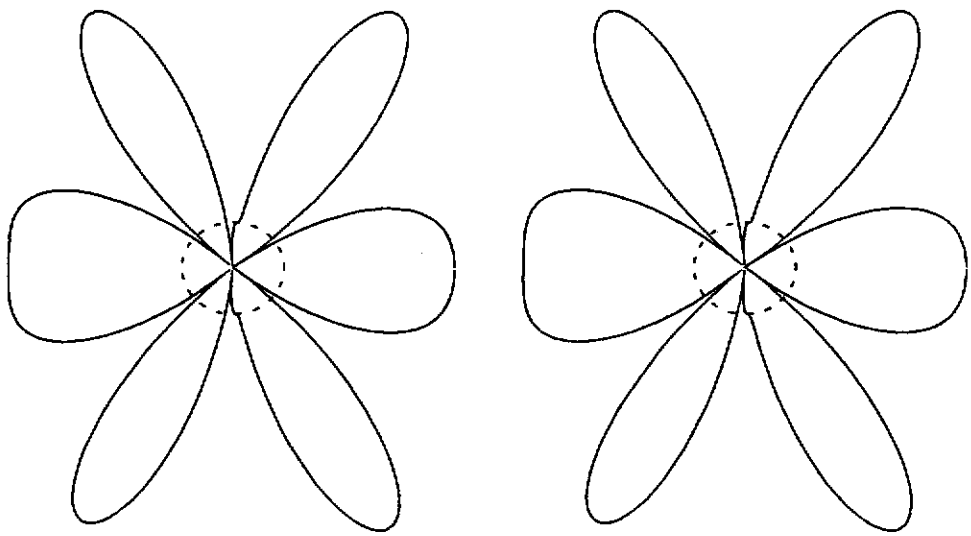
Figure 6.5 Optimum Output Power of the Array Processor in Fig. 6.4

From equation (6-9), each optimum point on the curve in Fig. 6.5 has two corresponding values of α and two corresponding weight vectors associated with it. For example take the look direction of $\theta = 86^{\circ}$. The two-stage optimisation problem has the following two solutions, both of which give the same minimum mean output power values

a)
$$\alpha_{opt1} = 9.48564, P_{opt} = -0.49 \text{ dB}$$
$$w_{opt}(\alpha_{opt1}) = \{1.27, -0.68, -0.81, -0.1, 1.1, 1.49, -1.11, -1.66, -0.56, 2.2\} \quad (6\text{-}10)$$

b)
$$\alpha_{opt2} = -9.48564, P_{opt} = -0.49 \text{ dB}$$
$$w_{opt}(\alpha_{opt2}) = \{1.1, -0.1, -0.81, -0.68, 1.27, -2.2, 0.56, 1.66, 1.11, -1.49\} \quad (6\text{-}11)$$

If we plot the polar magnitude responses for a) and b) we find that they are identical as shown in Fig. 6.6.



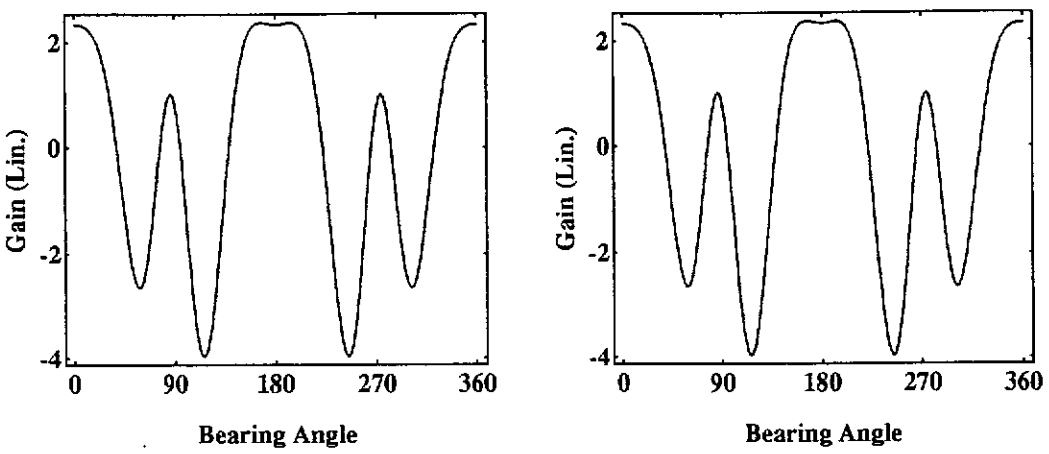
(a) (b)

Figure 6.6 Magnitude Response Plots

(a) plot obtained using the weight vector in a) above

(b) plot obtained using the weight vector in b)

However, if we plot the real and imaginary components of the response (Figs. 6.7 and 6.8) we find that the real component is not affected by the choice of optimum α while the imaginary component is inverted.

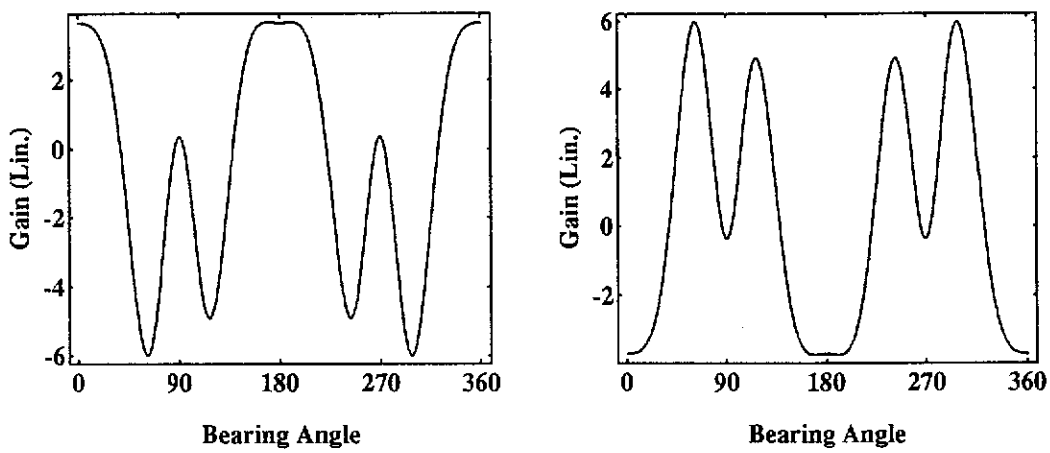


(a) (b)

Figure 6.7 Real Component of the Array Response

(a) plot obtained using the weight vector in a) above

(b) plot obtained using the weight vector in b)



(a) (b)

Figure 6.8 Imaginary Component of the Array Response

(a) plot obtained using the weight vector in a) above

(b) plot obtained using the weight vector in b)

6.3 *Effect on the Adaptive Processor*

The effect of more than one optimum solution on the optimum processor as discussed in the last section, while interesting, does not present any difficulties in implementation as the designer has the ability to decide which solution best suits the array processing application. However, this is not true in the real-time adaptive processing situation as the adaptive algorithm must decide "on-the-fly" which optimum solution to use. If the adaptive algorithm has no decision mechanism and simply chooses arbitrarily then the algorithm may *switch* repeatedly between the two optimum solutions. This effect is demonstrated in the following example.

Consider again the linear array shown in Fig. 6.4 and the new adaptive algorithm given in Table 4.1. Additional simulation parameters include:

- Array look direction $\theta_0 = 38^\circ$
- Forgetting factor, $\lambda = 0.99$
- Initialisation constant, $\delta = 100$
- Source and interference signals modelled as zero mean, white Gaussian distributed sequences.
- Element self noise = -20 dB (also modelled as zero-mean, white Gaussian distributed signal).

Fig. 6.9 shows the instantaneous output power of the array processor as it adapts to the signal environment and Fig. 6.10 shows the value of $\alpha_{opt}(n)$ as the adaptation process proceeds. In this plot no restrictions on the choice of $\alpha_{opt}(n)$ have been imposed and consequently the adaptive algorithm switched repeatedly between estimates of the two optimum α's given by α_{opt1} and α_{opt2} in (6-10) and (6-11). The optimum values are shown as horizontal lines in Fig. 6.10.

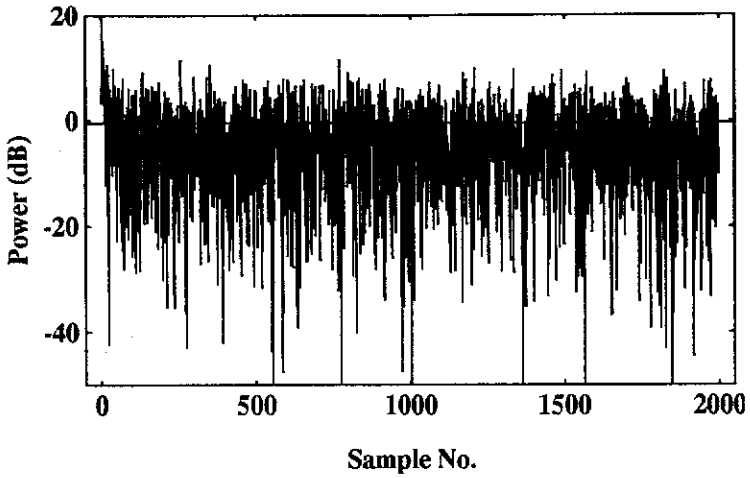


Figure 6.9 Instantaneous Array Output Power of the Linear Array Processor as it adapts to the Signal Environment

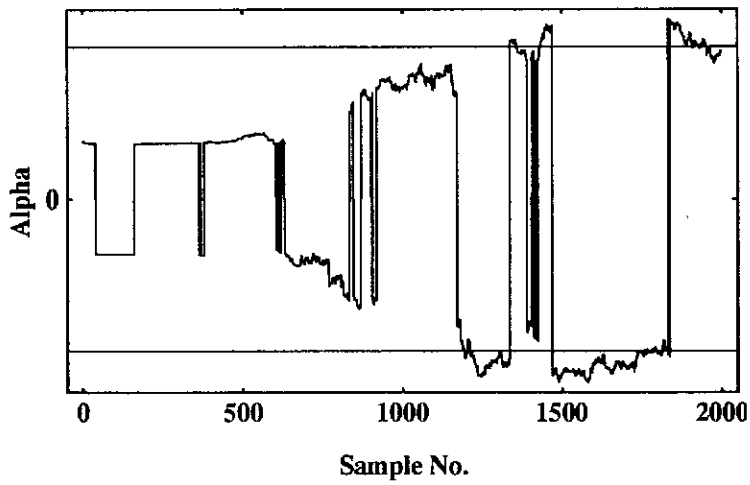


Figure 6.10 Switching in the Parameter $\alpha_{opt}(n)$ During and After Adaptation

While this parameter switching phenomenon may appear to be undesirable, surprisingly, it can have a quite beneficial effect on the processor in certain situations. Consider the scenario where a linear array processor is required to reconstruct both the in-phase and quadrature (real and imaginary) baseband components of some source signal. From the previous discussion we have seen that the adaptive derivative constrained array processor will switch repeatedly between two values of the parameter α causing the imaginary component of the array response to invert accordingly. In Chapter 2, we represented the complex response of the array for a source signal arriving from the direction (θ,ϕ) by

$$H(\theta,\phi) = s^H(\theta,\phi)w_c. \tag{6-12}$$

Let the real and imaginary parts of $H(\theta,\phi)$ be given by

$$H(\theta,\phi) = h_I + jh_Q. \tag{6-13}$$

For the adaptive linear array processor of interest here, (6-13) becomes

$$H(\theta,\phi,n) = h_I(n) + j\,\mathrm{sgn}(\alpha(n))h_Q(n) \tag{6-14}$$

where $H(\theta,\phi,n)$ is the complex gain at the nth iteration and $\mathrm{sgn}(\alpha(n))$ is either positive or negative depending on the value chosen for $\alpha_{opt}(n)$ at the nth iteration. For a signal source that is exactly matched to the array look direction (θ_0,ϕ_0), (6-14) becomes

$$H(\theta_0,\phi_0,n) = 1, \tag{6-15}$$

since the look direction constraint of (2-58) must hold. However, if there is some mismatch between the source signal and the array look direction, the gain of the array for the source direction will not be exactly $(1+j0)$. Let us represent the baseband message signal of the source at time sample n by

$$s(\theta,\phi,n) = s_I(n) + js_Q(n) \tag{6-16}$$

where $s_I(n)$ and $s_Q(n)$ are independent, in-phase and quadrature components of the source message at time n. From (6-14), the complex output of the array processor due to this source is given by

$$\begin{aligned} y_c &= H^*(\theta,\phi,n)s(\theta,\phi,n) \\ &= \big(h_I(n) - j\,\mathrm{sgn}(\alpha(n))h_Q(n)\big)\big(s_I(n) + js_Q(n)\big). \end{aligned} \tag{6-17}$$

The in-phase and quadrature components of the output are given therefore by

$$\begin{aligned} y_I(n) &= h_I(n)s_I(n) + \mathrm{sgn}(\alpha(n))h_Q(n)s_Q(n) \\ &\qquad \text{and} \\ y_Q(n) &= h_I(n)s_Q(n) - \mathrm{sgn}(\alpha(n))h_Q(n)s_I(n). \end{aligned} \tag{6-18}$$

From (6-18) we see that some *cross-channel interference* between the in-phase and quadrature signals has occurred where components of the quadrature and in-phase source signals appear in the unintended array output channels. This effect can be seen

more clearly by looking at the cross-correlation between the source message signals and the in-phase and quadrature array outputs as follows. Assuming $h_I(n)$ and $h_Q(n)$ are independent of $s_I(n)$ and $s_Q(n)$ then

$$\begin{aligned} \mathrm{E}\left[y_I(n)s_Q(n)\right] &= \mathrm{E}\left[h_I(n)s_I(n)s_Q(n)\right] + \mathrm{E}\left[h_Q(n)s_Q(n)s_Q(n)\,\mathrm{sgn}(\alpha(n))\right] \\ &= \mathrm{E}\left[h_Q(n)\right]\cdot\mathrm{E}\left[s_Q^2(n)\right]\cdot\mathrm{E}\left[\mathrm{sgn}(\alpha(n))\right] \end{aligned}, \quad (6\text{-}19)$$

and

$$\begin{aligned} \mathrm{E}\left[y_Q(n)s_I(n)\right] &= \mathrm{E}\left[h_I(n)s_Q(n)s_I(n)\right] + \mathrm{E}\left[h_Q(n)s_I(n)s_I(n)\,\mathrm{sgn}(\alpha(n))\right] \\ &= \mathrm{E}\left[h_Q(n)\right]\cdot\mathrm{E}\left[s_I^2(n)\right]\cdot\mathrm{E}\left[\mathrm{sgn}(\alpha(n))\right] \end{aligned}. \quad (6\text{-}20)$$

It is apparent from (6-19) and (6-20) that the amount of cross-channel interference between in-phase and quadrature channels can be reduced by making the quantity $\mathrm{sgn}(\alpha(n))$ random, i.e., by allowing the parameter α to switch randomly. In this situation, then, the switching of α has a *randomising* effect on the cross-channel interference. This effect is demonstrated in the following example.

Consider the simulation scenario used in the previous computer studies (Figs. 6.4 to 6.10). Here, a 0 dB source is located at 90° while the array look direction is 86°. This represents a mismatch between the source and array look direction of 4°. From the optimum processor we find that real component of the gain response in the direction of the source, $\theta = 90^\circ$, is

$$h_I\left(90^\circ\right) = 0.78, \quad (6\text{-}21)$$

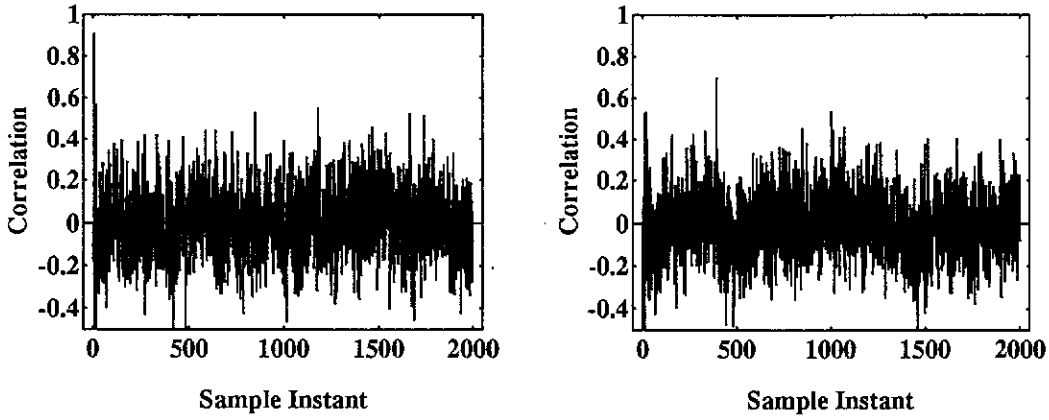
while the imaginary component is

$$h_Q\left(90^\circ\right) = \pm 0.36, \quad (6\text{-}22)$$

where the sign depends on which optimum α is chosen at a particular sample instant.

The following plots show the cross-correlation between the in-phase component of the source message and the quadrature component of the array output and between the quadrature component of the source message and the in-phase component of the array output respectively. In the first two plots α is permitted to switch freely between the optimum values. In the next two plots, however, the switching in α is

reduced by choosing the positive optimum value wherever possible. All of the plots were obtained by ensemble averaging 50 independent runs and identical data sets were used for the two simulations.

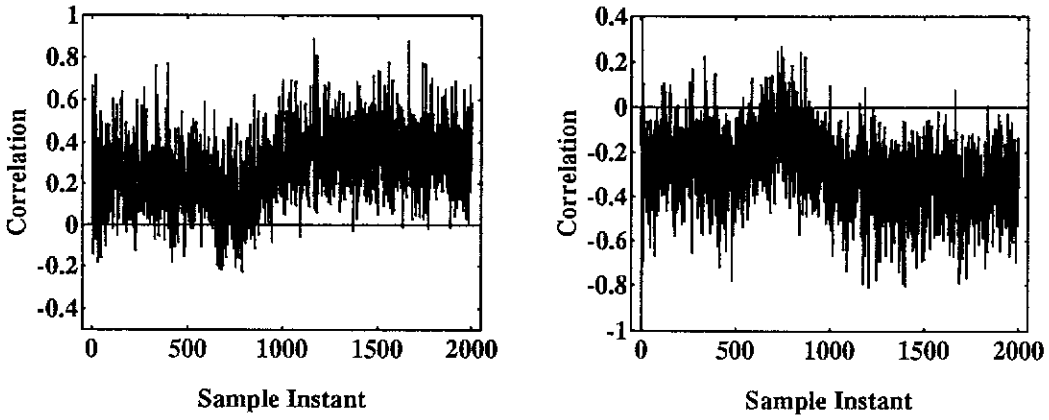


(a) (b)

Figure 6.11 Cross-correlation: α Permitted to Switch Freely

(a) Ensemble Averaged Cross-correlation between the real output and the imaginary input

(b) Ensemble Averaged Cross-correlation between the imaginary output and the real input



(a) (b)

Figure 6.12 Cross-correlation: Positive Value ofα Chosen Whenever Possible

(a) Ensemble Averaged Cross-correlation between the real output and the imaginary input

(b) Ensemble Averaged Cross-correlation between the imaginary output and the real input

In Fig. 6.11 where α has been allowed to switch freely, we see that there is little or no cross-channel interference between the in-phase and quadrature channels. In fact, the average value of the correlation after convergence (about 1000 samples) is 0.006 for Fig. 6.11(a) and -0.012 for Fig 6.11(b). In Fig. 6.12, however, where the switching of α has been restricted, we see that the cross-channel interference is significantly higher with average correlation values after convergence of 0.34 for Fig. 6.12(a) and -0.35 for Fig 6.12(b). Note how close these values are to the predicted values of ± 0.36 given in (6-22).

It is important to note that for a single run, $\alpha_{opt}(n)$ approximates one of the optimum values for extended periods. During these periods, one expects the cross-channel interference to increase as the randomising effect of the switching α, as discussed above, is not present. This suggests that $\alpha_{opt}(n)$ is not an ergodic process and further investigation of the switching α phenomenon must be undertaken.

7. Conclusions and Extensions

7.1 Summary and Conclusion

In this thesis a new adaptive recursive least squares narrowband antenna array processor with look direction plus first and second order NS spatial derivative constraints has been presented. We have investigated the performance of the new algorithm via computer studies and have identified factors critical to the numerical stability of the algorithm in a practical, finite precision environment. In addition we have proposed the inclusion of existing techniques to significantly reduce the possibility of numerical instability. Finally we have investigated issues relating to the multimodal nature of the optimisation problem and their effects on both the optimum and adaptive processors. In summary, the main contributions of this thesis are as follows:

1. A new adaptive RLS narrowband antenna array processor with second order NS derivative constraints has been presented. Second order NS derivative constraints have been shown to provide the optimum array processor with robustness against directional mismatch. The new algorithm allows the complex nonlinear optimisation problem associated with second order NS spatial derivative constraints to be solved efficiently in an adaptive fashion using sampled input data. Since the algorithm is essentially a variation of the RLS algorithm it offers the performance features of LS techniques, namely, fast convergence and high accuracy.

   The algorithm uses a two-stage approach to solve the optimisation problem and in Chapter 5 we have shown how global polynomial factorisation techniques can be exploited to significantly improve the computational efficiency of the second stage which essentially involves finding the roots of a time-varying polynomial.

Although this thesis deals only with a narrowband planar array operating in 2D space, the technique developed herein can be easily extended to 3D space operation as well as broadband antenna array processor applications. In addition, the algorithm can be applied to the more general case of adaptive RLS filtering with linear and quadratic equality constraints.

2. A new parameter, which is central to the round-off error propagation mechanism within the algorithm has been identified and which can be used to detect the onset of numerical instability within the algorithm. In addition, the practical implementation of the algorithm in a finite precision environment has been considered and in this regard two key aspects in the numerical robustness of the algorithm were investigated: (i) drift from the constraint space; and (ii) explosive divergence in the inverse correlation matrix update. Existing techniques were found to counteract the effects of these difficulties and computer studies demonstrated the effectiveness of these techniques.

3. In Chapter 6 the multimodal nature of the optimisation problem was studied and the effect of more than one optimum solution on both optimum and adaptive processors was investigated through computer studies.

   In particular it was shown that for a linear array with equally spaced elements aligned along the x-axis with phase centre at the origin and operating in 2D space, the optimum processor will always have two optimum solutions, each of which can be computed from a simple expression using the elements of a matrix which is updated by the new algorithm. This eliminates the need for the second stage of the two-stage minimisation, thus not only reducing the computational complexity of the whole adaptive algorithm but avoiding any potential problems associated with the parameter switching.

7.2 Suggestions for Further Research

There are several interesting extensions arising from the work carried out in this thesis.

1. Although the techniques suggested herein to stabilise the algorithm in a finite precision environment have been shown via computer studies to eliminate the problem of drift from the constraint space and significantly reduce the chance of numerical instability, a full analysis of the complex round-off-error propagation mechanism within the $Q(n)$ matrix recursion equations still remains to be done. This may highlight other techniques by which the numerical robustness of the algorithm can be further improved.

2. This thesis deals only with narrowband antenna array processors. An obvious extension, in this regard, is in the application of the new algorithm to broadband antenna array processing. This is a challenging problem since to investigate the adaptive processor by computer simulation, "snapshots" or instantaneous samples of the broadband array input vector need to be generated.

3. In Chapter 6 we briefly investigated the effects of the multimodal optimisation problem on both the optimum and adaptive array processors. While the work has highlighted some important aspects of the multimodal problem there still remain some interesting issues relating to the choice of the parameter α both in the optimum and adaptive problems. Although it was shown that the choice of this parameter affected the beam pattern of the array processor, it is still unclear what other characteristics of the processor the parameter influences. This has implications in both optimum and adaptive array processing, for in the optimum case the designer must somehow choose between the optimum solutions and in the adaptive case the effect of parameter switching on the array processor performance must be determined.

Appendix A

In this appendix, we present a brief derivation of Frosts Linearly Constrained LMS Algorithm.

A.1 The Optimum Problem

Consider firstly the linearly constrained quadratic cost minimisation problem

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimise}} \quad & P(\boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{R} \boldsymbol{w} \\ \text{subject to} \quad & \boldsymbol{C}^T \boldsymbol{w} = \boldsymbol{h} \end{aligned} \tag{A-1}$$

where $\boldsymbol{w} \in \mathcal{R}^L$ is the vector of L filter weights, $\boldsymbol{R} \in \mathcal{R}^{L \times L}$ is the input signal correlation matrix, $\boldsymbol{C} \in \mathcal{R}^{L \times M}$ is the constraint matrix and $\boldsymbol{h} \in \mathcal{R}^M$ is the constraint vector. Note that we have M linear constraints.

The optimum weight vector $\boldsymbol{w}_{opt}$ can be found for (A-1) by the method of Lagrange multipliers [1] as follows:

Define the Lagrangian formed by adjoining the constraint equation with the cost function as follows

$$J(\boldsymbol{w}) = \frac{1}{2} \boldsymbol{w}^T \boldsymbol{R} \boldsymbol{w} + \boldsymbol{\lambda}^T \left(\boldsymbol{C}^T \boldsymbol{w} - \boldsymbol{h} \right) \tag{A-2}$$

where $\boldsymbol{\lambda} \in \mathcal{R}^M$ is a vector of undetermined parameters termed the Lagrange multipliers. The minimisation is achieved by taking the gradient of $J(\boldsymbol{w})$ with respect to $\boldsymbol{w}$ and equating it to zero, i.e.,

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \boldsymbol{R} \boldsymbol{w} + \boldsymbol{C} \boldsymbol{\lambda} = 0. \tag{A-3}$$

Solving for $\boldsymbol{w}$ in (A-3), the optimum weight vector is given by

$$\boldsymbol{w}_{opt} = -\boldsymbol{R}^{-1} \boldsymbol{C} \boldsymbol{\lambda} \tag{A-4}$$

where the vector $\boldsymbol{\lambda}$ is yet to be determined. It is assumed that the correlation matrix, $\boldsymbol{R}$, is non-singular. Since the optimum weight vector must satisfy the constraints in

(A-1), substituting w_{opt} from (A-4) into the constraint equation gives

$$-\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\lambda = \boldsymbol{h} \tag{A-5}$$

or, re-arranging

$$\lambda = -\left(\boldsymbol{C}^T\boldsymbol{R}\boldsymbol{C}\right)^{-1}\boldsymbol{h}. \tag{A-6}$$

Substituting for λ in (A-4) gives the required expression for the optimum weight vector

$$\boldsymbol{w}_{opt} = \boldsymbol{R}^{-1}\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{h}. \tag{A-7}$$

A.2 Frost's Adaptive Algorithm

In the adaptive problem it is assumed that the correlation matrix $\boldsymbol{R}$ is unknown *a priori* and must be learned by the adaptive technique. To begin the derivation, however, we will temporarily assume that the input correlation matrix is known.

In constrained gradient descent techniques, the weight vector is initialised such that it satisfies the constraints. A suitable choice is

$$\boldsymbol{w}(0) = \boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{C}\right)^{-1}\boldsymbol{h}. \tag{A-8}$$

The weight vector is then stepped in the negative direction of the constrained gradient given by (A-3). The length of the step is proportional to the magnitude of the constrained gradient and is scaled by a constant μ. The gradient based update for the weight vector is given by

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\nabla_{\boldsymbol{w}}J(\boldsymbol{w}(n)) \tag{A-9}$$

or, from (A-3)

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\left[\boldsymbol{R}\boldsymbol{w}(n) + \boldsymbol{C}\lambda(n)\right] \tag{A-10}$$

The Lagrange multipliers, $\lambda(n)$, are chosen by requiring that $\boldsymbol{w}(n+1)$ satisfies the constraint equation in (A-1), so that

$$\boldsymbol{h} = \boldsymbol{C}^T\boldsymbol{w}(n+1) = \boldsymbol{C}^T\boldsymbol{w}(n) - \mu\boldsymbol{C}^T\boldsymbol{R}\boldsymbol{w}(n) - \mu\boldsymbol{C}^T\boldsymbol{C}\lambda(n) \tag{A-11}$$

Solving for the Lagrange multipliers and substituting into the weight update (A-10)

yields

$$w(n+1) = w(n) - \mu\left[I - C\left(C^T C\right)^{-1} C\right] R w(n) + C\left(C^T C\right)^{-1}\left[h - C^T w(n)\right] \tag{A-12}$$

Note that the term $h - C^T w(n)$ is not assumed to be zero. Defining the $M \times M$ matrix

$$P = I - C\left(C^{-1} C\right) C^T \tag{A-13}$$

the weight vector update may be written as

$$w(n+1) = P\left[w(n) - \mu R w(n)\right] + w(0) \tag{A-14}$$

In practice, the true autocorrelation matrix is unknown and is replaced with the instantaneous estimate $\hat{R} = x(n)x^T(n)$ where $x(n) \in \mathcal{R}^L$ is the input data vector. This yields the Frost stochastic, linearly constrained LMS algorithm [2]

$$w(0) = C\left(C^T C\right)^{-1} h \tag{A-15}$$

$$P = I - C\left(C^{-1} C\right) C^T \tag{A-16}$$

$$w(n+1) = P\left[w(n) - \mu y(n) x(n)\right] + w(0) \tag{A-17}$$

A.3 References

[1] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1991.

[2] O. L. Frost, III, "An Algorithm for Linearly Constrained Adaptive Array Processing", *Proc. IEEE*, vol. 60, no. 8, pp. 926-935, August 1972.

Appendix B

In this appendix, we examine the primary mechanism which brings about explosive divergence and consider an approach for stabilising the conventional RLS algorithm.

B.1 The Approach of Bottomley and Alexander

Explosive divergence is a problem of numerical instability which is well-documented for the standard RLS algorithm, [1, 2, 3, 4]. It occurs when the estimated inverse correlation matrix loses its property of positive definiteness due to finite precision effects in the time update equations.

To help understand the mechanism behind explosive divergence we need to look at the conventional RLS, or CRLS, algorithm and the one-dimensional, unconstrained adaptive filter shown in Fig B.1.

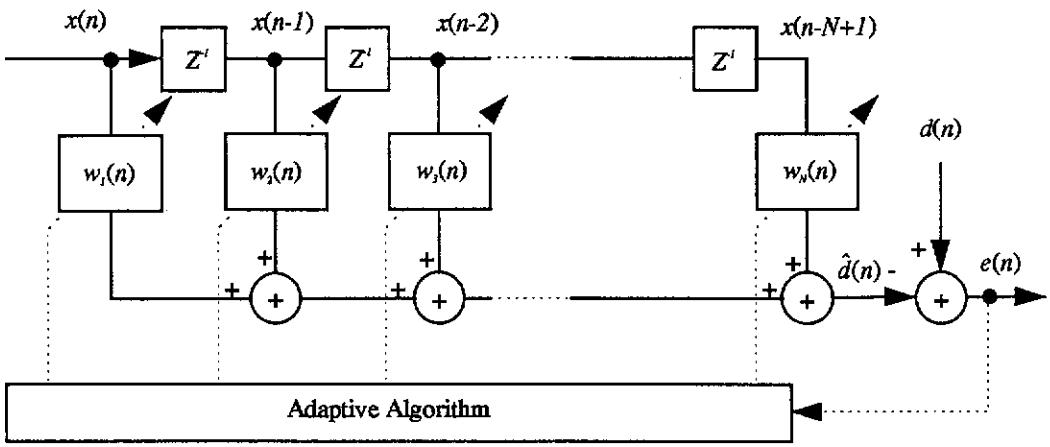


Figure B.1 Adaptive FIR filter of Length N with a Desired Response $d(n)$.

In [4] a form of the CRLS filter which algorithmically preserves the symmetry of the inverse correlation matrix was developed. (A part of this algorithm has already been used in Table 4.2.) The full version of this algorithm is given below and relates to the filter and notation of Fig. B.1.

Initialise:

$$\boldsymbol{w}(0) = \boldsymbol{x}(0) = \boldsymbol{0}_{2L}$$
$$\boldsymbol{R}^{-1}(0) = \delta \boldsymbol{I}_{2L} \tag{B-1}$$

For each new input sample, $x(n)$, at time n:

$$e(n|n-1) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n-1) \tag{B-2}$$

$$\boldsymbol{p}(n) = \boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n) \tag{B-3}$$

$$\mu(n) = \boldsymbol{x}^T(n)\boldsymbol{p}(n) \tag{B-4}$$

$$\beta(n) = 1/(\lambda + \mu(n)) \tag{B-5}$$

$$\boldsymbol{g}(n) = \beta(n)\boldsymbol{p}(n) \tag{B-6}$$

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{g}(n)e(n|n-1) \tag{B-7}$$

$$\boldsymbol{R}^{-1}(n) = \frac{1}{\lambda}\left[\boldsymbol{R}^{-1}(n-1) - \beta(n)\left(\boldsymbol{p}(n)\boldsymbol{p}^T(n)\right)\right] \tag{B-8}$$

Table B.1 Full Symmetry Preserving RLS Algorithm.

In Equation (B-2), the vector $\boldsymbol{x}(n) \in \mathcal{R}^N$ is composed of the N most recently available samples of the input process, i.e.,

$$\boldsymbol{x}(n) = \left[x(n), x(n-1), \ldots, x(n-N+1)\right]^T \tag{B-9}$$

and $e(n|n-1) \in \mathcal{R}$ is known as the *a priori estimation error* since it uses the current values of the input vector and desired response but the previous weight vector.

In [1] and [2], the explosive divergence scenario in the CRLS algorithm is described as being characterised by two simultaneous events: 1) an intermediate quantity, known as the *angle parameter*, given by $\beta(n)$ in (B-5), becomes negative; and 2) the weight error becomes suddenly large. In [1], Bottomley and Alexander showed that these events are brought about by a complex error propagation mechanism which tends to bias the inverse correlation matrix, $\boldsymbol{R}^{-1}(n)$, towards being negative definite. They show that by applying certain rounding rules when performing the calculations in Table B.1, the inherent bias toward negative definiteness in $\boldsymbol{R}^{-1}(n)$ can be counteracted.

The rounding rules are summarised in Table 4.3.

B.2 Computer Studies

In the following numerical example the explosive divergence scenario is simulated and the effect of the rounding rules is demonstrated. The simulation assumes a system identification application, as shown in Fig. B.2, using a 10 tap adaptive FIR filter, similar to that shown in Fig. B.1.

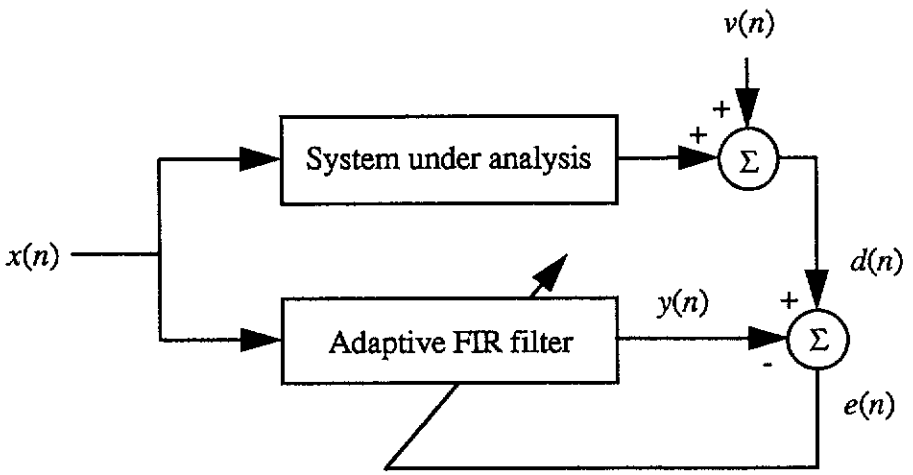


Figure B.2 System Identification Example.

The input to the system, $x(n)$, is modelled as a zero mean, white, Gaussian distributed sequence with a variance of 0 dB. The measurement noise, $v(n)$, is also a zero mean, white, Gaussian distributed sequence but with a variance of -10 dB. The system under analysis has an impulse response given by

$$H(z) = 0.35 + 0.17z^{-1} + 0.36z^{-2} + 0.34z^{-3} + 0.1z^{-4} + 0.38z^{-5} \\ +0.37z^{-6} + 0.44z^{-7} + 0.15z^{-8} + 0.31z^{-9} \qquad \text{(B-10)}$$

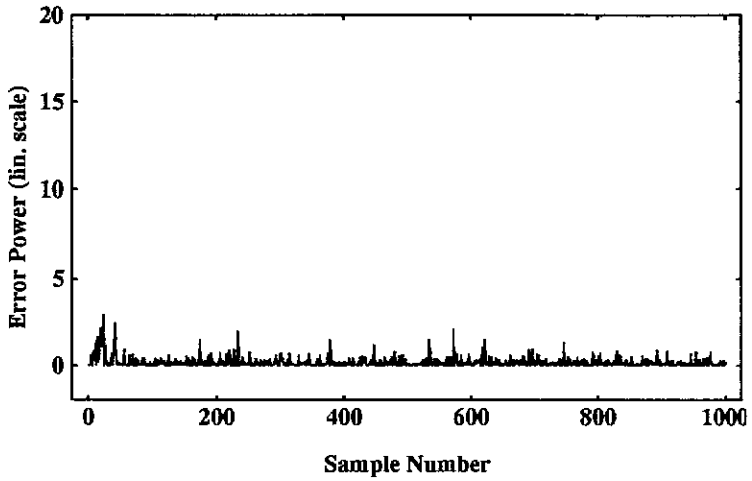
The algorithm in Table B.1 was used in the adaptive filter to update the tap weights so that the error, $e(n)$, would be minimised in the least squares sense.

The following algorithm parameters were used:

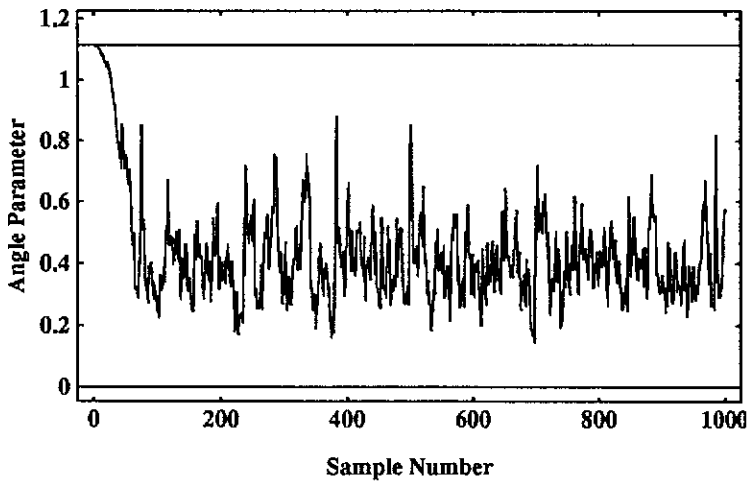
- $\delta = 0.001$ in (B-1) and
- forgetting factor, $\lambda = 0.9$ in (B-1) and (B-8).

Figs. B.3 to B.5 each show plots of three filter parameters as the filter adapts to the signal environment. In Fig. B.3, the calculations in Table B.1 were limited only by the machine precision of the computer used to simulate the filter, i.e., 16 decimal places of floating point precision. In Fig. B.4, the precision of the calculations was limited to only 2 decimal digits of floating point precision and conventional rounding was applied. In Fig. B.5 the floating point precision was also 2 decimal places, however, the calculations in the RLS algorithm were performed according to the rounding rules in Table 4.3. The input data sequence in all three simulations was identical.

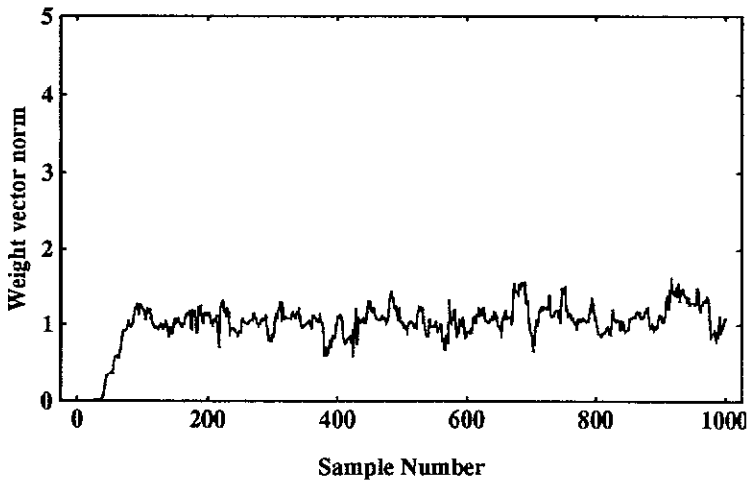
The three plots in each of the figures are, respectively, the instantaneous error power, $e^2(n|n-1)$ (linear scale), the angle parameter $\beta(n)$, and the weight vector norm $w^T(n)w(n)$ (linear scale), each plotted against the sample (or iteration) number.



(a)

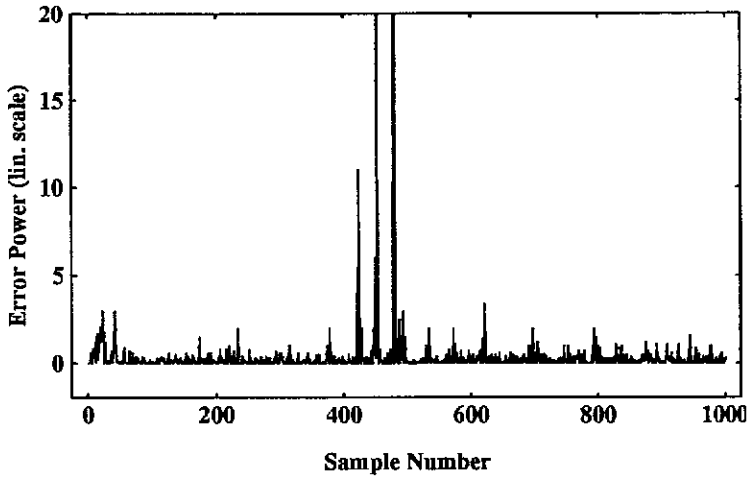


(b)

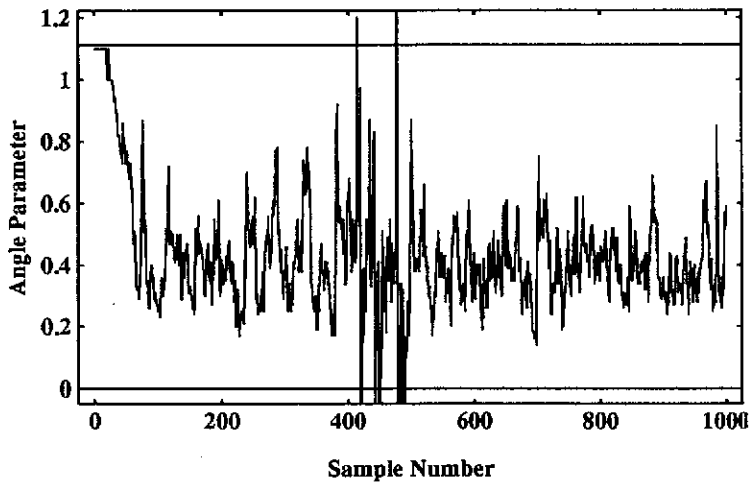


(c)

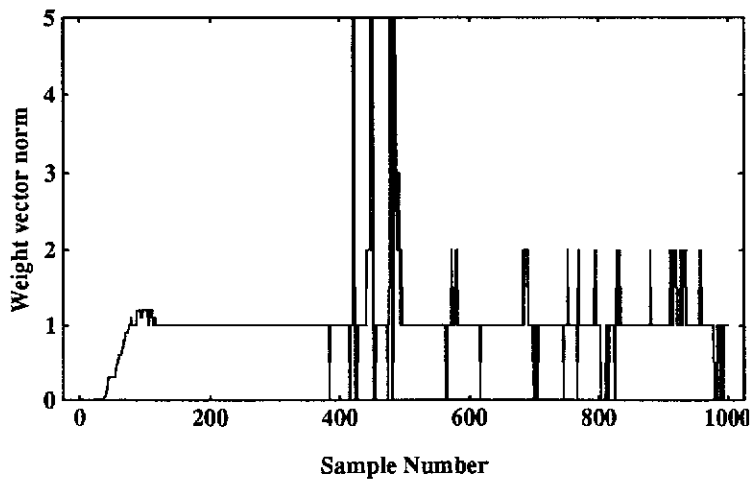
Figure B.3 Computational Precision = 16 decimal places: (a) Instantaneous Error Power, (b) Angle Parameter, and (c) Weight Vector Norm.



(a)

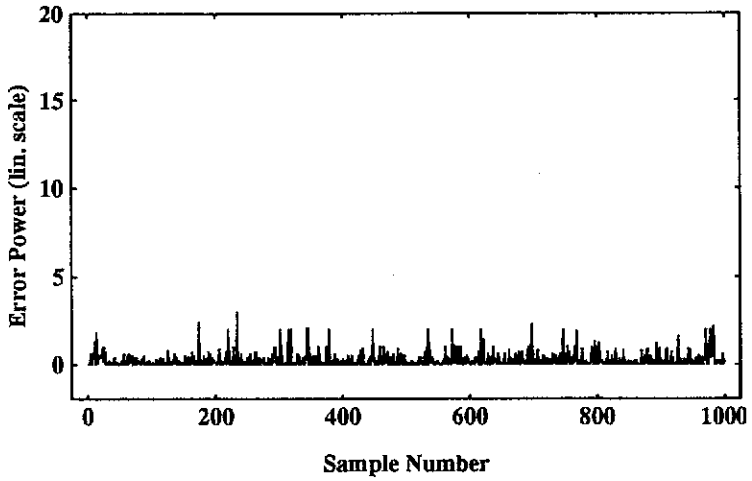


(b)

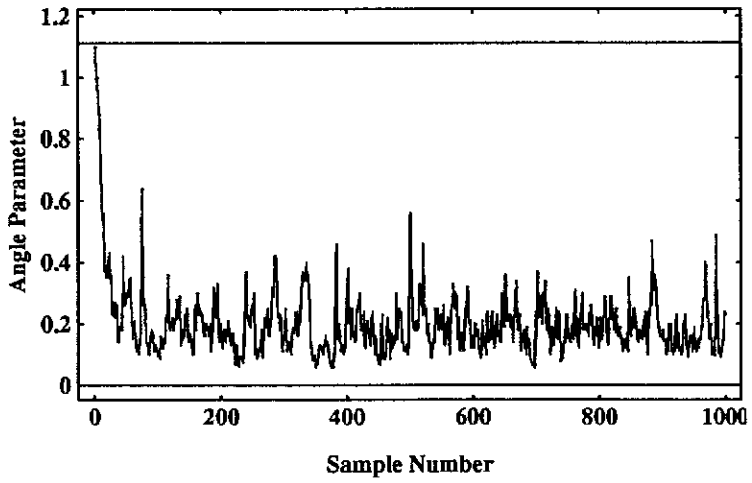


(c)

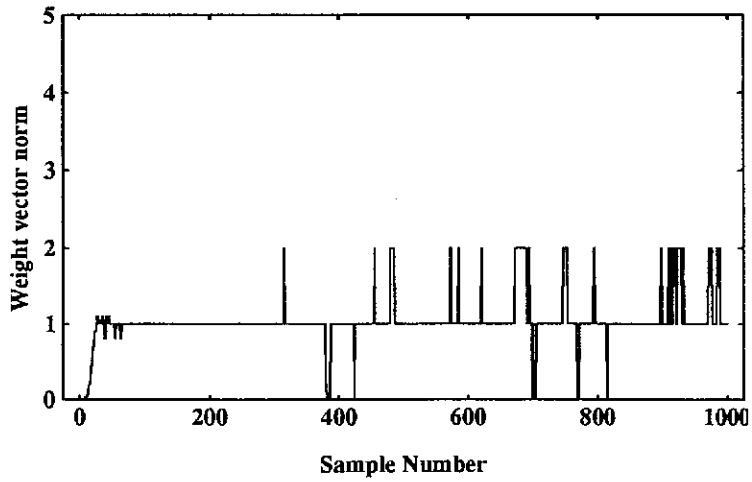
Figure B.4 Computational Precision = 2 decimal places, Conventional Rounding: (a) Instantaneous Error Power, (b) Angle Parameter, and (c) Weight Vector Norm.



(a)

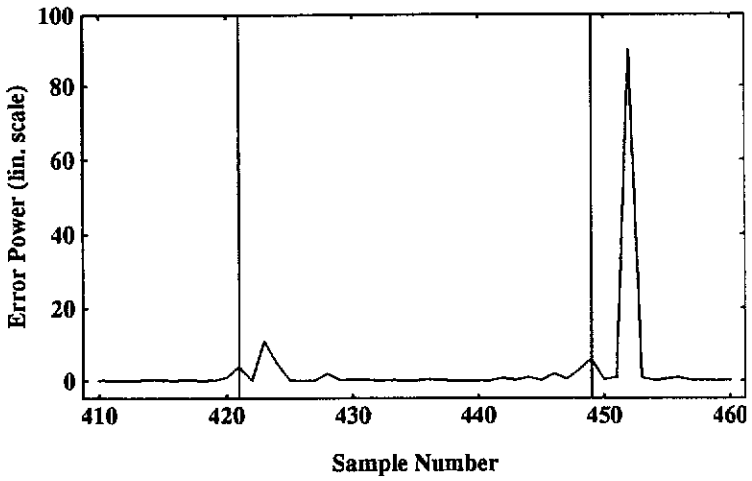


(b)

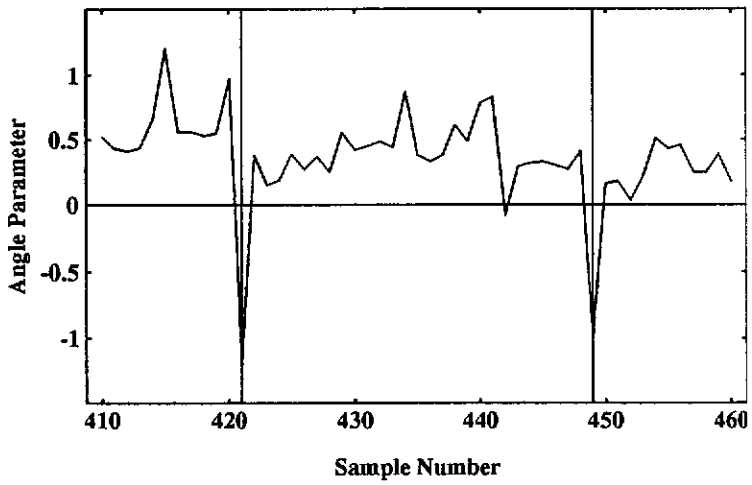


(c)

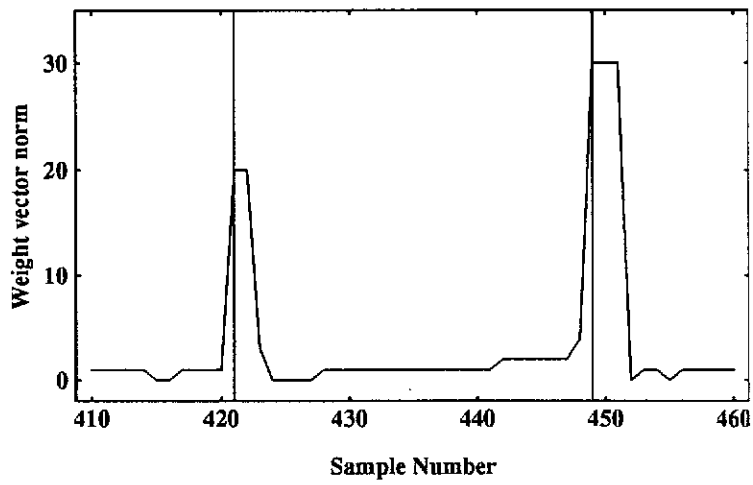
Figure B.5 Computational Precision = 2 decimal places, Rounding Rules Applied: (a) Instantaneous Error Power, (b) Angle Parameter, and (c) Weight Vector Norm.



(a)



(b)



(c)

Figure B.6 Expanded Sections of Figure B.4: (a) Instantaneous Error Power, (b) Angle Parameter, and (c) Weight Vector Norm.

Note that in Fig. B.4, where the precision has been reduced to two decimal places, the algorithm has become unstable. Between the 400^{th} and 500^{th} samples, the instantaneous error power suddenly increases in a series of spikes. This region of instability can be seen more clearly in Fig. B.6 where sections from the plots in Fig. B.4 have been expanded. The angle parameter suddenly becomes negative (indicating that the inverse correlation matrix has become negative definite) and at the same time the weight vector norm becomes suddenly large. Shortly afterward, the instantaneous error also becomes large.

In the simulation shown in Fig. B.5, the rounding rules have prevented the instability from occurring by biasing the inverse correlation matrix towards being positive definite.

It is accepted that the severe numerical precision environment of the second and third simulations, i.e., two decimal places of floating point precision, is unlikely in a practical application, however it was chosen only to illustrate the explosive divergence phenomenon in a low number of algorithm iterations. In a higher precision environment, the onset of explosive divergence may simply be delayed if measures are not taken to stabilise the algorithm.

B.3 References

[1] G E Bottomley, S T Alexander, "A Novel Approach for Stabilising Recursive Least Squares Filters", *IEEE Trans. Signal Processing*, vol. 39, no. 8, pp. 1770-1779, August 1991.

[2] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1991.

[3] M. H. Verhaegen, "Round-off Error Propagation in Four Generally Applicable, Recursive, Least-squares Estimation Schemes", *Automatica*, vol. 25, no. 3, pp. 437-444, 1989.

[4] G. E. Bottomley and S. T. Alexander, "A Theoretical Basis for the Divergence of Conventional Recursive Least Squares Filters", *Proc. 1989 IEEE ICASSP* (Glasgow, Scotland), pp. 908-911, May 1989.

Appendix C

In this appendix, we derive upper and lower bounds on the second angle parameter given in Equation (4-32) in Section 4.2.1.

The second angle parameter $\gamma(n)$, identified in Section 4.2.1 as being a critical factor in the stability of the new algorithm, is given by equation (4-32) as

$$\gamma(n) = 1 - \boldsymbol{v}^T(n)\boldsymbol{u}(n), \tag{C-1}$$

where $\boldsymbol{v}(n)$ and $\boldsymbol{u}(n)$ are M-dimensional vectors given by

$$\boldsymbol{u}(n) = \boldsymbol{C}^T\boldsymbol{g}(n), \tag{C-2}$$

and

$$\boldsymbol{v}^T(n) = \boldsymbol{x}^T(n)\boldsymbol{Q}(n-1). \tag{C-3}$$

The $2L$-dimensional gain vector, $\boldsymbol{g}(n)$, is given by

$$\boldsymbol{g}(n) = \frac{\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}{\lambda + \boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}, \tag{C-4}$$

and the $2L \times M$ matrix, $\boldsymbol{Q}(n-1)$, is given by

$$\boldsymbol{Q}(n-1) = \boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}. \tag{C-5}$$

Expanding (C-1) using (C-2) to (C-5), yields

$$1 - \boldsymbol{v}^T(n)\boldsymbol{u}(n) = 1 - \frac{\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}{\lambda + \boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}. \tag{C-6}$$

Let the quotient on the right hand side of (C-6) be represented by the scalar $J(n)$, i.e.,

$$J(n) = \frac{\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}{\lambda + \boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}. \tag{C-7}$$

Since the correlation matrix $\boldsymbol{R}(n-1)$ is positive definite, its inverse, $\boldsymbol{R}^{-1}(n-1)$, is also positive definite [1].

Pre- and post-multiplying $\boldsymbol{R}^{-1}(n-1)$ by the constraint matrix $\boldsymbol{C}$ we get

$$\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}. \tag{C-8}$$

Pre- and post-multiplying again by any M-dimensional real vector gives

$$\boldsymbol{a}^T\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\boldsymbol{a} \tag{C-9}$$

and we see that the matrix $\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}$ is positive definite since (C-9) can be written in the quadratic form

$$\boldsymbol{a}^T\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\boldsymbol{a} = \boldsymbol{m}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{m} \tag{C-10}$$

where m is any $2L$-dimensional vector. With $\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}$ being positive definite, so $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}$ is also positive definite.

Now, the numerator in (C-7) can be written in quadratic form as follows

$$\begin{aligned}\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)\\ = \boldsymbol{l}^T\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{l}\end{aligned}, \tag{C-11}$$

where

$$\boldsymbol{l}(n) = \boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n). \tag{C-12}$$

Since the matrix $\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}$ is positive definite, the lower bound on the numerator in (C-7) is zero, i.e.,

$$\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n) \geq 0. \tag{C-13}$$

Similarly, we see that the lower bound on the expression $\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)$ in the denominator of (C-7) is also zero, i.e.,

$$\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n) \geq 0. \tag{C-14}$$

From (C-7), (C-13) and (C-14) it is clear that for $\boldsymbol{x}(n) = \boldsymbol{0}$

$$J(n) = \frac{0}{\lambda + 0} = 0, \tag{C-15}$$

and the lower bound on $J(n)$ is, therefore, zero.

The upper bound on $J(n)$, however, is not as straight forward. For large $x(n)$, the forgetting factor λ becomes negligible and we can write

$$J(n) \cong \frac{\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{x}(n)}. \tag{C-16}$$

Since $\boldsymbol{R}^{-1}(n-1)$ is positive definite, it can factorised using, for example the Cholesky decomposition technique, as

$$\boldsymbol{R}^{-1}(n-1) = \left(\boldsymbol{A}^{-1}\right)^T\left(\boldsymbol{A}^{-1}\right). \tag{C-17}$$

Let

$$\boldsymbol{y} = \boldsymbol{A}^{-1}\boldsymbol{x}(n) \tag{C-18}$$

such that

$$\boldsymbol{x}(n) = \boldsymbol{A}\boldsymbol{y}. \tag{C-19}$$

The expression for $J(n)$ in (C-16) can now be re-written as the generalised Rayleigh quotient [1]

$$J(n) = \frac{\boldsymbol{y}^T\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} \tag{C-20}$$

where the matrix $\boldsymbol{\Phi}$ is given by

$$\boldsymbol{\Phi} = \boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1). \tag{C-21}$$

From matrix theory, the value of the Rayleigh quotient $J(n)$ is bounded by the maximum and minimum eigenvalues of the matrix $\left(\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A}\right)$ [1], i.e.,

$$\lambda_{\min}\left(\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A}\right) \le J(n) \le \lambda_{\max}\left(\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A}\right). \tag{C-22}$$

Now, the matrix $\left(\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A}\right)$ can be re-written as

$$
\begin{aligned}
\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A} &= \boldsymbol{A}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{A} \\
&= \boldsymbol{A}^T\left(\boldsymbol{A}^{-1}\right)^T\left(\boldsymbol{A}^{-1}\right)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\left(\boldsymbol{A}^{-1}\right)^T\left(\boldsymbol{A}^{-1}\right)\boldsymbol{A} \\
&= \left(\boldsymbol{A}^{-1}\right)\boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}(n-1)\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\left(\boldsymbol{A}^{-1}\right)^T \\
&= \left(\boldsymbol{A}^{-1}\right)\boldsymbol{C}\left[\boldsymbol{C}^T\left(\boldsymbol{A}^{-1}\right)^T\cdot\left(\boldsymbol{A}^{-1}\right)\boldsymbol{C}\right]\boldsymbol{C}^T\left(\boldsymbol{A}^{-1}\right)^T \\
&= \boldsymbol{M}\left[\boldsymbol{M}^T\boldsymbol{M}\right]^{-1}\boldsymbol{M}^T.
\end{aligned}
\tag{C-23}
$$

It is a trivial matter to show that $\left(\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A}\right)$ is idempotent, i.e., $\left(\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A}\right)^2 = \left(\boldsymbol{A}^T\boldsymbol{\Phi}\boldsymbol{A}\right)$. As the eigenvalues of an idempotent matrix are either 1 or 0 [1], the maximum and minimum eigenvalues of the matrix $\left(\left(\boldsymbol{A}^{-1}\right)^T\boldsymbol{\Phi}\boldsymbol{A}^{-1}\right)$ are therefore

$$
\begin{aligned}
&\lambda_{\min}\left(\left(\boldsymbol{A}^{-1}\right)^T\boldsymbol{\Phi}\boldsymbol{A}^{-1}\right) = 0 \\
&\qquad\text{and}\qquad\qquad . \\
&\lambda_{\max}\left(\left(\boldsymbol{A}^{-1}\right)^T\boldsymbol{\Phi}\boldsymbol{A}^{-1}\right) = 1
\end{aligned}
\tag{C-24}
$$

From (C-15) and (C-24) we see that the upper and lower bounds on $J(n)$ are

$$0 \le J(n) \le 1 \tag{C-25}$$

and from (C-6) and (C-7) it follows that the angle parameter $\gamma(n)$ is bounded as follows

$$0 \le \gamma(n) \le 1. \tag{C-26}$$

C.1 References

[1] B. Noble and J. W. Daniel, *Applied Linear Algebra*, third edition, Prentice-Hall International, 1988.

Appendix D

In this appendix, we derive the second and up to the sixth order Global polynomial root finding methods based on the Taylor series expansion.

D.1 Notation and Problem Definition

Consider the following general, complex valued rth order polynomial in z

$$f(z) = z^r + a_1 z^{r-1} + \cdots + a_{r-1} z + a_r \tag{D-1}$$

with roots $\lambda_k \in C$, $k \in Z_r^+$ such that

$$f(z) = (z-\lambda_1)(z-\lambda_2)\ldots(z-\lambda_r). \tag{D-2}$$

We wish to find some iterative process that, when given a set of initial estimates $\{\lambda_k(0) \in C, k \in Z_r^+\}$ to the roots of the polynomial, will provide successively better estimates as the process is repeated. We shall denote the set of root estimates at the nth iteration of the process by $\{\lambda_k(n) \in C, k \in Z_r^+\}$

D.2 Derivation

We make, firstly, the following definitions:

$$u(\lambda_k(n)) = \frac{f(\lambda_k(n))}{f'(\lambda_k(n))} \tag{D-3}$$

$$A_m(\lambda_k(n)) = \frac{f^{(m)}(\lambda_k(n))}{m! f'(\lambda_k(n))} \tag{D-4}$$

$$T_m(\lambda_k(n)) = \sum_{i=1, i\neq k}^{r} \frac{1}{(\lambda_k(n) - \lambda_i(n))^m} \tag{D-5}$$

where the prime indicates the first derivative of $f(z)$ with respect to z. The definitions (D-3) and (D-4) were originally presented in Traub [1] while (D-5) appeared in Farmer and Loizou [2].

For the sake of brevity, references to the particular root, λ_k, and to the iteration time,

n, will be omitted from the above expressions in the derivation that follows. Therefore, it is assumed that

$$u = u(\lambda_k(n)) \tag{D-6}$$

$$A_m = A_m(\lambda_k(n)) \tag{D-7}$$

$$T_m = T_m(\lambda_k(n)). \tag{D-8}$$

With the above definitions, it can be readily shown that [2]

$$f'(\lambda_k(n)) = \prod_{i=1, i\neq k}^{r} (\lambda_k(n) - \lambda_i(n)) + O(\varepsilon) \tag{D-9}$$

$$A_2 = T_1 + O(\varepsilon) \tag{D-10}$$

$$2A_3 = A_2^2 - T_2 + O(\varepsilon) \tag{D-11}$$

$$3A_4 = 3A_2A_3 - A_2^3 + T_3 + O(\varepsilon) \tag{D-12}$$

where $O(\varepsilon)$ is a remainder term and $\varepsilon = \max_k \left|(\lambda_k(n) - \lambda_k)\right|$, $k \in Z_r^+$.

Consider the Taylor Series expansion of $f(z)$ at the root estimate $\lambda_k(n)$

$$f(z) = P_p(z) + \frac{f^{(p)}(c)}{p!}(z - \lambda_k(n))^p \tag{D-13}$$

where

$$P_p(z) = \sum_{i=0}^{p-1} \frac{f^{(i)}(\lambda_k(n))}{i!}(z - \lambda_k(n))^i \tag{D-14}$$

and where c lies in the closed interval determined by z and $\lambda_k(n)$.

Omitting the right hand remainder term in (D-13) leaves a polynomial of degree $p-1$ in $(z - \lambda_k(n))$ given by (D-14) and having the property

$$f(z) \cong P_p(z). \tag{D-15}$$

We wish to find some z such that $P_p(z) = 0$ and we will make this z the next approximation to the root, i.e., $\lambda_k(n+1)$ is found by solving

$$P_p(\lambda_k(n+1)) = 0. \tag{D-16}$$

As an example, consider the case where $p = 2$. Combining (D-14) and (D-16) and

expanding the summation, we get

$$P_2(z) = f(\lambda_k(n)) + f'(\lambda_k(n))(z - \lambda_k(n)) = 0. \tag{D-17}$$

Solving now for z

$$z = \lambda_k(n+1) = \lambda_k(n) - \frac{f(\lambda_k(n))}{f'(\lambda_k(n))} \tag{D-18}$$

which is Newton's well known method.

For $p > 2$ however, we observe that the above process of (D-17) and (D-18) requires the solution of a polynomial of degree $(p-1)$. To avoid this problem we can make certain degree reducing changes which allow us to *linearise* the polynomial without changing the order of the iteration function (see [1, p. 94]). To demonstrate this technique, consider the case where $p = 3$

$$P_3(z) = f(\lambda) + f'(\lambda)(z - \lambda) + \frac{f''(\lambda)(z-\lambda)^2}{2} = 0. \tag{D-19}$$

Note that the root number k and the iteration number n have been omitted from (D-19). To reduce the degree of the polynomial we replace one of the factors in $(z-\lambda)^2$ with $z - \lambda = -u$. (Recall from Newton's method that $z = \lambda - u$.) Thus

$$f(\lambda) + f'(\lambda)(z - \lambda) + \frac{f''(\lambda)(z-\lambda)(-u)}{2} = 0. \tag{D-20}$$

Solving for z and after some manipulation, we get

$$z = \lambda_k(n+1) = \lambda_k(n) - \frac{u}{1 - A_2 u} \tag{D-21}$$

which is Halley's method. (The root and iteration number references have been included again in (D-21) to avoid confusion.)

We will also briefly describe the replacement scheme for $p = 4$ as it shows how a general replacement scheme may be constructed for higher degree polynomials. As can be seen from (D-14), $P_4(z)$ has the non-linear terms $(z-\lambda)^2$ and $(z-\lambda)^3$. If we now replace one of the factors in each term with $z - \lambda = -\frac{u}{1 - A_2 u}$ from (D-21), and

another of the factors in the cubic term with $z-\lambda=-u$, we get Kiss' iteration function

$$\lambda_k(n+1)=\lambda_k(n)-\frac{u(1-A_2u)}{1-2A_2u+A_3u^2}. \tag{D-22}$$

Iteration functions derived using the procedure just described can be expressed in the general form

$$\lambda_k(n+1)=\lambda_k(n)-\Phi_p \tag{D-23}$$

where Φ_p is given by

$$\begin{aligned}\Phi_2&=u,\\ \Phi_j&=\frac{u}{1-\sum_{i=2}^{j-1}\left[(-1)^i A_i \prod_{l=j-i+1}^{j-1}\Phi_l\right]},\\ j&=3,4,\ldots,p.\end{aligned} \tag{D-24}$$

The above replacement scheme was initially presented by Farmer and Loizou [2].

So far, we have only derived local methods (which provide better estimates of one root only). To make use of the information we have on estimates of the other roots we replace the highest order A_m in Φ_p in the iteration functions obtained using (D-23) and (D-24) with one of the approximations of (D-9) to (D-12). For example, in Halley's method in (D-21) we replace A_2 with T_1 and get the third order global method given in (D-26). The second to sixth order global methods obtained in this way are presented below.

Second Order method

$$\lambda_k(n+1)=\lambda_k(n)-\frac{f(\lambda_k(n))}{\prod_{i=1;i\neq k}^{r}(\lambda_k(n)-\lambda_i(n))} \tag{D-25}$$

Third Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{u}{1 - T_1 u} \tag{D-26}$$

Fourth Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{(1 - A_2 u)u}{1 - 2A_2 u + 0.5\left(A_2^2 - T_2\right)u^2} \tag{D-27}$$

Fifth Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{\left(1 - 2A_2 u + A_3 u^2\right)u}{1 - 3A_2 u + \left(2A_3 + A_2^2\right)u^2 - \left(3A_2^2 A_3 - A_2^3 + T_3\right)u^3/3} \tag{D-28}$$

Sixth Order method

$$\lambda_k(n+1) = \lambda_k(n) - \frac{[NUMERATOR]}{[DENOMINATOR]}$$

where

$$[NUMERATOR] = \left(1 - 3A_2 u + \left(2A_3 + A_2^2\right)u^2 - A_4 u^3\right)u \tag{D-29}$$

$$[DENOMINATOR] = 1 - 4A_2 u + \left(3A_3 + 3A_2^2\right)u^2 - \left(2A_2 A_3 + 2A_4\right)u^3 + \left(A_2^4 - 4A_2^2 A_3 + 4A_2 A_4 + 2A_3^2 - T_4\right)u^4/4$$

D.3 References

[1] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice-Hall Series in Automatic Computation, Englewood Cliffs, 1964.

[2] M. R. Farmer and G. Loizou, "A Class of Iteration Functions for Improving, Simultaneously, Approximations to the Zeros of a Polynomial", *BIT*, vol. 15, pp 250 - 258, 1975.

Appendix E

In this appendix, we describe a method of simulating the narrowband array processor input vector.

E.1 Generation of Gaussian Random Variables

Let U_1 and U_2 be two independent random variables uniformly distributed on the interval $\{0 \leftrightarrow 1\}$. We can generate two independent Gaussian random variables X and Y using U_1 and U_2 as follows:

$$X = \sigma_X\left[-2\ln(U_1)\right]^{1/2}\cos(2\pi U_2) + \mu_X \tag{E-1}$$

$$Y = \sigma_Y\left[-2\ln(U_1)\right]^{1/2}\sin(2\pi U_2) + \mu_Y \tag{E-2}$$

where X has variance σ_X^2 and mean μ_X, and Y has variance σ_Y^2 and mean μ_Y.

E.2 Generation of the Array Input Vector

We wish to simulate the array processor input vector for a farfield narrowband source with centre frequency ω_c in the direction (θ,ϕ). It is assumed that that the source can be modelled as a stationary, zero mean, Gaussian distributed, stochastic process.

We begin by constructing the vector $s(\theta,\phi)$, known as the steering vector, given by

$$s(\theta,\phi) = \left[e^{j\omega_c\tau_1(\theta,\phi)}, e^{j\omega_c\tau_2(\theta,\phi)}, \ldots, e^{j\omega_c\tau_L(\theta,\phi)}\right]^T \tag{E-3}$$

where the $\tau_i(\theta,\phi), i \in Z_L^+$ are given by (2-2).

We then generate two independent Gaussian random variables $s_I(t)$ and $s_Q(t)$ as described in Section E.1. with zero mean and unit variance, i.e.,

$$\begin{gathered} \mathrm{E}\left[s_I(t)s_Q(t)\right] = 0 \\ \text{and} \\ \mathrm{E}\left[s_I(t)s_I(t)\right] = \mathrm{E}\left[s_Q(t)s_Q(t)\right] = 1. \end{gathered} \tag{E-4}$$

Using these, we calculate the vector $\boldsymbol{x}(t) \in C^L$ given by

$$\boldsymbol{x}(t) = p_s\big(s_I(t) + js_Q(t)\big)\boldsymbol{s}(\theta,\phi) \tag{E-5}$$

where p_s is the mean square power on the array elements due to the source.

From (E-3) and (E-5), the real and imaginary parts of the ith element of $\boldsymbol{x}(t)$ can be written in expanded form as

$$x_{Ii}(t) = p_s s_I(t)\cos\big(\omega_c\tau_i(\theta,\phi)\big) - p_s s_Q(t)\sin\big(\omega_c\tau_i(\theta,\phi)\big) \tag{E-6}$$

and

$$x_{Qi}(t) = p_s s_I(t)\sin\big(\omega_c\tau_i(\theta,\phi)\big) + p_s s_Q(t)\cos\big(\omega_c\tau_i(\theta,\phi)\big) \tag{E-7}$$

Identifying $s_I(t)$ with $m_I(t)$ and $s_Q(t)$ with $m_Q(t)$ in (2-13) and (2-14), it follows that (E-6) and (E-7) can be used to simulate the in-phase and quadrature signals for the ith antenna element. Note that p_s is given by

$$p_s = \sqrt{\text{Source Signal Power}} \tag{E-8}$$

To simulate Gaussian distributed element self-noise, $2L$ independent, Gaussian distributed, zero mean, unit variance random variables are generated as discussed in Section E.1 and then added to the above expressions as follows

$$x_{Ii}(t) = p_s s_I(t)\cos\big(\omega_c\tau_i(\theta,\phi)\big) - p_s s_Q(t)\sin\big(\omega_c\tau_i(\theta,\phi)\big) + p_n n_{Ii}(t) \tag{E-9}$$

and

$$x_{Qi}(t) = p_s s_I(t)\sin\big(\omega_c\tau_i(\theta,\phi)\big) + p_s s_Q(t)\cos\big(\omega_c\tau_i(\theta,\phi)\big) + p_n n_{Qi}(t) \tag{E-10}$$

where $p_n(t)$ determines the mean square value of the element self-noise for all of the array elements and is given by

$$p_n = \sqrt{\text{Element Self Noise Power}} \tag{E-11}$$

Appendix F

In this appendix, we show that the optimisation problem for a minimum variance narrowband antenna array processor with NS derivative constraints is not affected by the phase of the look direction constraint.

In Chapter 2 we showed that the optimum weight vector for a minimum variance narrowband array processor with NS derivative constraints can be found from the following quadratically constrained optimisation problem

$$\mathcal{OP1} \qquad \min_{w_1} \; w_1^H R w_1 \tag{F-1}$$

$$\text{subject to} \qquad \begin{aligned} & s^H w_1 = e^{j\gamma} \\ & w_1^H A_i w_1 = 0, \; i = 1,2,\ldots,5 \end{aligned} \tag{F-2}$$

where the A_i are given by (2-50).

In this appendix we show that the optimum output power for this processor is independent of the choice of γ and that $\mathcal{OP1}$ can be reformulated as the optimisation problem given in $\mathcal{OP2}$ without loss of generality.

$$\mathcal{OP2} \qquad \min_{w_2} \; w_2^H R w_2 \tag{F-3}$$

$$\text{subject to} \qquad \begin{aligned} & s^H w_2 = 1 \\ & w_2^H A_i w_2 = 0, \; i = 1,2,\ldots,5 \end{aligned} \tag{F-4}$$

Let $\boldsymbol{y} = \boldsymbol{w}_1 e^{-j\gamma}$ in $\mathcal{OP}1$.

$\mathcal{OP}1$ can then be written as follows

$$\min_{\boldsymbol{y}} \ \boldsymbol{y}^H \boldsymbol{R} \boldsymbol{y} \tag{F-5}$$

$$\text{subject to} \quad \boldsymbol{s}^H \boldsymbol{y} = 1 \qquad \boldsymbol{y}^H \boldsymbol{A}_i \boldsymbol{y} = 0, \ i = 1,2,\ldots,5 \tag{F-6}$$

which is the same optimisation problem as $\mathcal{OP}2$.

Hence

$$\boldsymbol{y}_{opt} = \boldsymbol{w}_{1opt} e^{-j\gamma} = \boldsymbol{w}_{2opt} \tag{F-7}$$

and

$$\begin{aligned} \boldsymbol{y}_{opt}^H \boldsymbol{R} \boldsymbol{y}_{opt} &= \boldsymbol{w}_{1opt}^H e^{j\gamma} \boldsymbol{R} \boldsymbol{w}_{1opt} e^{-j\gamma} \\ &= \boldsymbol{w}_{1opt}^H \boldsymbol{R} \boldsymbol{w}_{1opt} \end{aligned} \tag{F-8}$$

Therefore

$$\boldsymbol{w}_{2opt}^H \boldsymbol{R} \boldsymbol{w}_{2opt} = \boldsymbol{w}_{1opt}^H \boldsymbol{R} \boldsymbol{w}_{1opt}, \tag{F-9}$$

that is, the optimum output power for the array processor is independent of γ and the elements of the optimum weight vector are simply rotated by γ.