Cascade Adaptive Array Structures

A Dissertation Presented to
The Faculty of the College of Engineering and Technology of
Ohio University

In Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

by
Timothy B. Hanson
June 1990
Acknowledgements

I would like to express my thanks to Dr. Joseph Essman for his patience and numerous suggestions that made this dissertation possible. Also I wish to thank the department of Electrical and Computer Engineering for its financial support. Thanks go to the members of my committee: Drs. Brown, Norris, Radcliff, and Tague for their interest and time.

Thanks go to my parents for listening to my complaints without complaint. Most especially, I appreciate the contribution of my wife Nadine without whom I could have never successfully undertaken the return to student status. Pourvue qu'elle n'ait plus jamais besoin de supporter un calvaire pareil. Cette thèse comme tout est dédiée à elle. Enfin merci à mon copain, Boule, qui lui aussi a supporté son petit calvaire à sa façon.
# Table of Contents

List of Figures vii
List of Tables ix
Notation Conventions x

Chapter 1 - Adaptive Arrays and Spatial Filtering

1.0 Background 1
1.1 The Wiener-Hopf Equation 4
1.2 Adaptive Signal Processing Algorithms 7
  1.2.1 The Least Mean Squares (LMS) Algorithm 8
  1.2.2 Sample Matrix Inversion 9
1.3 Previous Work by Others 10
1.4 Advantages of a Cascade Approach 12
1.5 Outline of Research 13

Chapter 2 - Antenna Arrays

2.0 Background 17
2.1 Adaptive Array Configuration 18
2.2 Definition of Narrowband Input Signals 20
2.3 Calculation of the Covariance Matrix 23
2.4 Eigenvalues of the Covariance Matrix 25
2.5 Array Pattern 33
2.6 Array Gain Pattern Nulls and Array Degrees of Freedom 34
2.7 Frost Beamformer 38
2.8 Formula for Array Output SINR 39
2.9 Grating Nulls and Subarray Spacing 43
2.10 Summary 46

Chapter 3 Optimum Cascade Arrays

3.0 Background 47
3.1 Cascade Array Configuration 48
3.2 Partially Adaptive Arrays 50
3.3 Subarray Covariance Matrices 51
  3.3.1 Equivalent Subarrays - Equal Covariance Matrices 53
3.4 Cascade Preprocessing 55
3.5 Cascade Gain Formula 55
3.6 Subarrays with Optimal Performance 57
3.7 Suboptimal Performance with Reduced Computational Complexity 63
3.8 Cascade SINR 64
References

Abstract
### List of Figures

1.1 Adaptive Antenna Array .................................. 3
2.1 Narrowband Processing ...................................... 19
2.2 Wideband Processing ....................................... 19
2.3 Interelement Path Difference .............................. 21
2.4 Davies Nulling Tree ....................................... 37
3.1 General Cascade Array ..................................... 49
3.2 Cascade Array ............................................... 58
3.3 Two Element Subarrays ..................................... 75
3.4 Three Element Subarrays ................................... 78
3.5 Four Element Subarrays .................................... 78
3.6 Four Stage Cascade ......................................... 79
3.7 Optimum 3 Element Array Pattern .......................... 83
3.8 Two Element Cascade ....................................... 86
3.9 Optimum 3 Element Array Pattern with 2 Element Subarrays .... 89
3.10 Five Element Array with 2 Element Subarrays .............. 93
3.11 Five Elements Pattern with 2 Element Subarrays .......... 93
3.12 Five Element Array with 3 Element Subarrays .............. 95
3.13 Five Element Array with 4 Element Subarrays .............. 99
3.14 Optimum 5 Element Array Pattern with 4 Element Subarrays .... 100
3.15 $\epsilon_i = 1$, SINR versus Interference Angle .......... 105
3.16 $\epsilon_i = 10$, SINR vs. Interference Angle ................ 105
3.17 $\epsilon_i = 100$, SINR vs. Interference Angle .............. 106
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.18</td>
<td>$\epsilon_i=1000$, SINR vs. Interference Angle</td>
<td>106</td>
</tr>
<tr>
<td>4.1</td>
<td>Three Element SMI Array Pattern</td>
<td>117</td>
</tr>
<tr>
<td>4.2</td>
<td>Five Element SMI Pattern</td>
<td>119</td>
</tr>
<tr>
<td>4.3</td>
<td>Three Element Array with 2 Element Subarrays Pattern</td>
<td>121</td>
</tr>
<tr>
<td>4.4</td>
<td>Five Element Array with 2 Element Subarrays Pattern</td>
<td>123</td>
</tr>
<tr>
<td>4.5</td>
<td>Five Element Array with 3 Element Subarrays Pattern</td>
<td>125</td>
</tr>
<tr>
<td>4.6</td>
<td>Five Element Array with 4 Element Subarrays Pattern</td>
<td>127</td>
</tr>
<tr>
<td>5.1</td>
<td>Coherent Interference Pattern</td>
<td>136</td>
</tr>
<tr>
<td>5.2</td>
<td>Su's Spatial Dithering Technique</td>
<td>140</td>
</tr>
<tr>
<td>5.3</td>
<td>Array Output with Signal Cancellation</td>
<td>143</td>
</tr>
<tr>
<td>5.4</td>
<td>3x2 with Coherent Interference</td>
<td>150</td>
</tr>
<tr>
<td>5.5</td>
<td>5x3 with Coherent Interference</td>
<td>152</td>
</tr>
<tr>
<td>5.6</td>
<td>5x2 with Coherent Interference</td>
<td>154</td>
</tr>
<tr>
<td>6.1</td>
<td>Quadrature Hybrid</td>
<td>158</td>
</tr>
<tr>
<td>6.2</td>
<td>Noisy Input Signal Antenna 1</td>
<td>165</td>
</tr>
<tr>
<td>6.3</td>
<td>In-phase Array Weights</td>
<td>165</td>
</tr>
<tr>
<td>6.4</td>
<td>Quadrature Array Weights</td>
<td>166</td>
</tr>
<tr>
<td>6.5</td>
<td>Three Element Output Error</td>
<td>166</td>
</tr>
<tr>
<td>6.6</td>
<td>Three Element Array Output</td>
<td>167</td>
</tr>
<tr>
<td>6.7</td>
<td>Three Element Pattern</td>
<td>168</td>
</tr>
<tr>
<td>6.8</td>
<td>Five Element LMS Array Pattern</td>
<td>170</td>
</tr>
<tr>
<td>6.9</td>
<td>Five Element Array with 3 Element Subarrays Pattern</td>
<td>172</td>
</tr>
<tr>
<td>6.10</td>
<td>Five Element Array with 4 Element Subarrays Pattern</td>
<td>174</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Optimum Performance Comparison</td>
<td>70</td>
</tr>
<tr>
<td>3.2</td>
<td>Example Performance Comparison</td>
<td>102</td>
</tr>
<tr>
<td>3.3</td>
<td>Example dB Off Optimum Performance</td>
<td>103</td>
</tr>
<tr>
<td>4.1</td>
<td>Performance Comparison Cascaded SMI</td>
<td>129</td>
</tr>
<tr>
<td>4.2</td>
<td>Simulated SMI Performance Comparison</td>
<td>130</td>
</tr>
<tr>
<td>4.3</td>
<td>Simulated SMI Performance Off Optimal Cascade</td>
<td>131</td>
</tr>
<tr>
<td>4.4</td>
<td>Simulated SMI Performance Off Optimal Non-cascade</td>
<td>132</td>
</tr>
<tr>
<td>6.1</td>
<td>Comparison of Cascaded and Non-cascaded LMS Performance</td>
<td>177</td>
</tr>
</tbody>
</table>
Notation

$A_d$ magnitude of desired signal
$A_i$ magnitude of interference
$d$ array inter-element spacing
$d_k$ array reference signal
$\epsilon_d$ desired signal to thermal noise ratio
$\epsilon_i$ interference to thermal noise ratio
$E(\cdot)$ expected value
$g(\theta)$ array gain
$g_c(\theta)$ cascade array gain
$G_i$ stage transformation matrix
$\lambda_i$ covariance matrix eigenvalue
$\lambda'_i$ normalized covariance matrix eigenvalue
$\mu$ LMS algorithm step size
$M$ number of subarrays
$N$ number of array elements
$O(\cdot)$ computational complexity
$\Phi_d$ desired signal inter-element phase shift
$\Phi_i$ interference inter-element phase shift
$P_d$ desired signal power out of array
$P_i$ interference power out of array
$P_n$ noise power out of array
$P_{\overline{\mathbf{X}}}$ cross correlation vector
$P_{\overline{\mathbf{x}}}$ cross correlation subarray
$P_y$ cross correlation cascade array
$Q_i$ subarray selection matrix
\[ R_{xx} \] input covariance matrix
\[ R'_{xx} \] normalized input covariance matrix
\[ R_{yy} \] second stage covariance matrix
\( \sigma \) thermal noise standard deviation
\[ s_k \] array output signal
\[ \theta_d \] desired signal angle
\[ \theta_i \] interference angle
\[ \text{Tr}[\cdot] \] trace of a matrix
\[ \mathbf{U}_d \] desired signal phase shift vector
\[ \mathbf{U}_i \] interference phase shift vector
\[ \mathbf{U}_t \] test signal phase shift vector
\[ \mathbf{W} \] array weight vector
\[ \mathbf{W}_{eq} \] equivalent weight vector
\[ \mathbf{W}_{est} \] estimated weight vector
\[ \mathbf{X} \] array input vector
\[ \mathbf{X}_d \] desired signal vector
\[ \mathbf{X}_i \] interference vector
\[ \mathbf{X}_n \] noise vector
\( (\cdot)^* \) complex conjugate
\( (\cdot)^H \) complex conjugate transpose
\( (\cdot)^T \) transpose
Chapter 1

Adaptive Arrays and Spatial Filtering

1.0 Background

One of the basic tasks of communication engineering is the design of optimum systems for the transmission and reception of information. When the channel and interference are stationary a fixed optimal processing method may be implemented. The resulting filter is non-time varying. Example fixed filter systems are the Wiener-Hopf filter and the matched filter. However, time varying channels and non-stationary interference occur in many environments. If the communications path is time varying, the channel is non-stationary and fixed techniques can be used only if the designer is willing to settle for a compromise system that performs optimally on the average. If globally optimum performance is desired then adaptive signal processing is the natural choice. The system will then adapt to the changing characteristics of the channel and interference. An adaptive system yields globally better performance.

When used for spatial filtering with an array antenna, an adaptive filter exploits the difference in spatial orientation of emitters with respect to the antenna array. Adaptive interference nulling with an antenna system is the subject of this dissertation. If the noise or interference is coming from a different direction from that of the desired signal an adaptive filter (linear combiner) can be used to suppress the
noise or interference by placing a null of the antenna pattern in the direction of the noise or interference. The pattern of an antenna array is altered by amplitude and phase weighing the array inputs. Using an adaptive signal processing algorithm an antenna array may sense and respond to a changing interference environment. An adaptive antenna array is composed of an array of antenna elements, a signal processor, a control algorithm, and a pattern-forming network (linear combiner). Figure 1.1 shows the structure of a narrowband adaptive antenna array. An adaptive antenna array consists of both a linear combiner and an algorithm for adjusting the combiner weights. There are many possible choices for the weight control algorithm. The array output is a linear combination of the array inputs. Adaptive signal processing algorithms find the location of the optimum weights on the performance surface. Gradient search methods can be used to search the performance surface for the minimum point. Numerous adaptive signal processing algorithms exist. For example, Newton's method, the method of steepest descent, the least mean square LMS method, a random search, sample matrix inversion, and recursive least squares methods may be used to find the optimum weight values. At convergence all of these methods approximate the optimum Wiener-Hopf solution, given in Section 1.1.

Two of the major concerns in adaptive antenna systems are computational complexity and rate of convergence.
Figure 1.1 Adaptive Antenna Array
The subject investigated in this dissertation is the application of a divide and conquer approach to array processing. That is, the array is partitioned and processed in subarrays. The subarrays are then cascaded in a tree structure. The major concerns involved in partitioning an antenna array into subarrays are identified and answered. Performance advantages of partitioning are identified.

Two algorithms are used for illustration in this dissertation. These methods are the LMS and SMI algorithms. The LMS algorithm is particularly useful because of its computational simplicity. The advantage of the SMI algorithm is its fast convergence speed. It is shown in Chapter 6 that the LMS algorithm convergence is improved by partitioning. In Chapter 4, the major benefit of partitioning for the SMI algorithm is a reduction in computational complexity. Additional benefits are discussed in Chapters 4 and 6. Partitioning of the inputs and cascading is the subject of this dissertation.

1.1 The Wiener-Hopf Equation

The Wiener-Hopf Equation is discussed here because it forms the basis for optimum antenna array processing. The classical method of interference canceling is to add a fixed filter to remove noise from the signal. The theory for doing this optimally was developed by Wiener for the non-recursive case and Kalman for the recursive case. The design of fixed
filters requires prior knowledge of the characteristics of the signal and the noise.

Adaptive filtering can be done both spatially and temporally. Among other performance criteria, the adaptive process can be based upon minimizing the mean square error (MSE) between the output of the system and a reference signal. However, adaptation may also be based on minimum bit error rate (BER) or maximum signal to interference and noise ratio (SINR). The choice of the mean square error performance criteria leads to a performance surface that is parabolic in weight space with a unique minimum. This assures that if a gradient procedure is used to find a minimum point it finds a global minimum and not merely a local minimum. Choosing the mean square error criteria leads to the Wiener-Hopf equation. The Wiener-Hopf equation can then be solved for the optimum weights of the linear combiner. The mean square error criterion is used in this work.

In order to find the optimum weights values consider minimizing the mean square error between the output of the array, \( s(t) \), and a reference signal, \( d(t) \).

\[
E[|d(t)-s(t)|^2] 
\]  \hspace{1cm} (1.1.1)

The output of the array is the inner product of the vector of the linear combiner weights \( \mathbf{W} \) with the vector of the inputs \( \mathbf{X} \), i.e.,

\[ \mathbf{w}^T \mathbf{x} \]

\[ ^1 \text{Boldface is used to denote vector or matrix quantities.} \]
where \( \mathbf{W}^H \) is a complex conjugate transpose of the combiner weights \([ w_1, w_2, \ldots, w_n ]^\top\) and \( \mathbf{X} \) is a vector of the array element inputs \([ x_1, x_2, \ldots, x_n ]^\top\). The array reference signal is \( d(t) \). Substituting (1.1.2) into the mean square error (1.1.1) and expanding yields

\[
E[ |d(t)|^2 - 2d(t)\mathbf{W}^H\mathbf{X} + (\mathbf{W}^H\mathbf{X})^2 ]
\]

(1.1.3)

\[
E[ |d(t)|^2 - 2\mathbf{W}^H E[d(t)\mathbf{X}^\top] + \mathbf{W}^H \mathbf{X}^\top \mathbf{X} \mathbf{W}^H]
\]

\( E(d(t)\mathbf{X}^\top) \) can be written as

\[
P_n = E[d(t)\mathbf{X}^\top]
\]

(1.1.4)

\( P_n \) is the cross-correlation between the input vector and the reference signal. \( E(\mathbf{X}^\top \mathbf{X}^\top) \) is called the covariance matrix of the inputs \( \mathbf{R}_{xx} \). The mean square error is a function of the linear combiner weights \( w_1, w_2, \ldots, w_n \). Substituting for \( E(d(t)\mathbf{X}^\top) \) and \( E(\mathbf{X}^\top \mathbf{X}^\top) \) in (1.1.3) yields the following for the mean square error.

\[
e(w_1, w_2, \ldots, w_n) = E[ |d(t) - s(t)|^2 ]
\]

\[
= E[ |d(t)|^2 ] - 2\mathbf{W}^H P_n + \mathbf{W}^H \mathbf{R}_{xx} \mathbf{W}
\]

(1.1.5)

In order to find the optimum array weights the partial derivative of the mean square error is taken with respect to each of the linear combiner weights and the result is set equal to zero.
The resulting set of equations is then solved for the optimum set of weights. The resulting system of equations is written as

\[ R_{xx} \hat{w}_{opt} = P_N \]  \hspace{1cm} (1.1.6)

Inversion of \( R_{xx} \) yields the optimum weight solution

\[ \hat{w}_{opt} = R_{xx}^{-1} P_N \]  \hspace{1cm} (1.1.7)

Equation (1.1.7) is known as the Wiener-Hopf equation.

1.2 Adaptive Signal Processing Algorithms

Although SMI requires estimation and inversion of the \( R_{xx} \) matrix, other adaptive signal processing algorithms do not require the estimation of the input covariance matrix \( R_{xx} \) and its inversion. Adaptive signal processing algorithms lower the computational complexity of finding the optimum weight values by avoiding the direct calculation of \( R_{xx} \) and its inversion. The algorithms fit into several categories. There are gradient based, recursive least squares techniques based, and SMI based methods. [37] Very fast adaptation algorithms are necessary for military radar, sonar, navigation, and communication equipment. Algorithms may be classified on the basis of computational complexity. A bold \( O(\cdot) \) is used to indicate order of computational complexity. The LMS algorithm is \( O(N) \), i.e., the complexity is
proportional to the number of antenna elements. The recursive least squares algorithm is $O(N^2)$. Sample matrix inversion is $O(N^3)$. LMS and SMI are at opposite ends of the complexity range and for this reason were chosen as examples.

1.2.1 The Least Mean Squares (LMS) Method

The LMS algorithm is a gradient method of searching the performance surface to find the location of the optimum weights at the performance surface minimum. [52] A special estimate of the gradient that is valid for linear combiner is used by the LMS algorithm. This estimate requires that the input vector and the desired response be available at each iteration.

The LMS method operates without knowledge of the direction of arrival of the desired signal, and without knowledge of the interference and noise. No direct measurement of the noise and interference is necessary. The LMS method is computationally simple. It does not require computation or inversion of input covariance matrix $R_{xx}$. The input signals are used only once (as they occur in the adaptation process) and they do not need to be stored. Only the values of the linear combiner weights need to be stored.

When gradient descent algorithms are used the time constants for the combiner weight transient are inversely proportional to the eigenvalues of the input covariance matrix. [52] Thus the modes associated with the interference
or noise having less power converge more slowly than those modes associated with noise or interference having greater power. For a large variation in input power the LMS algorithm will converge slowly. However, modified gradient techniques may be used to reduce the eigenvalue disparity. [39] The maximum eigenvalue of the covariance matrix is directly proportional to the size of the array. [9] Cascading arrays reduces the eigenvalue spread. In addition, the eigenvalue spread of succeeding stages is reduced. The cascaded LMS algorithm form the subject of Chapter 6.

1.2.2 Sample Matrix Inversion

The technique that converges the fastest is sample matrix inversion. [44] If the input covariance matrix $R_{xx}$ is known apriori it is sufficient to simply invert the matrix $R_{xx}$ and multiply the result onto $P_n$ to find the optimum array weights. In practice the input covariance matrix $R_{xx}$ must be estimated from the data. To assure accuracy in the estimation a number of input samples proportional to the array size must be used. The larger the array the more input samples must be used for accurate $R_{xx}$ matrix estimation.

Techniques such as sample matrix inversion become computationally infeasible in real-time as the number of antenna elements increases. If a large array is broken into subarrays then more complex algorithms such as SMI become feasible with the subarrays. Cascaded sample matrix inversion
is discussed in Chapter 4.

1.3 Previous Work by Others

Little is known about the convergence of cascaded adaptive filters. An example of cascaded adaptive filtering in the literature is the interaction between an echo canceller and a line equalizer considered in Van Gerwen et al. [56]. Van Gerwen shows that three different convergence phases exist and that the convergence time is considerably longer than one would expect from either filter alone.

In Eilts' [14] "Cascade Adaptive Arrays", the main array is broken up into subarrays over which a power inversion is done. The inputs are processed by power inversion arrays prior to using the LMS algorithm. The power inversion arrays attenuate the stronger signals and reduce the eigenvalue spread. The LMS algorithm convergence speed is limited by the eigenvalue spread of the input covariance matrix and the power inversion procedure helps the convergence of the LMS algorithm. In Eilts the term cascade array means the cascade of power inversion arrays and the LMS algorithm. The inputs are divided into separate subarrays over which the power inversion method is used. This is then followed by the LMS algorithm.

In White [54], "Adaptive Cascade Networks for Deep Nulling", using the Davies cascade array configuration and breaking up the array into subarrays of two elements each,
White shows that the cascade network will settle more rapidly than a conventional adaptive network when the eigenvalues are widely separated.

In Khanna and Madan [32], "Adaptive Beamforming Using a Cascade Configuration", an N-element array is reconfigured as a number of subarrays of 2 elements each, (Figure 3.3). This is known as the Davies adaptive array. Each of the two element subarrays is capable of steering one null in the direction of an interference source while maintaining gain in the desired signal direction. One of the problems with Khanna's approach is that the nulls produced do not point directly at the direction of the interference. For instance if there are two interferers of equal power at different angles the first stage will place a null between the two interferences rather than at either one in particular. The gain is maintained in the desired signal direction by a Frost constraint. The direction of the desired signal must be known a priori. At each stage in the cascade array the noise source contributing maximum power at that stage is nulled out. This approach has been shown by simulation to have fast convergence. Using Khanna and Madan's approach each adaptation requires only a 2x2 matrix inversion. Khanna and Madan use sample matrix inversion. The cascade configuration investigated by Khanna and Madan is based upon the Davies (Figure 2.4) cascade array configuration. Since at each stage of the array only two inputs are being dealt with, the
computations are particularly simple. The first stage of the array nulls out the strongest interference source. The second stage of the network nulls out the second strongest interfering source and so on. Adaptive techniques based on the gradient exhibit slow convergence because of the conflicting requirements of fast adaptation rate and small misadjustment noise. Fast adaptation requires a large step size, while small misadjustment rate requires a small step size. Gradient algorithms exhibit slow convergence when the eigenvalues of the input covariance matrix have a large spread.

1.4 Advantages of a Cascade Approach

No one has yet investigated cascading of arrays in general terms. The author will show that the following are advantages of cascade arrays:

- A modular time division multiplexed (TDM) processor using common software through the cascade is possible.
- The reduced dimensionality yields processors of lower complexity.
- Eigenvalue spread is reduced through the use of smaller arrays with corresponding convergence rate improvement when using gradient based algorithms [7], as shown in Section 2.4.
- When smaller arrays are used they will converge faster than a large array when using the least mean square (LMS) technique due to the reduction in the eigenvalues of the $R_{xx}$ matrix [7], as shown in Section 2.4.
• A smaller word size is necessary to accommodate lowered
eigenvalue spread at the cascade stages.
• Estimation delay decreases over that of a larger array when
using the sample matrix inversion (SMI) technique [44], as
shown in Chapter 4.
• Sensitivity to weight errors is decreased with a cascade
array due to the corrective effect of succeeding stages.
• Reliability is enhanced through the use of multiple smaller
signal processors. If a single processor fails another of a
pool of processors may take its place in the array or its
output may be suppressed by succeeding signal processors.
• A configuration of multiple smaller signal processors is
more economic. The multiple smaller signal processors used in
the cascade approach cost less than a single large signal
processor.
This dissertation demonstrates the previous advantages of a
cascade approach.

1.5 Outline of Research

Mathematical models and simulation software are developed
for cascade adaptive arrays. Optimal non-cascade arrays are
simulated for comparison with cascade arrays. The condition
for a cascade array to have the same (after convergence)
performance as a non-cascade array is developed in Section
3.6. In Chapters 4 and 6 the convergence of two cascaded
adaptive algorithms, sample matrix inversion (SMI) and least
mean square (LMS), is demonstrated through simulation and compared to that of non-cascade arrays using the same algorithms. A modular structure is discussed in Section 3.8 that allows the use of multiple smaller signal processors configured in a cascade. This modular structure is partially adaptive. Performance is degraded from that of a fully adaptive array. Formulas for the cascade gain and SINR are developed. The modular performance is compared to fully adaptive arrays in terms of SINR and mean square error at convergence. High complexity algorithms such as sample matrix inversion benefit greatly from the reduced dimensionality in the cascade as seen in Section 4.3. Dimensionality of the input vector is the primary difficulty in implementing SMI. The cascade reduces computational complexity at the cost of reduced SINR performance.

Adaptive subarrays are investigated in Chapter 3. There are numerous engineering tradeoffs that have not been fully investigated in the adaptive arrays literature. The subject of cascaded arrays shares aspects of partially adapted arrays and array preprocessors. Examples of both partially adapted arrays [4,27,38,57,58,59] and array preprocessors [32,55] can be found.

Partially adaptive arrays are useful in situations where the number of elements of the array and the speed of processing required are such that the signal processing
element of the adaptive processor is not capable of the necessary processing speed. An alternative to adapting over all the elements of the array is to select only certain elements of the array to adapt over. This results in a processor of lower complexity, but performance suffers. It is shown that cascading adaptive arrays can reduce processor complexity by breaking a large array up into a number of smaller subarrays that can then be processed by a smaller, cheaper signal processor. If the stage transformation is invertible then the converged performance of a cascade array is shown to be equivalent to that of a fully adapted array. Thus a steady-state performance penalty need not be paid, but in order to reduce dimensionality the array is made partially adaptive.

The cascade adaptive filter situation occurs in the cascaded array case and the convergence performance and interaction of cascaded adaptive filters is investigated. The convergence performance of both the sample matrix inversion technique SMI (Chapter 4) and the least mean square LMS algorithm (Chapter 6) is demonstrated. The simulation of cascade sample matrix inversion SMI and least mean square LMS techniques is set up as an example.

Rejection of coherent interference is a problem for adaptive arrays. An additional benefit of using a cascade is that the covariance matrices of the subarrays may be averaged
to spatially dither the antenna receiving point [46,53]. This permits the array to reject coherent interference. This forms the subject of Chapter 5. Various cascade configurations are evaluated. The simulation is set up so that spatial dithering is a natural by product of the cascade subarrays. Spatial dithering can modulate coherent interference by electronically jiggling the array. The array can then differentiate between the signal and coherent interference. An additional development is that the subarrays may be adaptively combined in a cascade instead of sacrificing array aperture. A formula for the resulting SINR is derived. Chapter 2 establishes necessary background to answer questions about cascade antenna arrays.
Chapter 2

Antenna Arrays

2.0 Background

This Chapter provides background that is used in the following chapters. Formulas will be derived for the optimum weights of an antenna array. The eigenvalues of $R_{xx}$ are then discussed in Section 2.4. The values of the eigenvalues are important for the discussion of the cascade LMS algorithm in Chapter 6. Formulas for the array pattern are also established for use in succeeding Chapters. A formula for the array SINR is established for later use, and finally the subject of grating nulls is established. The subject of grating nulls is important in subarray spacing.

The first application of adaptive filters to antennas was the sidelobe canceller of Applebaum whose purpose it was to place the antenna sidelobes so as to maximize the SINR (signal to interference and noise ratio). [3] Adaptive filtering can be used in both a spatial and a temporal sense. The adaptive process is usually based on minimizing the mean square error. The choice of the mean square error performance criteria leads to a performance surface with a unique minimum and tractable mathematics. When used for spatial filtering with an array, adaptive filters can exploit the difference in spatial orientation of emitters with respect to an array. If the noise or interference is coming from a different direction from that of the desired signal then an adaptive filter can be used to suppress the noise or interference power by placing nulls of the antenna pattern in the direction of the undesired
signals. An adaptive antenna array may be either narrow-band or wideband as represented in Figures 2.1 and 2.2.

The pattern of an antenna array may be altered by amplitude and phase weighing the array inputs. Using an adaptive signal processing algorithm an array may sense and respond to a changing signal environment. An adaptive array is composed of an array of elements, a signal processor, a control algorithm, and a pattern-forming network.

An array consists of a set of sensors whose outputs can be combined to produce a desired directivity pattern. Array processing is currently being used to produce directivity nulls in the direction of undesired interference. The adaptive array adjusts its directional pattern so as to maximize the signal to noise ratio. Adaptation of the directional pattern is done electronically through signal processing techniques.

The basis for finite impulse response (FIR) adaptive techniques is the linear combiner or transversal filter. The linear combiner may be viewed in two ways. These are the single input and the multiple input case. In the single input case the inputs to the linear combiner are sequential time samples. In the multiple input case the inputs to the linear combiner are simultaneous inputs from different sensors (antennas).

2.1 Adaptive Array Configuration

Referring to the Fig. 2.3 of an antenna array with a planewave arriving from direction \( \theta \), if a planewave signal arrives at an
Figure 2.1 Narrowband Processing

Figure 2.2 Wideband Processing
antenna array from direction \( \theta \) then the path difference between array elements with interelement spacing \( d \) is

\[
d \sin(\theta)
\]

The phase delay is

\[
\phi_d = 2\pi \frac{d}{\lambda} \sin(\theta)
\]

2.2 **Narrow-band Input Signals**

The examples of succeeding Chapters are confined to narrow-band arrays. Analytical formulas for narrow-band arrays are less complex than those for broadband arrays. To date most work in partially adaptive arrays has treated the narrow-band case. For narrow-band processing two real weights or one complex weight are necessary for each input of the linear combiner. Figure 2.1 shows the narrow-band case. The real weight is the in-phase weight \( W_{\text{inphase}} \) and the in parallel imaginary weight is the quadrature weight \( W_{\text{quad}} \). The \( W_{\text{quad}} \) weight corresponds to a 90° phase shifter followed by a real weight. Two weights \( W_{\text{inphase}} \) and \( W_{\text{quad}} \) in conjunction with the quarter period (90°) phase shift before \( W_{\text{quad}} \) allow weighing by any complex factor.

In a narrow-band array using only \( W_{\text{inphase}} \) and \( W_{\text{quad}} \) the nulls the array forms remain constant across the frequency band of interest. In terms of the modulation this means that the coherence distance in the direction of propagation should greatly exceed the length of the antenna in the same direction. Coherence means equivalent
Figure 2.3 Interelement Path Difference
values for the wavefront received at two points at different ranges from the source. If a modulated carrier is transmitted then the field shows a lack of coherence at range differences $\Gamma$

$$\Gamma > \frac{C}{BW_m}$$

where $BW_m$ is the modulation bandwidth and $c$ is the velocity of propagation. For the narrow-band case

$$\Gamma \ll \frac{C}{BW_m}$$

where the points of the receiving antenna closest and farthest from the source lie within the range interval $\Gamma$.

The circuit used for narrow-band weighting is not sufficient for wideband processing. When one is interested in receiving signals over a wide band of frequencies the 90 degree phase shifter needs to be replaced by a tapped delay line network, as the 90 degree phase shifter and two weights are insufficient. Figure 2.3 shows the wideband case. For wideband processing the weighing must be valid over a wide frequency range. The use of a transversal filter with multiple weights permits adjustment of gain and phase over the band of frequencies of interest. An important question is the determination of the necessary number of taps in the transversal filter.

For a signal to be narrow-band the bandwidth of the signal must be small in comparison to the center frequency. For radio systems this is a reasonable assumption. The desired signal for
the array is assumed to be a narrow-band CW planewave represented by

$$A_d e^{j\omega_d t}$$  \hspace{1cm} (2.2.1)

The interference is also a narrow-band CW planewave

$$A_i e^{j\omega_i t}$$  \hspace{1cm} (2.2.2)

The array elements receive Gaussian noise that is uncorrelated from element to element.

2.3 Calculation of the Covariance Matrix

In this section an analytical formula is developed following Compton [7]. This formula will be used repeatedly in this chapter and others. The received signal vector can be decomposed into three parts: the desired signal component, the interference component, and noise. The array input vector is

$$X = X_d + X_i + X_n$$  \hspace{1cm} (2.3.1)

where $X_d$ is the desired signal input vector, $X_i$ is the interference input vector, and $X_n$ is the Gaussian noise input vector. For an $N$ element antenna array the desired signal input vector is

$$X_d = [X_{d_1} \ X_{d_2} \ \ldots \ X_{d_M}]^T$$  \hspace{1cm} (2.3.2)

writing the desired input signal as

$$X_d = A_d e^{j\omega_d t} [1 \ e^{-j\delta_d} \ e^{-2j\delta_d} \ \ldots \ e^{-(N-1)j\delta_d}]^T$$

$$= A_d e^{j\omega_d t} U_d$$  \hspace{1cm} (2.3.3)

where $A_d$ is the amplitude of the desired signal, $\exp(j\omega_d t)$ is a
complex sinusoid, and $\mathbf{U}_d$ is the array desired signal phase shift vector. [7] The $\mathbf{U}_d$ vector accounts for the phase shift from array element to element.

The interference input vector is

$$X_i = [X_{i_1}, X_{i_2}, \ldots, X_{i_N}]^T$$

(2.3.4)

$$= A_d e^{j\omega_i t} [1, e^{-j\Phi_i}, e^{-2j\Phi_i}, \ldots, e^{-(N-1)j\Phi_i}]^T$$

(2.3.5)

where $A_i$ is the amplitude of the desired signal, $\exp(j\omega_i t)$ is a complex sinusoid planewave interference, and $\mathbf{U}_i$ is the array interference phase shift vector. $\mathbf{U}_i$ accounts for the interference interelement phase shift across the array.

The noise input vector is

$$X_n = [n_1(t), n_2(t), \ldots, n_N(t)]^T$$

(2.3.7)

No phase shift vector is necessary for the noise input because there is zero cross-correlation for the noise between elements. The array covariance matrix $R_{xx}$ is calculated by substituting the above expressions into the definition for $R_{xx}$.

$$R_{xx} = E[X^*X^T]$$

(2.3.8)

$$= E[\mathbf{A}_d^2 \mathbf{U}_d \mathbf{U}_d^T + A_d \mathbf{A}_d \ e^{j(\omega - \omega_d) t} \ \mathbf{U}_d \mathbf{U}_d^T + A_d \ e^{-j\omega_d t} \ \mathbf{U}_d \mathbf{X}_n^T$$

$$+ A_d \mathbf{A}_d \ e^{j(\omega - \omega) t} \ \mathbf{U}_d \mathbf{U}_d^T + A_d \mathbf{A}_d \ e^{-j\omega t} \ \mathbf{U}_d \mathbf{X}_n^T$$

$$+ A_d \ e^{j\omega t} \mathbf{X}_n^* \mathbf{U}_d^T + \mathbf{X}_n^* \mathbf{U}_d^T + \mathbf{X}_n^* \mathbf{X}_n^T]$$

$$= A_d^2 \mathbf{U}_d \mathbf{U}_d^T + A_d \mathbf{A}_d \mathbf{U}_d^T + \sigma^2 I$$
Assuming that $X_d$, $X_i$, and $X_n$ are not correlated, then the last two terms of the above equation are zero yielding

$$R_{xx} = A_d^2 U_d^T U_d + A_i^2 U_i^T U_i + \sigma^2 I \quad (2.3.8)$$

The correlated case is considered in Section 5.1. The noise is assumed uncorrelated with the other signals. Thus for narrow-band signals the input covariance matrix $R_{xx}$ can be calculated analytically using (2.3.8).

2.4 Eigenvalues of the Covariance Matrix

Due to their importance when using the LMS algorithm the eigenvalues of the covariance matrix are investigated. [7] The covariance matrix for continuous wave CW narrow-band signals is given by (2.3.8). Equation (2.3.8) is for the case where Gaussian noise with variance $\sigma^2$ is present along with a desired signal and interferer of magnitudes $A_d$ and $A_i$ respectively. The covariance matrix may be normalized by dividing $R_{xx}$ by the noise power $\sigma^2$.

$$R'_{xx} = \frac{R_{xx}}{\sigma^2} = e_d U_d^T U_d + e_i U_i^T U_i + I$$

where $e_d$ and $e_i$ are

$$e_d = \frac{A_d^2}{\sigma^2} \quad (2.4.2)$$

$$e_i = \frac{A_i^2}{\sigma^2} \quad (2.4.3)$$
The normalized covariance matrix is easier to use in calculations. The eigenvalues of the normalized covariance matrix are related to the unnormalized eigenvalues by

\[ \lambda'_i = \frac{\lambda_i}{\sigma^2} \quad \text{for} \ 1 \leq i \leq N \]  

(2.4.5)

The eigenvalue spread of the covariance matrix is the ratio of the largest to smallest eigenvalue.

\[ S = \frac{\lambda'_{\max}}{\lambda'_{\min}} \]  

(2.4.6)

The eigenvalue spread also corresponds to the time constant spread. With no interference the normalized covariance matrix becomes

\[ R'_{xx} = I + e_d U_d^T U_d \]  

(2.4.7)

The following development of the eigenvalues of \( R'_{xx} \) follows Compton [7]. Multiplying by \( U_d^* \) shows that \( U_d^* \) is an eigenvector of \( R'_{xx} \) with eigenvalue

\[ R'_{xx} U_d^* = [I + e_d U_d^T U_d] U_d^* \]

\[ = U_d^* [1 + e_d U_d^T U_d] \]

\[ = [1 + e_d U_d^T U_d] U_d^* \]

\[ = R'_{xx} U_d^* = \lambda'_1 U_d^* \]

where

\[ \lambda'_1 = 1 + e_d U_d^T U_d \]  

(2.4.8)

For the isotropic element pattern case, i.e. point sources
yielding the maximum eigenvalue

\[ \lambda_1 = \lambda_{\text{max}} = 1 + N \epsilon_d \]  

(2.4.10)

\( R'_{xx} \) is a square \( N \times N \) matrix with \( N-1 \) other eigenvalues. The other \( N-1 \) eigenvalues are all one, i.e.

\[ \lambda'_i = 1 \quad \text{for} \quad 2 \leq i \leq N \]  

(2.4.11)

The largest eigenvalue \( \lambda'_{\text{max}} \) is produced by the desired signal in the no interference case. The other \( N-1 \) unity eigenvalues are caused by the noise.

With interference terms present both desired signal and interference terms are present in the normalized covariance matrix \( R'_{xx} \). Two of the eigenvectors of this matrix must lie in the plane defined by the vectors \( U_d^* \) and \( U_i^* \), Baird et al. [37]. The combination of the two eigenvectors may be expressed as

\[ e = \alpha U_d^* + \beta U_i^* \]  

(2.4.13)

where \( \alpha \) and \( \beta \) are determined so that \( e \) is an eigenvector

\[ R'_{xx} e = \lambda' e \]  

(2.4.13)

where \( \lambda' \) is the corresponding eigenvalue. The constants \( \alpha \) and \( \beta \) are found by evaluating \( R'_{xx} e \).

\[
R'_{xx} \left[ \alpha U_d^* + \beta U_i^* \right] = \\
= \left[ I + e_i U_i^T U_i^* + e_d U_d^T U_d^* \right] \left[ \alpha U_d^* + \beta U_i^* \right]
\]

For \( e \) to be an eigenvector, terms \( \alpha \) and \( \beta \) are chosen so
The equation can be then be rewritten as

\[
= \left[ \alpha U_d^* + \beta U_1^* \right] + \alpha U_d^* \left[ e_d U_d^T U_d^* + \frac{\beta}{\alpha} e_d U_d^T \right] + \beta U_1^* \left[ e_i U_1^T U_1^* + \frac{\alpha}{\beta} e_i U_1^T U_1^* \right]
\]

\[
e_d U_d^T U_d^* + \frac{\beta}{\alpha} e_d U_d^T U_1^T = e_i U_1^T U_1^* + \frac{\alpha}{\beta} e_i U_1^T U_d^*
\]

Defining

\[
Y = \frac{\beta}{\alpha}
\]

The equation can be then be rewritten as

\[
e_d U_d^T U_1^T Y^2 - \left[ e_i U_1^T U_1^* - e_d U_d^T U_d^* \right] Y - e_i U_1^T U_d^* = 0
\]

This can then be solved by the quadratic formula yielding two solutions \(Y_1\) and \(Y_2\).

\[
Y_1 = \frac{1}{2e_d U_d^T U_1^*} [a+b]
\]

\[
Y_2 = \frac{1}{2e_d U_d^T U_1^*} [a-b]
\]

where

\[
a = e_i U_1^T U_1^* - e_d U_d^T U_d^*
\]

and

\[
b = \sqrt{a^2 + 4e_d^2 U_1^T U_1^*}^2
\]

The remaining \(N-2\) normalized eigenvalues are unity. The two non-unity eigenvalues are

\[
\lambda_1' = 1 + e_d U_d^T U_d^* + [a+b]
\]
\[ \lambda'_2 = 1 + e_d U_d^T U_d^* + [a-b] \]

The eigenvalues are all real because \( R'_{xx} \) is a Hermitian matrix. The matrix \( R'_{xx} \) is also non-negative definite because its components

\[ R'_{ss} = e_d U_d^* U_d^T \]
\[ R'_{ii} = e_i U_i^* U_i^T \]
\[ R'_{nn} = I \]

are non-negative definite and the sum of non-negative definite matrices is non-negative definite. For the normalized covariance matrix \( R'_{xx} \)

\[ \lambda'_i \geq 1 \quad \text{for } 1 \leq i \leq N \]

where the \( \lambda'_i \) are the eigenvalues of the normalized covariance matrix. The sum of the normalized eigenvalues is

\[ \sum_{i=1}^{N} \lambda'_i = N + e_d U_d^T U_d^* + e_i U_i^T U_i^* \]

The sum of the eigenvalues is equal to the total power received on all the array elements. The number of eigenvalues different from unity is equal to the number of signals incident on the array.

Next consider what happens to the eigenvalues when one interferer is present along with the desired signal. Also the magnitude of the interferer will be much greater than that of the desired signal.
\[ e_i \gg e_d \]

If for the given angle

\[ e_i U_i^T U_i^* > e_d U_d^T U_d^* > 1 \]

then the terms "a" and "b" can be approximated. Writing "b" as

\[ b = a \sqrt{1 + \frac{4e_i e_d |U_i^T U_i^*|^2}{a^2}} \]

Using the expansion

\[ \sqrt{1+x} \approx 1 + \frac{x}{2} \]

valid when \( x \) is very small, "b" may be approximated as

\[ b \approx a + \frac{2e_i e_d |U_i^T U_i^*|^2}{a} - a + 2e_d \frac{|U_i^T U_d^*|^2}{U_i^T U_i^*} \]

Substituting yields the two largest eigenvalues \( \lambda_1' \) and \( \lambda_2' \)

\[ \lambda_1' = 1 + e_i U_i^T U_i^* + e_d \frac{|U_i^T U_d^*|^2}{U_i^T U_i^*} = 1 + e_i U_i^T U_i^* \]

and

\[ \lambda_2' = 1 + e_d \left[ U_d^T U_d^* - \frac{|U_i^T U_d^*|^2}{U_i^T U_i^*} \right] \]

If the element patterns are isotropic then the two largest eigenvalues simplify to

\[ \lambda_1' = 1 + N e_i + \frac{e_d}{N} |U_i^T U_d^*|^2 \quad \text{(2.4.15)} \]

and
The sum of the eigenvalues of the normalized covariance matrix is

\[ \sum_{i=1}^{N} \lambda'_i = N [1 + e_d + e_i] \] (2.4.17)

The weight convergence behavior of the LMS algorithm is

\[ \tilde{W}(t) = \sum_{i=1}^{N} C_i e^{-\mu \lambda_i t} + R_{xx}^{-1} P_N \] (2.4.18)

The first term constitutes the transient response while the second term represents the steady state weights. The \( C_i \) form a vector which depends upon the initial condition of the weights. For convergence and stability of the LMS algorithm the following condition must hold

\[ 0 < \mu < \left[ \frac{2}{\lambda_{\text{max}}} \right] \] (2.4.19)

The eigenvalues of the covariance matrix depend on the signal and interference powers. For one narrow-band interference the maximum eigenvalue of the normalized covariance matrix is

\[ \lambda'_{\text{max}} = N e_i + 1 \] (2.4.20)

where \( N \) is the number of elements in the subarray and \( e_i \) is the interference to noise ratio

\[ e_i = \frac{A_i}{\sigma^2} \]

\( A_i \) is the amplitude of the narrow-band interferer. \( \sigma^2 \) is the
variance of the Gaussian noise at the array input. It can be seen that the value of $\mu$ for stability is limited by the maximum eigenvalue of the covariance matrix

$$0 < \mu < \frac{1}{N\epsilon_i + 1}$$

(2.4.22)

For fast convergence the transient component of the equation for $W(t)$ should die out quickly. For this to happen the following product should be as large as possible

$$\mu \lambda_i$$

for a given signal environment the $\lambda_i$'s are fixed so $\mu$ must be maximized. The time constants of the LMS algorithm are

$$\tau_i = \frac{1}{\mu \lambda_i}$$

(2.4.24)

One way to increase $\tau_i$ is to decrease $\epsilon_i$. However, given the signal environment, $\epsilon_i$ is fixed so the only remaining alternative is to decrease $N$, the size of the array. In general a smaller subarray may use a larger value of $\mu$ and will thus converge more quickly than a large array. The maximum and minimum time constants are examined.

$$\tau_{\text{max}} = \frac{1}{\mu \lambda_{\text{max}}}$$

$$\tau_{\text{min}} = \frac{1}{\mu \lambda_{\text{min}}}$$

If the maximum value of $\mu$ is used and $N\epsilon_i$ is assumed to be much larger than 1, then
\[ \mu = \frac{1}{Ne_i} \]

Substituting for \( \lambda_{\text{max}} \) and \( \mu \), \( \tau_{\text{max}} \) is

\[ \tau_{\text{max}} = Ne_i \]

Thus the maximum time constant is directly proportional to \( N \).

There is a tradeoff between the size of \( \mu \) and the misadjustment noise of the resulting output. The larger \( \mu \) becomes the larger the misadjustment noise becomes.

2.5 **Array Pattern**

To find the gain of an antenna array a unit magnitude test sinusoid \( e^{j\omega t} \) is rotated around the array and the output of the array is measured. The test signal vector is

\[ X_t = e^{j\omega t} [1 \ e^{j\phi_t(\theta)} \ ... \ e^{j(N-1)\phi_t(\theta)}]^T \]

\[ = e^{j\omega t}U_t(\theta) \quad (2.5.1) \]

where

\[ \phi_t(\theta) = 2\pi \frac{d}{\lambda} \sin(\theta) \quad (2.5.2) \]

is the interelement phase shift for the test signal input. The output of the array with the unit magnitude test signal is the gain pattern of the antenna array.

\[ g(\theta) = |W^TX_t| \]

\[ = |W^*e^{j\omega t}U_t(\theta)| \]
The $e^{j\omega t}$ can be dropped from the expression because its magnitude is one.

$$g(\theta) = |W^TU_c(\theta)| \quad (2.5.3)$$

Thus the interelement phase shift vector need only be rotated around the array. Equation (2.5.3) is valid for non-cascaded arrays.

2.6 Antenna Gain Nulls and the Degrees of Freedom for the Non-cascaded Array

As the number of elements in the subarrays is decreased the number of interferences that the subarrays may null decreases. Indeed if the number of interferences is too large the array will shut down, i.e., the output will approach zero in order to minimize the error. For this reason the number of degrees of freedom of an array is investigated in this section. First it is shown that the array pattern can be expressed as a polynomial. The output signal of a fully adapted array is

$$s(t) = W^TX$$

where the input signal vector $X$ is

$$X = e^{j\omega t} [1 \ e^{-j\phi_1} \ e^{-j\phi_2} \ ... \ e^{-j(N-1)\phi_1}]$$

and $W$ is the weight vector for the array.

$$W = [w_1 \ w_2 \ w_3 \ ... \ w_4]^T$$

this yields an array output signal $s(t)$
(2.6.1) can be factored as

\[ S(t) = e^{j\omega t} \left[ w_1 + w_2 e^{-j\Phi_t} + \ldots + w_N e^{-j(N-1)\Phi_t} \right] \]  

The term in brackets is the array gain pattern. Since there are \( N-1 \) coefficients in the bracketed expression there are \( N-1 \) degrees of freedom in the gain expression for an \( N \) element array.

Dropping the \( e^{j\omega t} \) from (2.6.1) and making the following substitution

\[ z = e^{j\Phi_t} = e^{j(2\pi \frac{d}{\lambda} \sin(\theta))} \]  

the gain pattern for an \( N \)-element linear array can be expressed as a \((N-1)\)th order polynomial. The antenna gain polynomial is

\[ g(z) = w_1 + w_2 z + w_3 z^2 + \ldots + w_N z^{N-1} \]

\[ = \sum_{k=1}^{N} w_k z^{k-1} \]  

\( \Phi \) is the interelement phase shift of a test input signal of magnitude one and \( d \) is the interelement spacing. The coefficients of the polynomial are complex and represent the array weight magnitude and phase.

\[ w_k = |w_k| e^{j\Phi_k} \]

The gain polynomial (2.6.1) has \( N-1 \) roots and can be factored as here the \( b_k \)'s are complex zeros of the pattern polynomial. \( z \) is a
\[ g(z) = w_n(z-b_1)(z-b_2)(z-b_3)\ldots(z-b_{N-1}) \]

phasor with magnitude 1. The roots of the polynomial on the unit circle represent zeros of the antenna pattern.

When \( d > \lambda/2 \) the zeros of the pattern may repeat over the 180 degrees of real angles and if \( d < \lambda/2 \) there may be zeros of the pattern outside the range of real angles. The maximum number of zeros is \( N-1 \) but there may be no true zeros if all the roots are located off the unit circle. In a non-ideal array the zeros are made less deep because of the effects of mutual coupling and weight tolerance. [40]

For a particular interference location a weight (zero) may be found to place a zero at the interference angle. In this case the pattern may be factored as

\[ g(z) = (z-b_j)g_1(z) \]

Figure 2.4 shows a Davies array that can be factored this way.
Figure 2.4 Davies' Nulling Tree
The basis of the Davies [37] array is the independent steering of antenna pattern zeros with a nulling tree. Where subarrays are used the number of degrees of freedom of the subarrays is smaller than that of the entire array. When the number of signals exceeds the number of degrees of freedom of the array the adaptive array is overcome by the number of interferences. In this case the weights of an LMS array go to zero.

2.7 Frost Beamformer

Due to the limit on the number of interferers established in Section 2.6 a cure called the Frost beamformer is investigated. The Frost beamformer constrains the gain in the direction of the desired signal and minimizes the output power of the array. The desired signal direction must be known apriori.

When the number of degrees of freedom is overcome in order to avoid the weights going to zero as in the LMS array case, the Frost beamformer constrains the gain to be one in the desired signal direction. For the Frost beamformer the steady state weights are

\[ w_{frost} = \frac{R_{xx}^{-1}U_d^*}{U_d^*R_{xx}^{-1}U_d} \]

\[ = kR_{xx}^{-1}U_d^* \quad (2.7.2) \]

where
\[
k = \frac{1}{\mathbf{U}_d^\top \mathbf{R}_{xx}^{-1} \mathbf{U}_d^*}
\]

is a scalar because \( \mathbf{U}_d^\top \) is \((1 \times N)\), \( \mathbf{R}_{xx}^{-1} \) is \((N \times N)\), and \( \mathbf{U}_d^* \) is \((N \times 1)\). The antenna gain for the Frost beamformer is

\[
g(\theta) = |k \mathbf{R}_{xx}^{-1} \mathbf{U}_d^\top \mathbf{U}_d(\theta)|
\]

\[
= |k \mathbf{R}_{xx}^{-1} \mathbf{U}_d^\top \mathbf{U}_d(\theta)|
\]

\[
= |k \mathbf{W}^\top \mathbf{U}_d(\theta)|
\]

which is the same pattern as a non-Frost array multiplied by the constant \( k \). The pattern shape does not change only its amplitude is altered by the Frost constraint. The weights for a Frost beamformer [37] can be calculated using

\[
\mathbf{W}_{\text{frost}} = \mathbf{R}_{xx}^{-1} \mathbf{U}_d^\top \left[ \mathbf{U}_d^\top \mathbf{R}_{xx}^{-1} \mathbf{U}_d \right]^{-1}
\]

(2.7.4)

2.8 Formula for Array Output SINR

A formula is established for the SINR of the array. [7] This formula is then used in the development of grating nulls. The formula is also used again in Section 3.8 to establish an expression for the SINR of a cascade array. If the output of the array \( s(t) \) is \( \mathbf{W}^\top \mathbf{x} \) and the array input vector \( \mathbf{x} \), has three components: the desired signal, the interference, and the noise, then each component of the input vector may be considered separately in order to find the signal to interference and noise.
ratio (SINR). For the desired signal

\[ P_d = E[|W^T X_d|^2] \]

\[ = A_d^2 |W^T U_d|^2 \]

for the interference power

\[ P_i = E[|W^T U_i|^2] \]

for the noise power

\[ P_n = E[|W^T X_n|^2] \]

\[ = \frac{\sigma^2}{2} W^H W \]

The SINR is just

\[ SINR = \frac{P_d}{P_i + P_n} \] \hspace{1cm} (2.8.1)

The SINR can be found in terms of the weights for reference and comparison purposes.

\[ P_d = A_d^2 W^T U_d [W^T U_d]^H \]

\[ = A_d^2 W^T U_d U_d^H W^* \]

\[ = W^T R_{ss} W^* \] \hspace{1cm} (2.8.2)

for the interference

\[ P_i = W^T R_{ii} W^* \] \hspace{1cm} (2.8.3)

for the noise

\[ P_n = W^T R_{nn} W^* \]
The SINR in terms of the weights is

\[ \text{SINR} = \frac{W^T R_{ss} W^*}{W^T [R_{ii} + \sigma^2 I] W^*} \]

Using the matrix inversion lemma [7]

\[ Q = B - \beta Z^* Z^T \]

\[ Q^{-1} = B^{-1} - \tau B^{-1} Z^* Z^T B^{-1} \]

where \( B \) is a nonsingular \( N \times N \) matrix, \( Z \) is an \( N \times 1 \) column vector, \( \beta \) is a scalar, and \( \tau \) and \( \beta \) are related by

\[ \tau^{-1} + \beta^{-1} = Z^T B^{-1} Z^* \]

may be used twice to find the optimum array weights as a function of the phase shift vectors, the signal to noise ratio, and the interference to noise ratio. This development follows Compton [7]. If only the thermal noise and interference components of the covariance matrix are considered then

\[ \left[ \sigma^2 I + A_i^2 U_i^* U_i^T \right]^{-1} \]

\[ = \frac{1}{\sigma^2} \left[ I - \frac{U_i^* U_i^T}{e + U_i^T U_i^*} \right] \]

If now the lemma is used one more time with the substitutions

\[ B = \sigma^2 I + A_i^2 U_i^* U_i^T \]

and
\[ Z = U_d, \quad \beta = -A_d^2 \]

then the inverse of the normalized covariance matrix is

\[
[R_{xx}]^{-1} = \frac{1}{\sigma^2} I + \frac{\tau}{\sigma^4} U_d^T U_d
\]

\[
- \frac{1}{\sigma^2 + U_d^T U_d} \left[ \frac{1}{\sigma^2 + U_i^T U_i} \right] U_i^T U_i
\]

\[
+ \frac{\tau}{\sigma^4} \frac{U_i^T U_i^*}{\sigma^2 + U_i^T U_i} U_d^T U_d + \frac{\tau}{\sigma^4} \frac{U_i^T U_i^*}{\sigma^2 + U_i^T U_i} U_i^T U_i
\]

where

\[
\tau^{-1} = \frac{1}{A_d^2} \frac{1}{\sigma^2 U_i^T (U_i^T U_i) (U_i^T U_i^*)}
\]

or taking the inverse

\[
\tau = A_d^2 \left[ 1 + \frac{\gamma}{\sigma^2} \right]^{-1}
\]

where

\[
\gamma = A_d^2 \left[ U_i^T U_i^* - \frac{U_i^T U_i^* (U_i^T U_i)}{\sigma^2 + U_i^T U_i} \right]
\]

The optimum weights are now found using (1.1.7)

\[
W_{opt} = R_{xx}^{-1} P = R_{xx}^{-1} A_d U_d^*
\]

\[
= A_d^2 \left[ 1 - \frac{\tau}{\sigma^2} \frac{\gamma}{A_d^2} \left[ U_i^T U_i^* - \frac{U_i^T U_i^*}{\sigma^2 + U_i^T U_i} U_i^* \right] \right]
\]
the output power of the desired signal is
\[ P_d = \frac{A_d^2}{2} |w^T u_d|^2 = \frac{1}{2} \left[ \frac{\gamma}{\sigma^2 + \gamma} \right]^2 \]

the output power of the interference is
\[ P_i = \frac{A_i^2}{2} |w^T u_i|^2 \]

\[ = \frac{A_d^2 A_i^2}{2} \left[ \frac{1}{\sigma^2 + \gamma} \right]^2 \left[ \frac{e_i^{-1}}{e_i^{-1} + U_i^T U_i} \right]^2 |U_i^T u_d|^2 \]

the output power of the noise is
\[ P_n = \frac{A_d^2 A_i^2}{2} \left[ \frac{1}{\sigma^2 + \gamma} \right]^2 \left[ \frac{e_i^{-1}}{A_d^2} - \left| U_i^T u_d \right|^2 \left[ \frac{e_i^{-1}}{e_i^{-1} + U_i^T U_i} \right] \right]^2 \]

Substituting the equations for \( P_d, P_i, \) and \( P_d \) into (2.8.1) and simplifying yields
\[ SINR = e_d \left[ U_d^T u_d^* - \frac{U_i^T U_i^*}{e_i^{-1} + U_i^T U_i} \right] \]

(2.8.6)

2.9 Grating Nulls and Subarray Spacing

In order to show that the inter-subarray phase center spacing should not be greater than \( \lambda/2 \), grating nulls are investigated in this section. It is important to consider the effect of inter-element spacing \( d \) on array performance, because if spatial sampling is insufficient grating nulls can occur. Grating null problems
occur when the interference and desired signal produce signals at the array that have the same interelement amplitude and phase shifts. The interference and desired signal are then indistinguishable at the array elements, even though the desired signal and interference may have come from different directions.

To examine the effect of array spacing on grating nulls the SINR of the array is written as a function of interelement spacing and its behavior as a function of the array spacing is examined. The desired signal and interference input vectors are:

\[ X_d = A_d e^{j\omega c} \left[ 1 \quad e^{-j\phi_d} \ldots \quad e^{-j(N-1)\phi_d} \right]^T \]

\[ X_i = A_i e^{j\omega c} \left[ 1 \quad e^{-j\phi_i} \ldots \quad e^{-j(N-1)\phi_i} \right]^T \]

The array interelement phase shifts for the desired signal and interference are:

\[ \phi_d = \frac{2\pi d}{\lambda} \sin(\theta_d) \]

and

\[ \phi_i = \frac{2\pi d}{\lambda} \sin(\theta_i) \]

d is the interelement spacing. For the numerator of the second term of (2.8.6)

\[ U_d^T U_d^* = U_i^T U_i^* = N \]

Substituting, the SINR of the array is
\[ \text{SINR} = \varepsilon_d \left[ N - \frac{|U_d^T U_i^*|^2}{\varepsilon_i^{-1} + N} \right] \]

In the numerator of the second term of this expression, the product \( u_d^T u_i^* \) is expanded as:

\[ U_d^T U_i^* = 1 + e^{j(\phi_i - \phi_d)} + e^{2j(\phi_i - \phi_d)} + \ldots \]

If the simplest case, a two element array, is used as an example then the two element array products are:

\[ U_d^T U_i^* = 1 + e^{j(\phi_i - \phi_d)} \]

\[ |U_d^T U_i^*|^2 = 4 \cos^2 \left( \frac{\phi_i - \phi_d}{2} \right) \]

The SINR for the two element array is then:

\[ \text{SINR} = \varepsilon_d \left[ 2 - \frac{4 \cos^2 \left( \frac{\phi_i - \phi_d}{2} \right)}{\varepsilon_i^{-1} + 1} \right] \]

The SINR will drop whenever the signal vectors of the interference and desired signal are parallel, i.e., linearly dependent.

\[ U_d = K U_i \]

Setting corresponding terms in the desired signal and interference phase shift vectors equal implies:

\[ \frac{2\pi d}{\lambda} \sin(\theta_d) = \frac{2\pi d}{\lambda} \sin(\theta_i) - 2\pi n \quad \text{(2.9.1)} \]

where \( n \) is an integer. The \( n = 0 \) solution always exist and if \( d/\lambda \) is large enough then solutions for other values of \( n \) may exist.
also. These other solutions for n not equal to 0 are the grating null solutions. For these cases the SINR drops to a low value and the desired signal is indistinguishable from the interference. For a grating null there are additional undesired nulls in the beam pattern. If these undesired nulls lie in the direction of the desired signal then the array fails. The significance of grating nulls is that subarray phase center spacing must be less than $\lambda/2$.

2.10 Summary

This chapter has provided background on the explicit calculation of the optimum weights. An explicit formula for array gain has been established. The fact that the eigenvalues are proportional to the array size has been shown. The direct proportionality of the largest eigenvalue to array size will set the convergence speed of the LMS arrays in Chapter 6. Array dimensionality has been discussed due to the limit on the number of interferences that the array can cope with. If the number of degrees of freedom is overcome a Frost constraint can be used. An explicit formula for the SINR was developed for use in succeeding chapters. The SINR formula will be used for performance comparisons. The subject of grating nulls appears again in the next chapter. Chapter 3 discusses several examples of optimum cascade arrays using the Wiener-Hopf equation to solve for the optimum weights. The optimum SINR and null depth are also found.
Chapter 3

Optimum Cascade Arrays

3.0 Background

In this chapter the general cascade configuration and its notation is discussed. The notation is the author's. A cascade gain formula is developed and used in the examples. In Section 3.3 it is shown that it is necessary to solve for only one subarray per stage. The weights of the subarray computed at each stage may then be copied to the other subarrays at that stage. This greatly lowers computational complexity. The optimum cascade weight values are found using the Wiener-Hopf equation. Thus non-cascade structures may be compared to cascade structures at convergence. As discussed in Chapter 1, there are numerous reasons to cascade arrays. A time division multiplexed (TDM) processor unit is possible because all the arrays at the same layer have the same optimum weight solution. Smaller subarrays yield processors of lower complexity and these smaller signal processors are less costly. Also there are advantages of using common hardware. The processor may serve several purposes (algorithms) simultaneously and be time division multiplexed. The eigenvalue spread is less for smaller arrays, as shown in Section 2.4. When gradient algorithms are used the step size $\mu$ can be larger with corresponding convergence rate improvement, as shown in Section 2.4. When smaller arrays are used they will converge faster than a large array when using
the least mean square (LMS) technique. Smaller word size is necessary to accommodate lowered eigenvalue spread in $R_{xx}$.

If the sample matrix inversion technique is used estimation delay decreases over that of a larger array as shown in Section 4.2. One of the problems with the SMI technique is that it is essentially open-loop. Through cascading sensitivity to weight errors is decreased because succeeding stages can compensate for previous stage weight errors.

The reliability of the cascade is enhanced through multiple smaller signal processors. If a single processor fails another of a pool of processors may take its place in the array or its output may be suppressed by succeeding signal processors.

3.1 Cascade Arrays Configuration

The general cascade configuration is discussed in this section. The inputs to an antenna array may be partitioned into a set of subarrays. Each of the subarrays is then adapted and their outputs are considered as a set of inputs to the next layer of subarrays. The process is then continued until there is only a single final subarray. Figure 3.1 portrays this general cascade array case. Each of the subgroups is treated as one input to the next stage of adaptive combiners. The first group of subarrays implements transformation $G_1$, the second group of subarrays implements
Figure 3.1 General Cascade Array
transformation $G_z$, and the last group implements $G_n$. Using this approach one small signal processor can be time division multiplexed between all the subarrays. Also the computational complexity is lowered. When the initial subarrays have the same structure (interelement phase shift) then their input covariance matrices are equal as proven in Section 3.2.1. The received signal at the subarrays in a cascade stage differs only in phase from the initial subarray in the layer. The cross-correlation vectors are also equal, thus all subarrays have the same optimum weight solution. It is not necessary to adapt all of the subarrays individually but merely to adapt over one subarray and copy its weight solution to the other subarrays. This has been observed by simulation of a linear 9 element array arranged as three subarrays of three elements each. Subarrays are the basis for another technique called spatial dithering. Spatial dithering is the electronic jiggling (or movement) of the antenna array in order to modulate coherent interference. This allows the array to suppress coherent interference, which would otherwise not be possible with a conventional array.

3.2 Partially Adaptive Arrays

Partial adaptivity and cascade arrays have much in common. A cascade array that reduces computational complexity is necessarily a partially adaptive array. Partial adaptivity is useful when the number of inputs is very large. In
partially adaptive arrays the adaptive processor adapts over only a subset of the total number of array elements. Subarrays occur in partially adaptive arrays. A fully adaptive array offers the most flexibility, however, a partially adaptive array may be the only technological alternative as the number of array elements increases. If the number of array elements is large, the signal processor may not be able to handle the large number of inputs. Note that the use of subarrays spanning all the array elements is a technique that can result in a fully adaptive array even when the number of elements may number in the thousands. Breaking a very large array up into subarrays is a natural way to handle very large arrays. Each of the subarrays may then use an identical signal processor or a single time division multiplexed (TDM) processor. Algorithms with higher complexity, $O(N^2)$ and $O(N^3)$, can be used with smaller arrays. Non-gradient, greater than $O(N)$ complexity, algorithms may be used because higher complexity algorithms are possible in real-time with fewer inputs.

3.3 Subarray Covariance Matrices

Notation is developed here to calculate the examples of this chapter and chapter 4. Subarrays partition the total input covariance matrix $R_{xx}$ into submatrices. As an example consider a three element array (Figure 3.2) treated as three subarrays. The total input covariance matrix is
If $R_{xx}$ is real then it is symmetric and if complex notation is used then the $R_{xx}$ matrix is Hermitian.

$$R_{xx} = (R_{xx}^*)^T = R_{xx}^H$$

Since $R_{xx}$ is symmetric only the diagonal and that portion of $R_{xx}$ either above or below the diagonal must be known to uniquely specify the $R_{xx}$ matrix. If subarray 1 consists of antenna elements 1 and 2 then the input covariance matrix for subarray 1 can be formulated in a new way as

$$R_{xx1} = Q_1 R_{xx} Q_1^T$$ (3.3.1)

where

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

The $Q_1$ matrix picks those elements out of $R_{xx}$ that form the subarray input covariance matrix $R_{xx1}$. The correlation vector for subarray 1, $P_1$, is

$$P_1 = Q_1 U_d^T$$ (3.3.2)

$$W_{\text{opt}} = R_{xx}^{-1} P_1$$ (3.3.3)

$$= (Q_1 R_{xx} Q_1^T)^{-1} Q_1 U_d^*$$

If subarray 2 consists of array elements 1 and 3 then the input covariance matrix for subarray 2 is
The input covariance matrix for subarray 3 consisting of array elements 2 and 3 is

\[ R_{xx3} = Q_3 R_{xx} Q_3^T \]

where

\[ Q_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

and the correlation vector for subarray 3 is

\[ P_3 = Q_3 U_d^* \]

The \( Q \) matrices select the proper partition of the total input covariance matrix \( R_{xx} \). In this case the partitions cover all of \( R_{xx} \), however in general this need not be the case. For sub-optimal performance some entries in \( R_{xx} \) may not be covered.

### 3.3.1 Equivalent Subarrays - Equal Covariance Matrices

Only one subarray per level of the cascade must be adapted. The other subarrays at its level may use a copy of its weights. In order to show the above statement a linear antenna array is given with two equivalent subarrays. The
subarrays are equivalent in the sense that they have the same interelement phase shifts. The desired signal steering vector for the subarray 1 is

\[ U_{d1} = [1, e^{-j\phi_1}, \ldots, e^{-j(N-1)\phi_1}]^T \]  \hspace{1cm} (3.3.1.1)

The desired signal steering vector for subarray 2 is

\[ U_{d2} = e^{-jr} [1, e^{-j\phi_2}, \ldots, e^{-j(N-1)\phi_2}]^T \] \hspace{1cm} (3.3.1.2)

\[ = e^{-jr} U_{d1} \]

where \( r \) is the phase delay between the first element of subarray 1 and 2. When only the signal and Gaussian noise are present the input covariance matrix for the first subarray is

\[ R_{xx1} = A_d^2 U_{d1} U_{d1}^T + \sigma^2 I \]  \hspace{1cm} (3.3.1.3)

For the second subarray the input covariance matrix is

\[ R_{xx2} = A_d^2 U_{d2} U_{d2}^T + \sigma^2 I \]  \hspace{1cm} (3.3.1.4)

substituting (3.3.1.2) into (3.3.1.4) yields

\[ = A_d^2 e^{jr} U_{d1}^T e^{-jr} U_{d1} + \sigma^2 I \]  \hspace{1cm} (3.3.1.5)

\[ = A_d^2 U_{d1} U_{d1}^T + \sigma^2 I \]

\[ = R_{xx1} \]  \hspace{1cm} (3.3.1.6)

Thus the covariance matrices of the two subarrays are equal. Since the desired signal at the first element of each subarray is used as the reference.

\[ P_1 = U_{d1}^* = P_2 \]

So both subarrays 1 and 2 have the same optimal weight
solution.

3.4 Cascade Preprocessing

Cascade array preprocessors are used to lower the eigenvalue spread of the covariance matrix [14]. Indeed, this is exactly what a cascade array does. This is the goal of a preprocessor for gradient methods because gradient methods converge slowly for wide eigenvalue spread. Adaptive techniques based on the gradient exhibit slow convergence because of conflicting requirements of fast adaptation rate and small misadjustment noise. Fast adaptation rate requires a large step size, whereas small misadjustment error requires a small step size. Gradient algorithms convergence slowly when the eigenvalues of the covariance matrix have a large spread. Several examples of preprocessors occur in the recent literature. An advantage of the author's general cascade is that the eigenvalues are reduced at the second stage and a larger $\mu$ may be used at the second stage while still maintaining the same misadjustment noise as the first stage.

3.5 Cascade Gain

The overall effect of a two stage cascade array can be represented by two transformations. First there is the $(N \times M)$ subarray input transformation $G$, which is followed by the $(M \times 1)$ final combiner transformation. If $N$ inputs are represented as
These inputs are transformed by the subarray transformation to M subarray outputs. This is represented by the equation

\[ Y = G_1^T X \] (3.5.2)

where

\[ Y = [y_1 \ y_2 \ \ldots \ y_M]^T \] (3.5.3)

\( Y \) is a vector of the initial subarray outputs. The M subarray outputs are then transformed by the final combiner to produce the final array output.

\[ s(t) = W_{yopc}^T Y = G_2^T Y \] (3.5.4)

Substituting for \( Y \)

\[ s(t) = G_2^T G_1^T X \] (3.5.5)

The gain of the cascade array is

\[ g_c(\theta) = |U_c^T(\theta) G_1 G_2| \] (3.5.6)

where \( U_c(\theta) \) is the phase shift vector for a unit magnitude test input signal. \( G_1 \) is the subarray transformation and \( G_2 \) is a matrix of the second stage cascade array weights. If instead of a two stage network the cascade consists of a three stage network with third stage transformation \( G_3 \), then the gain can be written

\[ g_c(\theta) = |U_c^T(\theta) G_1 G_2 G_3| \] (3.5.6)
3.6 Subarrays with Optimal Performance

It is shown here that when the input subarrays perform an invertible transformation $G_i$ of the inputs $X$ then the performance of a cascaded array is the same as that of a non-cascaded array. A proof follows for a two stage cascade array. Consider two cases: 1) a 3 element array that is combined as a fully adapted 3 element array and 2) a 3 element array that is combined as 3 subarrays of 2 elements the outputs of which are adaptively combined as a 3 element array (Figure 3.2). For the cascade case in the Figure 3.2 the outputs of the subarrays can be written as:

$$y_1 = w_{11}x_1 + w_{21}x_2 = G_{11}^TX$$

$$y_2 = w_{21}x_2 + w_{22}x_3 = G_{12}^TX$$

$$y_3 = w_{31}x_1 + w_{32}x_3 = G_{13}^T$$

where

$$G_{11} = [w_{11} w_{12} 0]^T$$

$$G_{12} = [0 w_{21} w_{22}]^T$$

$$G_{13} = [w_{31} 0 w_{32}]^T$$

and

$$X = [x_1 x_2 x_3]^T$$
x's are array inputs
µ's are array weights

Figure 3.2 Cascade Array
or in matrix notation:

\[
\begin{bmatrix}
Y_1 \\
Y_2 \\
Y_3
\end{bmatrix} =
\begin{bmatrix}
w_{11} & w_{12} & 0 \\
0 & w_{21} & w_{22} \\
w_{31} & 0 & w_{32}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
\]

(3.6.4)

The $G_i$'s and $Y$'s may be grouped as:

\[G_1 = \begin{bmatrix} G_{11} & G_{12} & G_{13} \end{bmatrix}^T\]

(3.6.5)

and then $Y$, the vector of initial subarray outputs, can be written

\[Y = G_1^T X\]

The solution for the optimum weights $W_{\text{opt}}$ of the second stage cascade array with $Y$ as the input is

\[W_{\text{opt}} = R_{yy}^{-1} P_y\]

(3.6.7)

where $R_{yy}$ is the covariance matrix at the second stage

\[R_{yy} = E[Y Y^\top]\]

(3.6.8)

and $P_y$ is the cross correlation at the cascade stage

\[P_y = E[Y d]\]

(3.6.9)

where $d$ is the reference signal. The output of the cascade array with $Y$ as input is

\[s(t) = Y^\top W = Y^\top R_{yy}^{-1} P_y = G_2^T Y\]

(3.6.10)

Substituting $Y = G_1^T X$ into the equation for $R_{yy}$, the input covariance matrix for second stage of the cascade is have

\[R_{yy} = E[(G_1^T X)^\top (G_1^T X)^\top]\]

since the $G_i$'s are constant matrices they may be taken outside
\[ = E[G_1^H X^* X^T G_1] \] 

(3.6.11)

of the expectation operator

\[ R_{yy} = G_1^H E[X^* X^T] G_1 \]

\[ = G_1^H R_{xx} G_1 \] 

(3.6.12)

Hence the input covariance matrix for the cascade stage is the input covariance matrix \( R_{xx} \) of the first stage transformed by \( G_1 \), the input subarray transformation. The cross correlation vector with the desired signal at the cascade stage is

\[ P_y = E[Y^*d] \] 

(3.6.13)

substituting for \( Y \)

\[ P_y = E[(G_1^H X^*)d] \]

removing \( G_1 \) from the expectation

\[ P_y = G_1 E[X^*d] \] 

(3.6.15)

\[ = G_1^H P_x \]

So the cross correlation of the second stage is the same as that of the first stage transformed by the \( G_1^H \) vector. The final array output signal is

\[ s(t) = Y^T G_2 \] 

(3.6.16)

substituting for the second stage weights \( G_2 \) in (3.6.16)

\[ s(t) = Y^T R_{yy}^{-1} P_y \] 

(3.6.17)

substituting for \( Y \)
Now using the matrix identity

\[ [A \ B \ C]^{-1} = C^{-1} B^{-1} A^{-1} \]

to simplify

\[ \begin{bmatrix} G_i^H R_{xx} G_i \end{bmatrix}^{-1} = G_i^{-1} R_{xx}^{-1} (G_i^H)^{-1} \]  \hspace{1cm} (3.6.19)

Rearranging (3.6.18) the expression for the output signal is

\[ s(t) = X^T \left[ G_i^* G_i^{-1} R_{xx}^{-1} \left[ G_i^H \right]^{-1} G_i^H P_x \right] \]  \hspace{1cm} (3.6.20)

the \( G_i \) transformation cancels in (3.6.20) and the array output is

\[ = X^T R_{xx}^{-1} P_x \]  \hspace{1cm} (3.6.21)

With a gain of

\[ g(\theta) = | \left[ R_{xx}^{-1} P_x \right]^T U_c(\theta) | \]

\[ = | W^T U_c(\theta) | \]

This is the same gain as a fully adapted (non-cascade) array with inputs \( x_1, x_2, \) and \( x_3 \) proving that the performance of the cascaded array is the same as that of the fully adapted array. Thus we may expect the SINR performance of a cascaded array with invertible subarray transformation \( G_i \) to be the same as that for a fully adapted array. The simplification that makes the performance the same is valid only if the subarray transformation matrix \( G_i \) is invertible. For \( G_i \) to be
invertible it is necessary for the number of subarrays to equal the number of array elements. For the purposes of computational reduction this is not particularly interesting because it means that the number of inputs to be dealt with is not decreased by the $G_1$ transformation. For the case where the transformation $G_1$ is non-invertible the simplification can not be done and the formula for the output remains

$$s(t) = X^T G_1 [G_1^H R_{xx} G_1]^{-1} G_1^H P_x$$  (3.6.22)

and the output signal can be written as

$$s(t) = X^T G_1 R_{yy}^{-1} P_y$$

$$= G_2^T G_1^T X$$  (3.6.23)

For this case the cascade array antenna pattern is the magnitude of $s(t)$ as a test signal of magnitude one is rotated around the array. The antenna pattern is

$$g_c(\theta) = | U_c^T(\theta) G_1 R_{yy}^{-1} P_y |$$

$$= | U_c^T(\theta) G_1 G_2 |$$  (3.6.24)

In this case the performance of the cascade array will not be the same as that of a fully adapted array.

A necessary condition for the subarray transformation $G_1$ to be invertible is that the transformation matrix $G_1$ be square. In terms of the dimensions of the matrix $G_1$ this means that the number of subarrays must equal the total number of elements in the overall array. This is not a very useful case because the number of inputs at the second stage will be
the same as the number of input at the first stage. No dimensionality reduction takes place. However, small subarrays may be used at the first stage just for eigenvalue reduction. These outputs may then be fed to a cascade fully adaptive array. In this case the array will have optimal performance and will converge faster than a non-cascaded array.

3.7 Suboptimal Performance with Reduced Computational Complexity

If the number of subarrays in each transformation stage is equal to the number of elements in the array then this implies that at the next stage there has been no reduction in the number of inputs. Essentially one array is being replaced by another array of the same size. No computational complexity reduction is taking place, although the eigenvalue spread in the next stage will be reduced. In order to reduce the complexity of computation there should be less subarrays than there are total elements in the array. This results in fewer inputs to the next stage. A tree type inverted triangle structure results as seen in Section 3.9. However, when there are fewer subarrays than there are elements in the array the G transformation will not be invertible and performance equivalent to a fully adapted array can not be expected, although, the performance of the partially adapted array may closely approximate that of a fully adapted array. The
dimensions of the transformation matrix $G_1$ are

\[(\text{no. rows}) \times (\text{no. columns}) = (\text{no. of subarrays}) \times (\text{no. of elements})\]

where the number of elements is the total number of array elements.

Consider the case of a 3 element array where only two subarrays are used. In this case the dimensions of $G_1$ are 2x3 and the transformation $G_1$ is non-invertible. The performance can not be expected to be the same as that of a fully adapted array. In this case the columns of the $G_1$ matrix, $G_{11}$ and $G_{12}$ are

\[G_{12} = \begin{bmatrix} w_{11} & w_{12} & 0 \end{bmatrix}^T\]

\[G_{12} = \begin{bmatrix} 0 & w_{11} & w_{12} \end{bmatrix}^T\]

The first stage subarray transformation matrix $G_1$ is

\[G_1 = \begin{bmatrix} w_{11} & 0 \\ w_{12} & w_{11} \\ 0 & w_{12} \end{bmatrix}\]

The second stage transformation matrix is

\[G_2 = \begin{bmatrix} w_{21} & w_{22} \end{bmatrix}\]

3.8 Cascade SINR

To evaluate the cascade SINR consider an $N$ element antenna array divided into $K$ subarrays of $M$ elements each. For one interferer only the SINR of the subarrays is
Since the front adaptive subarrays use the same weight values they all have the same SINR. If the array is operating at a level of high interference, then the interference will be deeply nulled by the subarrays. In this case uncorrelated thermal noise will be the dominant undesired component of the subarray outputs. If the outputs of the subarrays are conventionally combined then the output SINR of a single subarray is multiplied by \( K \), the number of subarrays.

\[
SINR_{\text{subarray}} = \varepsilon_d \left[ M - \frac{|U_d^T U_i^\ast|^2}{\varepsilon_i^{-1} + U_i^T U_i^\ast} \right] \tag{3.8.1}
\]

\[
SINR_{\text{conventional}} = K \varepsilon_d \left[ N - \frac{K |U_d^T U_i^\ast|^2}{\varepsilon_i^{-1} + M} \right]
\]

\[
= \varepsilon_d \left[ N - K \frac{|U_d^T U_i^\ast|^2}{\varepsilon_i^{-1} + M} \right] \tag{3.8.2}
\]

If \( |U_d^T U_i^\ast| \) is small then the performance of the conventionally combined adaptive subarrays approaches that of a fully adapted array. Small values of \( |U_d^T U_i^\ast| \) denote orthogonality of the \( U_d \) and \( U_i^\ast \) phase shift vectors. The largest value of the product occurs when \( U_d \) and \( U_i^\ast \) are linearly related, i.e.,

\[ U_d = K U_i^\ast \]

for \( K = 1 \)

\[ |U_d^T U_i^\ast|^2 = M^2 \]

and
Now consider adaptively combining the front subarrays. The amplitude of the uncorrelated noise out of the subarrays is increased \( K \) times. The power of the noise is increased \( K \) times. The desired signal adds in phase so the amplitude is multiplied by \( K|g(\theta_d)| \). \( g(\theta_d) \) is the gain of the subarrays in the direction of the desired signal. The desired signal power is increased by \( K^2|g(\theta_d)|^2 \). The interference power is decreased by \( K^2|g(\theta_i)|^2 \) where \( g(\theta_i) \) is the gain of the subarrays in the direction of the interference. The desired signal power is increased \( K^2 \) times. For \( K \) element subarrays of \( M \) elements each

\[
SINR_{\text{conventional}} = N e_d \left[ \frac{M}{e_d^2 + M} \right]
\]  

(3.8.3)

\[
SINR_{\text{cas}} = \frac{K^2 P_d}{K P_n + K^2 P_i}
\]  

(3.8.4)

\[
= \frac{K^2 e_d |g(\theta_d)|^2}{K P_n + K^2 e_i |g(\theta_i)|^2}
\]

\[
= \frac{e_d |g(\theta_d)|^2}{\frac{P_n}{K} + e_i |g(\theta_i)|^2}
\]

where \( P_d, P_n, \) and \( P_i \) are the desired signal, noise, and interference powers out of a subarray. \( P_n, P_i, \) and \( P_d \) can be found using (2.8.4), (2.8.3), and (2.8.2) respectively. For noise with variance one the noise power is

\[
P_n = \mathbf{W}^T \mathbf{W^*}
\]  

(3.8.5)
For this development the $U_d$, $U_i$, and $W_{opt}$ vectors are for a subarray. For one interference the subarray weights are [28]

$$W_{opt} = \left[ I - \frac{e_i U_i^T U_i^T}{1+e_i U_i^T U_i^T} \right] U_d^* \quad (3.8.6)$$

substituting (3.8.6) into (3.8.5) results in

$$P_n = M + \left[ U_d^T U_i^T U_i^T U_d^T \right] \left[ \frac{e_i^2}{1+e_i M^2} - \frac{2e_i}{1+e_i M} \right] \quad (3.8.7)$$

let

$$\rho = U_d^T U_i^T U_i^T U_d^T$$

$\rho$ is a real number. For $M = 2$

$$\rho = 2[1+\cos(\phi_i+\phi_d)]$$

For $M = 3$

$$\rho = 3 + 2\cos[2(\phi_i+\phi_d)] + 3\cos(\phi_i+\phi_d)$$

For $P_i$

$$P_i = e_i |g(\theta_i)|^2$$

where

$$g(\theta_i) = |U_i^T W| \quad (3.8.8)$$

Substituting (3.8.6) into (3.8.8) and simplifying yields

$$P_i = e_i \left| \frac{U_i^T U_d^*}{1+\rho e_i} \right|^2 \quad (3.8.9)$$

For the desired signal
where

\[ P_d = e_d | g(\theta_d) |^2 \] \hspace{1cm} (3.8.10)

\[ g(\theta_d) = |U_d^T W| \] \hspace{1cm} (3.8.11)

\[ = |U_d^T U_d^* - \frac{e_i U_d^T U_i U_i^T U_d^*}{1 + e_i U_i^T U_i^*}| \]

\[ = |M - \frac{e_i U_d^T U_i U_i^T U_d^*}{1 + M e_i}| \] \hspace{1cm} (3.8.12)

\[ |g(\theta_d) |^2 = \frac{1}{[1 + M e_i]^2} |M(1 + M e_i) - e_i|^2 \]

Substituting (3.8.7), (3.8.9), and (3.8.12) into (3.8.4) yields the total overall adaptive cascade SINR.

\[ SINR_{cascade} = \frac{K^2 e_d |M(1 + M e_i) - e_i|^2}{K[M(1 + M e_i)^2 + e_i - 2(1 + e_i M)] + K^2 e_i |U_i^T U_d|^2} \]

\[ SINR_{cascade} = \frac{e_d |M(1 + M e_i) - e_i|^2}{M(1 + M e_i)^2 + e_i - 2(1 + e_i M) + e_i} \]

If it is assumed that the interference is nulled well by the first stage then \( \rho \) is small, this leads to the following simplification

\[ SINR_{cascade} = K M e_d \]

\[ = N e_d \]

3.9 Performance After Convergence of Several Structures

Independent of the algorithm used to adapt the weights in
the cascade structures the gain of the proposed structures can be evaluated. The expression for the cascade SINR of a two stage array is used to compare different arrays.

\[
SINR_{\text{cas}} = \frac{P_d}{P_1 + P_n}
\]

\[
SINR_{\text{cascade}} = \frac{G_2^H G_1^H R_{ss} G_1 G_2}{G_2^H G_1^H [R_{11} + R_{nn}] G_1 G_2}
\] (3.9.1)

in general

\[
SINR_{\text{cascade}} = \frac{G_N^H \cdots G_1^H R_{ss} G_1 \cdots G_N}{G_N^H \cdots G_1^H [R_{11} + R_{nn}] G_1 \cdots G_N}
\]

\[
= \frac{W_{eq}^H R_{ss} W_{eq}}{W_{eq}^H [R_{11} + R_{nn}] W_{eq}}
\] (3.9.2)
### Table 3.1 Optimum Performance Comparison

<table>
<thead>
<tr>
<th>SINR</th>
<th>Non-cascade</th>
<th>Cascade</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINR</td>
<td>$\frac{W^H R_{ss} W}{W^H (R_{11} + R_{nn}) W}$</td>
<td>$\frac{W_{eq}^H R_{ss} W_{eq}}{W_{eq}^H (R_{11} + R_{nn}) W_{eq}}$</td>
</tr>
<tr>
<td>SINR with one interference</td>
<td>$e_d \left[ N - \frac{</td>
<td>U_d^T U_d^*</td>
</tr>
<tr>
<td>MSE</td>
<td>$\frac{A_d^2}{2} - 2A_d^3 W^H U_d^* + W^H R_{xx} W$</td>
<td>$\frac{A_d^2}{2} - 2A_d^3 W_{eq} U_d^* + W_{eq}^H R_{xx} W_{eq}$</td>
</tr>
<tr>
<td>Maximum Eigenvalue</td>
<td>$Me_d + 1$</td>
<td>$Me_d + 1$</td>
</tr>
<tr>
<td>Number of Mult. to Find Output</td>
<td>$N$</td>
<td>$\left[ \frac{N-1}{M-1} \right] + M \sum_{i=0}^{[K-i(M-1)]}$</td>
</tr>
<tr>
<td>Complexity of Weight Update</td>
<td>$N^3$</td>
<td>$M^3 \left[ \frac{N-1}{M-1} \right]$</td>
</tr>
</tbody>
</table>
3.10 **Cascade Eigenvalues**

As discussed in Section 2, the maximum eigenvalue is caused by the strongest interferer. The function of the first stage is to null out the strongest interferer, so that the maximum eigenvalue of the cascade input covariance matrix will be smaller than that of the front stage. The reduced maximum eigenvalue allows a larger step size for the cascade LMS algorithm.

3.10.1 **Adaptive Control of μ**

Consecutive stages of the cascade network may use larger values of μ due to the eigenvalue spread reduction that takes place in the cascade input covariance matrix. Since misadjustment is proportional to the product μTr[R_{xx}] and Tr[R_{xx}] decreases at cascade stages a larger μ may be used while maintaining the same misadjustment at succeeding stages. When the network nears convergence the value of μ should be smaller to cut down on the misadjustment noise.

3.11 **Array Weight Sensitivity**

Nitzberg [41], Jablon [31], and Davis [13] have discussed array weight sensitivity. If the weights are quantized and quadrature weighting is used. At convergence

\[ \hat{W} = W_{opt} + \Delta \hat{W} \]

where \( \Delta \hat{W} \) is a vector of random weight errors. If uniform quantization is used over the range -1 to +1 then the MSE is
$MSE = \frac{\Delta^2}{12}$  \hspace{1cm} (3.11.3)

For independent errors in the quadrature channels the MSE is doubled.

$$\sigma^2 = \frac{\Delta^2}{6}$$

The array output power is

$$P_{out} = E \left[ (W_{opt} + \Delta W) R_{xx} (W_{opt} + \Delta W) \right]$$

$$= W_{opt}^H R_{xx} W_{opt} + E \left[ W^H R_{xx} W \right]$$

representing $\Delta W$ by $\sigma^2 U$, where $U$ is a vector of uncorrelated elements whose variance is one. The last term is then

$$P_{out} = E \left[ (W_{opt} + \Delta W) R_{xx} (W_{opt} + \Delta W) \right]$$

$$= W_{opt}^H R_{xx} W_{opt} + E \left[ W^H R_{xx} W \right]$$  \hspace{1cm} (3.11.4)

$P_{out}$ is then

$$P_{out} = P_{opt} + \sigma^2 \text{ Tr}[R_{xx}]$$

$$= P_{opt} \left[ 1 + \frac{\sigma^2 \text{ Tr}[R_{xx}]}{P_{opt}} \right]$$  \hspace{1cm} (3.11.5)

where $\sigma^2$ represents undesired output power. The interference power out of the array is

$$P_I = \epsilon_i E \left[ |W^T U_i|^2 \right]$$

$$= \epsilon_i E \left[ |(W_{opt} + \Delta W)^T U_i|^2 \right]$$

When the interference is deeply nulled
\[ W_{opt}^T U_i = 0 \]

and

\[
P_i = e_i E \left[ |W^T U_i|^2 \right] = e_i E \left[ |\sigma U^T U_i|^2 \right]
\]

\[= e_i \sigma^2 E \left[ U^T U_i \right]^2 = e_i \sigma^2 E \left[ U_i^H U U^T U_i \right] \]

\[= \sigma^2 e_i N \quad (3.11.8)\]

Thus the quantization noise is proportional to the number of elements. Setting the excess noise equal the level of the thermal noise of the array

\[ P_i(\text{max}) = N \sigma^2 e_i \]

Solving for the variance of the weights

\[ \sigma = \frac{\sigma n}{e_i} \quad (3.11.9) \]

Substituting into (3.11.3)

\[ = \sqrt{e_i} \frac{1}{\sqrt{e_i}} \quad (3.11.10) \]

The weight range is 2 so the number of steps is

\[ N = \frac{2}{\Delta} \]

The first \( n \) satisfying

\[ 2^n \geq N \]

\[ n \geq \log_2(N) \]

is chosen as the word size for the weight vector.
3.12 Example Cascade Structures

The following figures constitute a family of cascade arrays that lower the computational complexity of the weight calculation. In all cases the subarrays overlap because if the phase centers of the subarrays are too far separated then grating nulls will appear, as discussed in Section 2.8. Figure 3.3 illustrates a two element per subarray three stage cascade. In Figure 3.3 each of the input subarrays has the same pattern \( g_{\text{sub}}(\theta) \). The front stage of Figure 3.3 can be regarded as three antennas with pattern \( g_{\text{sub}}(\theta) \). Their phase shift vector is then

\[
U_d = g_{\text{su}}(\theta) \begin{bmatrix} 1 \\ e^{-j\phi_d} \\ e^{-2j\phi_d} \end{bmatrix}
\]

which is the same as a three element array with each element having the pattern \( g_{\text{sub}}(\theta) \).
Figure 3.3 Two Element Subarrays
The first layer of subarrays performs transformation $G_1$ of the inputs. The second stage performs transformation $G_2$ of the first stage outputs. Finally the last stage performs transformation $G_3$ of the second stage outputs. This produces the final output. The output of the first stage is

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 \\ 0 & 0 & w_{11} & w_{12} \end{bmatrix} X$$

$$= G_1^T X$$

the output of the second stage is

$$Z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} w_{21} & w_{22} & 0 \\ 0 & w_{21} & w_{22} \end{bmatrix} Y$$

$$= G_2^T Y$$

$$= G_2^T G_1^T X$$

the final output is

$$s(t) = w_{31} z_1 + w_{32} z_2$$

$$= [w_{31} \ \ w_{32}] Z$$

$$= G_3^T G_2^T G_1^T X$$

where

$$G_3 = [w_{31} \ \ w_{32}]^T$$

at each stage there is an input covariance matrix. For the first stage $R_{xx}$ only elements $r_{11}$, $r_{12}$, $r_{21}$, and $r_{22}$ need to be
calculated. Since the partition is symmetric $r_{21} = r_{12}$ and only three terms need to be calculated. The weights at each stage must be computed using a Frost constraint to prevent the subarrays from shutting off when their degrees of freedom are overcome. The gain is constrained to be one in the desired signal direction. For the two element subarray case the gain in the product of the transformations at each stage. The general form for transformation of a stage is

$$G_k^T = \begin{bmatrix}
    w_{k1} & w_{k2} & 0 & 0 & \ldots & 0 & 0 \\
    0 & w_{k1} & w_{k2} & 0 & \ldots & 0 & 0 \\
    0 & 0 & w_{k1} & w_{k2} & 0 & \ldots & 0 \\
    \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    0 & 0 & 0 & \ldots & w_{k1} & w_{k2} & 0 \\
    0 & 0 & 0 & \ldots & 0 & w_{k1} & w_{k2}
\end{bmatrix} \quad (3.12.1)$$

where for stage one the dimensions are $(N-1) \times N$. For stage two the dimensions are $(N-2) \times (N-1)$. This form of network forms a tree of decreasing dimensions.

Figure 3.4 illustrates a three element per subarray two stage array. The resulting formulas are similar to Figure 3.3, however the individual stage transformations now form a band of three weights in each row of the stage transformation matrices. Figure 3.5 illustrates a four element per subarray two stage array. Figure 3.6 illustrates a 3 element per subarray four stage array.
Figure 3.4 Three Element Subarrays

Figure 3.5 Four Element Subarrays
Figure 3.6 Four Stage Cascade
A two element cascade exists for any N element array. However, if three element subarrays are used a pure three element cascade exists only for \( N = 3 + 2i \) elements, where \( i = 0, 1, \ldots \). In general for M element subarrays a perfect cascade exists only for
\[
N = M + (M-1)i
\]
elements, where \( i = 0, 1, \ldots \).

For a two element cascade there are \( N-1 \) stages. For a three element cascade there are \( (N-1)/2 \) stages. The general formula for the number of stages using M element subarrays is
\[
\frac{N-1}{M-1}
\]
The number of multiplications necessary to produce an output of the cascade is
\[
M \sum_{i=0}^{[\frac{N-1}{M-1}]} [K-i(M-1)]
\]
as opposed to N for a fully adapted array.

3.13 Optimum Non-Cascade Solutions

To establish a basis for comparison the weights and pattern for a three and a five element array are calculated. Example 3.13.1 The parameters used to calculate the optimum weights in the examples follow: the interelement to wavelength spacing ratio is
\[
\frac{d}{\lambda} = 0.5
\]

the variance of the noise is

\[\sigma^2 = 1\]

the desired signal to Gaussian noise ratio

\[e_d = 1\]

The amplitude of the desired signal is

\[A_d = \sqrt{\sigma^2 e_d}\]

the interference to Gaussian noise ratio

\[e_1 = 10\]

The amplitude of the interference is

\[A_1 = \sqrt{\sigma^2 e_1}\]

the angle of the desired signal

\[\theta_d = 50^\circ\]

the angle of interference

\[\theta_1 = 90^\circ\]

First the optimum weight solution is calculated for comparison to the LMS algorithm solution. Using (2.3.9) the input covariance matrix is calculated
The cross correlation vector is calculated using (1.2.4).

\[
R_{xx} = \begin{bmatrix}
12 & -10.742-0.671j & 10.101+0.995j \\
-10.742+0.671j & 12 & -10.742-0.671j \\
10.101-0.995j & -10.742+0.671j & 12
\end{bmatrix}
\]

The optimum weight values are calculated using (1.2.6).

\[
P_N = \begin{bmatrix}
1 \\
-0.742+0.671j \\
0.101-0.995j
\end{bmatrix}
\]

Finally using (2.5.3) and the optimal array weights values the array pattern is plotted. Figure 3.7 is the optimal array pattern. The gain at the interference and desired signal angles is

\[
g(\theta_d) = 0.503 \\
g(\theta_i) = 0.04
\]

The voltage at the interference angle is -21.99 dB down from the gain at the desired signal angle. The SINR is found using (3.9.1).

\[
SINR_{conv} = 0.044 \text{ dB}
\]
Figure 3.7 Optimum 3 Element Array Pattern
Example 3.13.2. As a larger example for comparison a five element array is simulated using the same parameters. Using (2.3.9) the covariance matrix is

\[
R_{xx} = \begin{bmatrix}
12 & -10.7 - 0.7j & 10.1 + 1j & -9.4 - 0.8j & 9 + 0.2j \\
-10.7 + 0.7j & 12 & -10.7 - 0.7j & 10.1 + 1j & -9.4 - 0.8j \\
10.1 - 1j & -10.7 + 0.7j & 12 & -10.7 - 0.7j & 10.1 + 1j \\
-9.4 + 0.8j & 10.1 - 1j & -10.7 + 0.7j & 12 & -10.7 - 0.7j \\
9 - 0.2j & -9.4 + 0.8j & 10.1 - 1j & -10.7 + 0.7j & 12
\end{bmatrix}
\]

Using (1.2.4) the cross correlation vector \( P_N \) is

\[
P_N = \begin{bmatrix}
1 \\
-0.742 + 0.671j \\
0.101 - 0.995j \\
0.593 + 0.806j \\
-0.98 - 0.2j
\end{bmatrix}
\]

From (1.2.6) the optimum weights are

\[
\hat{w}_{\text{opt}} = \begin{bmatrix}
0.206 + 0.114j \\
-0.15 + 0.032j \\
0.01 - 0.103j \\
0.141 + 0.061j \\
-0.225 + 0.071j
\end{bmatrix}
\]

The gains at the desired signal and interference angles for the optimum 5 element array pattern are

\[
g(\theta_d) = 0.782
\]
\[
g(\theta_i) = 0.011
\]

The voltage at the interference angle is -37.04 dB down from
the gain at the desired signal angle. The SINR is found using (3.9.1).

\[ SINR_{conv} = 5.547 \text{ dB} \]

### 3.14 Optimum Cascade Patterns

**Example 3.14.1** Using the same parameters as Example 3.13.1 the optimum weights and pattern for Figure 3.8 are calculated. First using the same parameters as in section 3.13, the optimum subarray weights are found. Then the pattern is plotted using the optimum cascade weights. The Q matrices for this case are

\[ Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \]

\[ Q_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

Subarray input covariance matrix one is calculated using (3.3.1).

\[ R_{xx1} = \begin{bmatrix} 12 & -10.742-0.671j \\ -10.742+0.671j & 12 \end{bmatrix} \]

The subarray cross correlation is calculated using (3.6.15).

\[ P_1 = \begin{bmatrix} 1 \\ -0.742+0.671j \end{bmatrix} \]

From (3.6.7) the subarray optimum weights are
Figure 3.8 Two Element Cascade Array
The first stage transformation is

\[
G_1^T = \begin{bmatrix} w_{11} & w_{12} & 0 \\ 0 & w_{11} & w_{12} \end{bmatrix}
\]

The same weight values are used to compute the output of both first stage subarrays. Using (3.6.12) the cascade covariance matrix is

\[
R_{yy} = \begin{bmatrix} 0.552 & 0.334 + 0.119j \\ 0.334 - 0.119j & 0.552 \end{bmatrix}
\]

Using (3.6.9) the cascade cross correlation vector is

\[
P_y = \begin{bmatrix} 0.552 \\ 0.552 \end{bmatrix}
\]

Solving (3.6.7) the cascade stage weights are

\[
w_{21} = 0.831 - 0.131j \\
w_{22} = -0.704 + 0.46j
\]

The second stage transformation is

\[
G_2 = \begin{bmatrix} w_{21} \\ w_{22} \end{bmatrix}
\]

The overall transfer function for the weights of the cascade is

\[
G_1G_2 = \begin{bmatrix} 0.137 + 0.181j \\ -0.111 + 0.1j \\ -0.166 - 0.154j \end{bmatrix}
\] (3.14.1)
These weights are close in value to the weights of the non-cascaded case (Example 3.13.1). The stronger the interference the closer the weight values of the cascade and non-cascade case become. The voltage gain at the desired and interference signal directions is

\[ g(\theta_d) = 0.423 \]

\[ g(\theta_i) = 0.109 \]

The voltage gain at the interference angle is -11.78 dB down from the gain at the desired signal angle. The SINR is found using (3.9.1).

\[ \text{SINR}_{\text{cascade}} = -1.356 \text{ dB} \]

Figure 3.9 shows the optimum array pattern for Example 3.14.1.
Figure 3.9  3 Elements, 2 Element Subarrays
Example 3.14.2 The optimum cascade of a five element array treated as two element subarrays is calculated. The first stage subarray selection matrix $Q_1$ is

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The first stage subarray weights are

$$w_{11} = 0.127 + 0.238 \quad w_{12} = 0.065 + 0.262j$$

The first stage transformation is

$$G_1^T = \begin{bmatrix} w_{11} & w_{12} & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 \\ 0 & 0 & w_{11} & w_{12} \\ 0 & 0 & 0 & w_{11} \end{bmatrix}$$

The second stage covariance matrix is

$$R_{yy} = \begin{bmatrix} 0.25 & -0.02 - 0.06j & 0.05 + 0.06j & -0.01 - 0.05j \\ -0.02 + 0.06j & 0.25 & -0.02 - 0.06j & 0.05 + 0.06j \\ 0.05 - 0.06j & -0.02 + 0.06j & 0.25 & -0.02 - 0.06j \\ -0.01 + 0.05j & 0.05 - 0.06j & -0.02 + 0.06j & 0.25 \end{bmatrix}$$

with second stage cross correlation

$$P_y = \begin{bmatrix} 0.254 \\ -0.189 + 0.171j \\ 0.026 - 0.253j \\ 0.151 + 0.205j \end{bmatrix}$$

The second stage subarray selection matrix is
The second stage weights are then

\[ w_{21} = 0.831 - 0.131j \]
\[ w_{22} = -0.704 + 0.46j \]

The second stage transformation is then

\[
G_2^T = \begin{bmatrix}
w_{21} & w_{22} & 0 & 0 \\
0 & w_{21} & w_{22} & 0 \\
0 & 0 & w_{21} & w_{22}
\end{bmatrix}
\]

The third stage covariance matrix is then

\[
R_{zz} = \begin{bmatrix}
0.42 & -0.25 - 0.19j & 0.09 + 0.17j \\
-0.25 + 0.19j & 0.42 & -0.25 - 0.19j \\
0.09 - 0.17j & -0.25 + 0.19j & 0.42
\end{bmatrix}
\]

The third stage cross correlation is

\[
P_z = \begin{bmatrix}
0.423 \\
-0.313 + 0.258j \\
0.043 - 0.42j
\end{bmatrix}
\]

The third stage subarray selection matrix is

\[
Q_3 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]

The third stage subarray weights are

\[ w_{31} = 0.583 + 0.13j \]
\[ w_{32} = -0.346 + 0.488j \]

The third stage transformation is
The fourth and final covariance matrix is

\[ G_3^T = \begin{bmatrix} w_{31} & w_{32} & 0 \\ 0 & w_{31} & w_{32} \end{bmatrix} \]

The final stage cross correlation vector is

\[ \begin{bmatrix} 0.493 & -0.301 - 0.268j \\ -0.301 + 0.268j & 0.493 \end{bmatrix} \]

The final stage weights are

\[ w_{41} = 0.55 + 0.019j \]
\[ w_{42} = -0.395 + 0.383j \]

The final stage transformation is

\[ G_4^T = [ w_{41} \quad w_{42} ] \]

Using (3.5.6) to find the overall equivalent weights yields

\[ W_{eq} = G_4^T G_3^T G_2^T G_1^T \]

with

\[ W_{eq} = \begin{bmatrix} 0.029 + 0.069j \\ -0.188 - 0.005j \\ 0.021 - 0.213j \\ 0.185 + 0.033j \\ -0.042 + 0.062j \end{bmatrix} \]
The optimum array pattern is

Figure 3.10 5 Elements, 2 Element Subarrays

Figure 3.11 5 Elements, 2 Element Subarrays
The voltage at the interference angle is -13.87 dB down from the gain at the desired signal angle. The SINR is found using (3.9.1).

\[ \text{SINR}_{\text{cascade}} = 0.743 \text{ dB} \]

**Example 3.14.3.** The optimum weights are calculated for the five element array with three element subarrays in Figure 3.4. For Figure 3.4

\[ Q_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \]

Using (3.3.1) the subarray covariance matrix \( \mathbf{R}_{xx1} \) is found. \( \mathbf{P}_{w1} \) is found using (3.3.2). The optimum first stage subarray weights are found using (3.3.3).

\[
\begin{align*}
    w_{11} &= 0.202 + 0.267j \\
    w_{12} &= -0.073 + 0.066j \\
    w_{13} &= -0.246 - 0.228j \\
\end{align*}
\]

The first stage transformation matrix is

\[
G_1^T = \begin{bmatrix}
    w_{11} & w_{12} & w_{13} & 0 & 0 \\
    0 & w_{11} & w_{12} & w_{13} & 0 \\
    0 & 0 & w_{11} & w_{12} & w_{13} \\
\end{bmatrix}
\]

Figure 3.9 shows the optimum array pattern of Example 3.14.3.
Figure 3.12 5 Element Cascade, 3 Element Subarrays
The second stage covariance matrix and cross-correlation vector are found using (3.6.12) and (3.6.9).

\[ R_{xy} = \begin{bmatrix} 0.503 & -0.197-0.235j & -0.069+0.232j \\ -0.197+0.235j & 0.503 & -0.197-0.235j \\ -0.069-0.232j & -0.197+0.235j & 0.503 \end{bmatrix} \]

\[ P_{xy} = \begin{bmatrix} 0.503 \\ -0.373+0.33j \\ 0.051-0.5j \end{bmatrix} \]

The optimum cascade weights are

\[ w_{21} = 0.582-0.243j \]
\[ w_{22} = -0.249+0.225j \]
\[ w_{23} = 0.3-0.555j \]

The second stage transformation is

\[ G_2 = \begin{bmatrix} w_{21} \\ w_{22} \\ w_{23} \end{bmatrix} \]

The total overall transformation is \( \mathbf{a}_2^\top \mathbf{G}_1^\top \) yielding

\[ \mathbf{w}_{eq} = \begin{bmatrix} 0.182+0.107j \\ -0.137+0.035j \\ 0.014-0.138j \\ 0.127+0.062j \\ -0.2+0.068j \end{bmatrix} \]

The cascade gain at the desired signal and interference angles is
The voltage at the interference angle is $-21.84$ dB down from the gain at the desired signal angle. The cascade SINR is found using (3.9.1).

\[ \text{SINR}_{\text{cascade}} = 4.864 \text{ dB} \]

**Example 3.14.4** A five element array is done using 4 element subarrays. The first stage subarray selection matrix $Q_1$ is

\[
Q_1 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The first stage subarray covariance matrix is

\[
\begin{bmatrix}
12 & -10.7-0.7j & 10.1+1j & -9.4-0.8j \\
-10.7+0.7j & 12 & -10.7-0.7j & 10.1+1j \\
10.1-1j & -10.7-1j & 12 & -10.7-0.7j \\
-9.4+0.8j & 10.1-1j & -10.7+0.7j & 12
\end{bmatrix}
\]

The first stage cross correlation vector is

\[
P_{x1} = \begin{bmatrix}
1 \\
-0.742+0.671j \\
0.101-0.995j \\
0.593+0.806j
\end{bmatrix}
\]

The first stage optimum weight solution is
The first stage transformation matrix is

\[ G_1^T = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & 0 \\
0 & w_{11} & w_{12} & w_{13} & w_{14} \end{bmatrix} \]

The second stage covariance matrix is

\[ R_{yy} = \begin{bmatrix} 0.68 & -0.395-0.393j \\
-0.395+0.393j & 0.68 \end{bmatrix} \]

The second stage cross correlation vector is

\[ P_y = \begin{bmatrix} 0.68 \\
-0.505+0.456j \end{bmatrix} \]

The second stage optimum weights are

\[ w_{21} = 0.552-0.117j \]
\[ w_{22} = -0.488+0.284j \]

The second stage transformation is

\[ G_2 = \begin{bmatrix} w_{21} \\
w_{12} \end{bmatrix} \]

The equivalent set of weights is

\[ W_{eq} = \begin{bmatrix} 0.012+0.086j \\
-0.07-0.044j \\
0.029-0.046j \\
0.063+0.086j \\
-0.041-0.003j \end{bmatrix} \]
Figure 3.13 5 Elements, 4 Element Subarrays
The optimum array pattern is

Figure 3.14 5 Elements, 4 Element Subarrays
The voltage gain at the desired and interference signal directions is

\[ g(\theta_d) = 0.752 \]

\[ g(\theta_i) = 0.024 \]

The voltage gain at the interference angle is -29.92 dB down from the gain at the desired signal angle. The SINR is found using (3.9.1).

\[ \text{SINR}_{\text{cascade}} = 4.815 \text{ dB} \]
Table 3.2 Example Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>$g(\theta_d)$</th>
<th>$g(\theta_r)$</th>
<th>dB Down</th>
<th>SINR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 elements</td>
<td>0.782</td>
<td>0.011</td>
<td>37.04</td>
<td>5.547</td>
</tr>
<tr>
<td>5 elements with 4</td>
<td>0.752</td>
<td>0.024</td>
<td>29.92</td>
<td>4.815</td>
</tr>
<tr>
<td>element subarrays</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 elements with 3</td>
<td>0.754</td>
<td>0.061</td>
<td>21.84</td>
<td>4.864</td>
</tr>
<tr>
<td>element subarrays</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 elements with 2</td>
<td>0.543</td>
<td>0.11</td>
<td>13.87</td>
<td>0.743</td>
</tr>
<tr>
<td>element subarrays</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 elements</td>
<td>0.503</td>
<td>0.04</td>
<td>21.99</td>
<td>0.044</td>
</tr>
<tr>
<td>3 elements with 2</td>
<td>0.423</td>
<td>0.109</td>
<td>11.78</td>
<td>-1.356</td>
</tr>
<tr>
<td>element subarrays</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3.3 Example dB Off Optimum Performance

<table>
<thead>
<tr>
<th></th>
<th>Null Depth Off Optimal</th>
<th>SINR Off Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 elements</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5 elements with 4 element subarrays</td>
<td>+7.12</td>
<td>-0.7320</td>
</tr>
<tr>
<td>5 elements with 3 element subarrays</td>
<td>+15.2</td>
<td>-0.683</td>
</tr>
<tr>
<td>5 elements with 2 element subarrays</td>
<td>+23.17</td>
<td>-4.804</td>
</tr>
<tr>
<td>3 elements</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3 elements with 2 element subarrays</td>
<td>+10.21</td>
<td>-1.4</td>
</tr>
</tbody>
</table>
3.15 SINR Versus Interference Angle

The following plots show the SINR performance of a five element array using 2, 3, and 4 element subarrays. The SINR was found using (3.9.2). The optimum SINR has been evaluated for the example 5 element cascades of Chapter 3. However, the desired signal angle has been taken to be zero degrees and the interference angle has been allowed to vary between 0 and 90 degrees. The larger the interference magnitude the closer the performances of the 2, 3, and 4 element subarray antenna systems become. It can be seen that the performance of the 3 and 4 element subarray examples is close to that of a fully adapted array. However, the 2 element subarray example performance falls short of the others.
Interference Angle in degrees

Fig. 3.15 $\epsilon_1 = 1$

Fig. 3.16 $\epsilon_1 = 10$
Fig. 3.17 \( \epsilon_1 = 100 \)

Fig. 3.18 \( \epsilon_1 = 1000 \)
3.16 Summary

In this chapter it has been shown that equivalent subarrays have equivalent covariance matrices. This means that the weights need be calculated for one subarray only. The other subarrays may use a copy of the first subarrays weights. A formula for the cascade gain has been developed for performance comparison. In addition, a formula for the cascade SINR has been developed. The fact that succeeding stages will have a lower eigenvalue spread has been shown. Several example arrays were evaluated in both cascade and non-cascade versions. A theoretical performance comparison is given in Table 3.1. Finally the optimum SINR of a fully adapted five element array is compared to the same array using 2, 3, and 4 element subarrays.
Chapter 4
Cascade Sample Matrix Inversion

4.0 Sample Matrix Inversion

An advantage of the sample matrix inversion technique is that its convergence speed is independent of the eigenvalue spread of $R_{xx}$, the signal powers, and the arrival angles. A disadvantage is that it is open loop and does not use the array output to correct for errors. If one of the weights is in error large errors can occur when nulling strong interference. A large number of bits is necessary to assure accurate performance. An additional problem is that the computer solution for the weights deteriorates as the eigenvalue spread increases. An accurate routine is necessary for high eigenvalue spread conditions.

Sample matrix inversion (SMI) requires estimating and inverting the $N \times N$ $R_{xx}$ matrix on a periodic basis. The estimate must be repeated on a periodic basis if the input signal statistics change. The SMI technique is an open loop technique and requires high accuracy in the combiner weight values. Feedback approaches, by contrast, can help to correct weight inaccuracies by adapting around them.

4.1 Complexity of Cascaded SMI Technique

Estimation of the $R_{xx}$ matrix requires $KN(N+1)/2$ complex multiplications where $K$ is the number of samples used in the estimate, and $N$ is the number of elements in the array.
Inversion of the resulting Hermitian matrix requires $N^{3/2} + N^2$ complex multiplications. Computing the weights requires $N^2$ multiplications. The method requires that $[N(N+3)]/2$ autocorrelation and cross-correlation measurements be made in order to obtain estimates of $R_{xx}$ and $P_n$. Finite processing speed prohibits the use of SMI for large arrays.

4.2 Estimation of the Covariance Matrix

The input covariance matrix can be estimated using the following maximum likelihood estimator, yielding an unbiased estimate having minimum variance

$$\hat{R}_{xx} = \frac{1}{K} \sum_{k=1}^{K} X_k^* X_k^T \quad (4.2.1)$$

The weights may be calculated using

$$W_{est} = \hat{R}_{xx}^{-1} U_d^* \quad (4.2.2)$$

if the desired signal direction is known. If the desired signal direction is not known then the cross-correlation vector must be estimated using

$$\hat{P}_N = \frac{1}{K} \sum_{j=1}^{K} X_j^* d_j$$

The weight vector is then calculated on a periodic basis using

$$W_{est} = \hat{R}_{xx}^{-1} \hat{P}_N \quad (4.2.3)$$

Both techniques allows the array to adapt to changing input
conditions. The matrix $R_{xx}$ is known as the sample covariance matrix. The weight vector is an estimate of the true optimum weight vector $W_{opt}$. Since $R_{xx}$ is only an estimate of the true covariance matrix the SINR will be slightly degraded. The SINR obtained with the optimum weights is

$$SINR_{opt} = A_d^2 \frac{|W_{opt}^T U_d|^2}{W_{opt}^T [R_{ii} + R_{nn}] W_{opt}}$$  \hspace{1cm} (4.2.4)$$

The value of the $SINR_{est}$ with the value of the weights obtained using the weights determined with the estimated covariance matrix is

$$SINR_{est} = A_d^2 \frac{|W_{est}^T U_d|^2}{W_{est}^T [R_{ii} + R_{nn}] W_{est}}$$  \hspace{1cm} (4.2.5)$$

The ratio of the two SINRs

$$\rho = \frac{SINR_{est}}{SINR_{opt}}$$  \hspace{1cm} (4.2.5)$$

is a random variable with range

$$0 \leq \rho \leq 1$$  \hspace{1cm} (4.2.7)$$

To determine the number of samples necessary for $R_{xx}$ estimation it is necessary to relate the average value of $\rho$ to the number of samples used in the estimation. Reed et al. [44] have derived an expression for the pdf of $\rho$. The probability density function is a beta function.
where

\[ 0 \leq \rho \leq 1 \quad (4.2.9) \]

\( N \) is the number of elements in the array and \( K \) is the number of samples. The average value of \( \rho \) is found from

\[ E[\rho] = \int_0^1 \rho \Pr(\rho) \, d\rho = \frac{K+2-N}{K+1} \quad (4.2.10) \]

When \( \rho = \frac{1}{2} \), there is a 3 dB degradation in the SINR using the SMI method. For less than 3 dB degradation \[44\]

\[ \frac{K+2-N}{K+1} > \frac{1}{2} \quad (4.2.11) \]

yielding

\[ K > 2N-3 \quad (4.2.12) \]

The number of necessary samples does not depend upon the eigenvalue spread. When the sample matrix inversion SMI technique is used the number of samples necessary before the estimate of the covariance matrix will yield a set of weights \( \mathbf{W}_{est} \) with an output within 3 dB of the optimal is

\[ 2N-3 \quad (4.2.13) \]

where \( N \) is the number of elements in the array. Thus a smaller array requires less samples to perform the estimate of the input covariance matrix. For example a nine element array
requires 15 samples.

Smaller subarrays require a corresponding smaller number of input samples in order to estimate the input covariance matrix. The number of operations for weight calculation is proportional to $N^3$. The number of calculations increases quickly with array size. For a nine element array 729 computations are necessary.

An additional matrix estimation equation [28] is

$$
\hat{R}_{xx} = (1-\alpha) \sum_{n=0}^{N} X_{k-n}^* X_{k-n}^T \quad \text{for } 0<\alpha<1
$$

(4.2.15)

this can be estimated recursively by

$$
\hat{R}_{xx}^n = \alpha \hat{R}_{xx}^{n-1} + (1-\alpha) X_k^* X_k^T
$$

(4.2.15)

The SMI technique can be applied to a large array by breaking the large array into smaller subarrays that can be adapted using the SMI technique. The subarray outputs are then cascaded into another array which is also adapted using the SMI technique. The cascading of arrays allows the succeeding stages to monitor preceding stages.

To perform the sample SMI method on a fully adapted array 2N-3 computations are necessary for $R'_{xx}$ estimation followed by $N^3/2 + N^2$ computations for matrix inversion and $N^2$ calculations for the weight calculation. So the number of multiplications is

$$
\frac{3}{2} [N^3+N^2-N]
$$

(4.2.16)
Next the number of computations necessary to implement the cascaded SMI method with different subarray sizes is examined. For an array of $N$ elements that is broken into subarrays of 2 elements each there are $N-1$ levels in a cascaded array. Each layer in the cascade is composed of two element subarrays. For two element subarrays the number of computations is

\[(\text{no. of levels}) \times (\text{estimation delay per level}) + (\text{no. of levels}) \times (\text{matrix inversion operations})\]

\[= 2M(N-1) + \left[ \frac{M^3}{2} + M^2 \right] (N-1) \quad \text{where } M=2\]

\[= 4(N-1) + 8(N-1)\]

\[= 12(N-1) \quad (4.2.17)\]

as opposed to

\[\frac{3}{2}[N^3 + N^2 - N] \quad (4.2.18)\]

multiplications for a fully adapted array.

In general the equation for the estimation delay of a cascaded array of size $N$ with subarray size $M$ is

\[\frac{N-1}{M-1} [2M-3]\]

Equation (4.2.18) is valid only if estimation takes place at each stage. This is not necessary. Only the first stage needs to be estimated. The succeeding covariance matrices can be found using (3.6.12). In this case the total number of
samples for estimation needs to be

\[ 2N - 3 \]

The estimation occurs at the first stage only. The succeeding stage covariance matrices do not need to be estimated but can be calculated using the first stage covariance matrix. If estimation is done at each stage the number of estimation samples is

\[ \frac{N-1}{M-1} \left[ 2M - 3 \right] \]

The total number multiplications for matrix inversion in the cascade is

\[ \frac{N-1}{M-1} \left[ \frac{M^3}{2} + M^2 \right] \]  \hspace{1cm} (4.2.20)

The number of multiplications for weight calculation in the cascade is

\[ \left[ \frac{N-1}{M-1} \right] M^2 \]

The total number of multiplications for a cascade is

\[ \frac{3}{2} \frac{N-1}{M-1} [M^3 + M^2 - M] \]

However, when viewed from an individual processor perspective the number of multiplications is

\[ \frac{3}{2} [M^3 + M^2 - M] \]
4.3 Matrix Inversion

Note that it is not necessary to compute the inverse of $R_{xx}$, an iterative procedure such as Gaussian elimination is less complex. If there are only $N_s \ll N$ strong sources present then the number of samples necessary for estimation may be relaxed to $2N_s$ if the estimated covariance matrix $R'_{xx}$ is rendered invertible by augmenting its diagonals by an $I$ supplement:

$$R'_{xx} = I + \frac{1}{K} \sum_{k=1}^{K} X_k^* X_k^T$$  \hspace{1cm} (4.3.1)

The matrix inverse may be found recursively by the inversion formula [37]

$$[R'_{xx}]^{-1} = \alpha^{-1} [R_{xx}, k-1]^{-1} - \frac{(1-\alpha) \alpha^{-2} R^{-1} X_k^* X_k^T R_{xx}, k-1}{1 + (1-\alpha) \alpha^{-2} X_k^T R_{xx}, k-1 X_k^*}$$

This recursive estimate may then be used to find a recursive equation for $W_{est}$ by substitution

$$W_{est} = R_{xx}, k^{-1} U_d$$

This recursive technique is due to Lunde [28]. It requires $N^2$ computations per sample as opposed to $N^3$ operations for sample inversion.

4.4 Weight Sensitivity

When the SMI technique is used in a fully adapted array the output is very sensitive to errors in the weights [40]. These errors may be caused by hardware imperfections in the
implementation. The SMI technique is open loop hence there is no feedback to correct for hardware imperfections or weight errors as in other approaches incorporating feedback. Also with very high eigenvalue spreads the condition number of the input covariance matrix becomes very high and numerical inaccuracies can occur in the inversion routine. The iterative matrix inversion may be modified to conform to a Kalman filter with feedback.

4.5 Simulation of SMI

As in Chapters 3 and 4 a three element and five element array are simulated. The SMI algorithm is used.

Example 4.5.1. The SMI algorithm is used to adapt the weights of a three element array. The same parameters are used as in Example 3.13.1. Ten samples are used to estimate $R_{xx}$. The voltage gains at the desired signal and interference angles are

$$g(\theta_d) = 0.66409$$

$$g(\theta_i) = 0.08177$$

The interference gain is -18.19 dB down from the desired signal gain. This is 8.65 dB off the optimal performance of Example (3.13.1). From (3.9.1) the SINR is

$$\text{SINR}_{\text{conv}} = -1.50351 \text{ dB}$$

which is -1.547 dB off the optimal performance of Example (3.13.1).
Figure 4.1 Three Element SMI Array Pattern
The number of multiplications for the 3 element SMI example is 50. Figure 4.1 shows the array pattern.

**Example 4.5.2.** The five element array of Example 4.4.2 simulated using the SMI technique. The voltage gains at the desired signal and interference angles are

\[ g(\theta_d) = 6.884 \]

\[ g(\theta_i) = 0.858 \]

The interference gain is -18.09 dB down from the desired signal gain. This is 18.65 dB off the optimal performance of Example (3.13.2). From (3.9.1) the SINR is

\[ SINR_{conv} = 3.843 \text{ dB} \]

which is -1.704 dB off the optimal performance of Example (3.13.2). The number of multiplications for the 5 element SMI example is 218. Figure 4.2 shows the array pattern.
Figure 4.2 Five Element SMI Array Pattern
4.6 Simulation of Cascaded SMI

The input covariance matrix $R_{xx}$ of the array is directly estimated. Then the subarray covariance matrix is found using

$$R_{xx1} = Q_1^T R_{xx} Q_1$$

The weights of the first subarray are found using

$$W_1 = R_{xx1}^{-1} Q_1 U_d^*$$

The first subarray weights are used to form the $G_1$ matrix. The covariance matrix of the next stage is found using

$$R_{yy} = G_1^H R_{xx} G_1$$

The second stage subarray covariance is found using

$$R_{yy1} = Q_2^T R_{yy} Q_2$$

The weights of the second stage subarray are used to find $G_2$, and then the same process repeats at the next stage.

**Example 4.6.1.** A three element array broken into two element subarrays is simulated using the same parameters as Example 3.14.1. This is a cascade SMI example. The voltage gains at the desired signal and interference angles are

$$g(\theta_d) = 0.289$$

$$g(\theta_i) = 0.083$$

The interference gain is $-10.84$ dB down from the desired signal gain. This is $+0.92$ dB off the optimal performance of Example (3.14.1).
Figure 4.3 3 Elements, 2 Element Subarrays
The number of multiplications for this example is 30. From (3.9.1) the SINR is

$$SINR_{cascade} = -2.30023 \text{ dB}$$

which is -1.945 dB off the optimal performance of Example (3.14.1). Figure 4.3 shows the array pattern.

**Example 4.6.2.** A five element array is treated as two element subarrays. This is the same array as Example 3.14.2. The same parameters are used. The voltage gain at the desired signal and interference angles is

$$g(\theta_d) = 0.489$$

$$g(\theta_i) = 0.166$$

The interference gain is -9.38 dB down from the desired signal gain. This is +4.49 dB off the optimal performance of Example (3.14.2). From (3.9.1) the SINR is

$$SINR_{cascade} = -3.681 \text{ dB}$$

which is -2.938 dB off the optimal performance of Example (3.14.2). The number of multiplications for this example is 60. Figure 4.4 shows the array pattern.
Figure 4.4 5 Elements, 2 Element Subarrays
Example 4.6.3. A five element array is treated as three element subarrays. This is the same array as Example 3.14.3. The same parameters are used. The voltage gain at the desired signal and interference angles is

\[ g(\theta_d) = 0.752 \]

\[ g(\theta_i) = 0.071 \]

The interference gain is -20.5 dB down from the desired signal gain. This is -1.34 dB off the optimal performance of Example (3.14.3). From (3.9.1) the SINR is

\[ \text{SINR}_{\text{cascade}} = 3.199 \text{ dB} \]

which is -1.665 dB off the optimal performance of Example (3.14.3). The number of multiplications for this example is 99. Figure 4.5 shows the array pattern.
Figure 4.5 5 Elements, 3 Element Subarrays
Example 4.6.4. A five element array is treated as four element subarrays. This is the same array as Example 3.14.4. The same parameters are used. The voltage gain at the desired signal and interference angles is

\[ g(\theta_d) = 1.1268 \]

\[ g(\theta_i) = 0.0543 \]

The interference gain is -26.35 dB down from the desired signal gain. This is -2.21 dB off the optimal performance of Example (3.14.4). From (3.9.1) the SINR is

\[ SINR_{cascade} = 2.037 \text{ dB} \]

which is -2.778 dB off the optimal performance of Example (3.14.4). The number of multiplications for this example is 228. Figure 4.6 shows the array pattern.
Figure 4.6 5 Elements, 4 Element Subarrays
4.7 **Comparison of Cascaded SMI with Non-Cascaded SMI**

The cascaded SMI requires $2M-3$ as opposed to $2N-3$ samples for estimation of the input covariance matrix. The cascaded SMI technique also needs

$$\frac{N-1}{M-1} M^3$$

operations for matrix inversion as opposed to $N^3$ for a fully adapted array. Consider a 9 element array with 3 element subarrays in this case there are 4 layers in the cascade. The number of calculations for the cascade is $4(3^3) = 108$. For a fully adapted array $9^3 = 729$ calculations are necessary. Both arrays require the same number of samples for covariance matrix estimation. The following tables provide a summary and performance comparison.
Table 4.1 Performance Comparison Cascaded SMI

<table>
<thead>
<tr>
<th></th>
<th>Non-cascade</th>
<th>Cascade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation Delay</td>
<td>$2N-3$</td>
<td>$\left<a href="2M-3"> \frac{N-1}{M-1} \right</a>$</td>
</tr>
<tr>
<td>Inversion Complexity</td>
<td>$N^3$</td>
<td>$M^3 \left[ \frac{N-1}{M-1} \right]$</td>
</tr>
<tr>
<td>Complexity of Output</td>
<td>$N$</td>
<td>$\frac{N-1}{N-1-M} \sum_{i=0}^{M-1} [K-i(M-1)]$</td>
</tr>
<tr>
<td>Computation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total No. of Multiplications</td>
<td>$\frac{3}{2}[N^3+N^2-N]$</td>
<td>$\frac{3}{2}[N-1][M^3+M^2-M]$</td>
</tr>
<tr>
<td>Multiplication per Stage</td>
<td>$\frac{3}{2}[N^3+N^2-N]$</td>
<td>$\frac{3}{2}[M^3+M^2-M]$</td>
</tr>
</tbody>
</table>
Table 4.2 Simulated SMI Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>$g(\theta_d)$</th>
<th>$g(\theta_i)$</th>
<th>dB Down</th>
<th>SINR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 elements</td>
<td>6.884</td>
<td>0.858</td>
<td>-18.09</td>
<td>3.843</td>
</tr>
<tr>
<td>5 elements with 4</td>
<td>1.1268</td>
<td>0.0543</td>
<td>-26.35</td>
<td>2.037</td>
</tr>
<tr>
<td>element subarrays</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 elements with 3</td>
<td>0.752</td>
<td>0.071</td>
<td>-20.5</td>
<td>3.19</td>
</tr>
<tr>
<td>element subarrays</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 elements with 2</td>
<td>0.489</td>
<td>0.166</td>
<td>-9.38</td>
<td>-3.681</td>
</tr>
<tr>
<td>element subarrays</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 elements</td>
<td>0.664</td>
<td>0.0817</td>
<td>-18.19</td>
<td>-1.504</td>
</tr>
<tr>
<td>3 elements with 2</td>
<td>0.289</td>
<td>0.083</td>
<td>-10.84</td>
<td>-2.300</td>
</tr>
</tbody>
</table>
Table 4.3 Simulated SMI Performance Off Optimal Cascade

<table>
<thead>
<tr>
<th>Elements</th>
<th>dB Down</th>
<th>SINR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 elements</td>
<td>+18.95</td>
<td>-1.704</td>
</tr>
<tr>
<td>5 elements with 4 element subarrays</td>
<td>+3.57</td>
<td>-2.778</td>
</tr>
<tr>
<td>5 elements with 3 element subarrays</td>
<td>+1.34</td>
<td>-1.674</td>
</tr>
<tr>
<td>5 elements with 2 element subarrays</td>
<td>+4.49</td>
<td>-4.424</td>
</tr>
<tr>
<td>3 elements</td>
<td>+3.8</td>
<td>-1.548</td>
</tr>
<tr>
<td>3 elements with 2 element subarrays</td>
<td>+0.94</td>
<td>-0.944</td>
</tr>
</tbody>
</table>
Table 4.4 Simulated SMI Performance Off Optimal Non-Cascade

<table>
<thead>
<tr>
<th>Configuration</th>
<th>dB Down</th>
<th>SINR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 elements</td>
<td>+12.51</td>
<td>-1.704</td>
</tr>
<tr>
<td>5 elements with 4 element subarrays</td>
<td>+2.93</td>
<td>-3.51</td>
</tr>
<tr>
<td>5 elements with 3 element subarrays</td>
<td>+1.34</td>
<td>-2.35</td>
</tr>
<tr>
<td>5 elements with 2 element subarrays</td>
<td>+4.49</td>
<td>-9.228</td>
</tr>
<tr>
<td>3 elements</td>
<td>+3.8</td>
<td>-1.548</td>
</tr>
<tr>
<td>3 elements with 2 element subarrays</td>
<td>+11.15</td>
<td>-2.344</td>
</tr>
</tbody>
</table>
4.8 Summary

The computational complexity reduction of the cascade SMI technique has been discussed in this chapter. The SMI technique has been used to simulate the same examples as in Chapter 3. Finally, Table 4.2 is a performance comparison for the cascade SMI technique. It has been shown that the cascade SMI technique converges very quickly.
Chapter 5

Coherent Interference Rejection

5.0 Coherent Interference and Spatial Dithering

Adaptive algorithms fail in the presence of coherent interference. The definition of the desired and interference signal to the array is

\[ X(n) = e^{j\omega t} \left[ A_d e^{j\phi_d} + A_i e^{j\phi_i} \right] \text{ for } 0 < n < (N-1) \]

where \( A_d \) is the magnitude of the desired signal, \( A_i \) is the magnitude of the interference, \( n \) is the number of the array element, and \( \phi_d \) and \( \phi_i \) are the random phases for the desired signal and interference respectively. The random phases are chosen from the following distribution.

\[ f_\psi(\psi) = \frac{1}{2\pi} \text{ for } \pi \leq \psi \leq -\pi \]

where \( f_\psi(\psi) = 0 \) otherwise. For the interference to be coherent the random phases of the desired signal and interference must be zero or equal.

Example 5.1 To demonstrate what happens to an adaptive array in the presence of coherent interference, Example 4.5.1 is simulated with the random phase angles of the desired signal and interference set equal to zero. This is a three element array. This simulation yields the gain (Fig. 5.1) at the desired and interference signals and the output SINR

\[ g(\theta_d) = 0.0704 \]
Example (5.1) illustrates the failure of the SMI algorithm to null the coherent interference.

The structure of cascade adaptive arrays leads to spatial dithering. Motivated by the need for arrays to function in the presence of coherent interference, Widrow et al. [53] have proposed the spatial dither algorithm. This algorithm functions in the presence of coherent interference by electronically dithering the antenna position in directions orthogonal to the look direction. Other names for this method are the 3/4" plywood method and the spatial averaging method. This method breaks up the array into groups of subarrays. Each of the subarrays is then adapted. The spatial dither method breaks the coherence of the interference up by modulating the interference and spreading the interference power out in the frequency domain.

The phase relation between the desired signal and the coherent interference is randomized by moving the receiving point. The phases at a particular element change differently according to the directions of arrival. If the cross correlation of the desired signal and a coherent interference is measured at several points it will be reduced by averaging. This is the concept behind the use of spatial dither. [46]
Fig. 5.1 3 Elements with Coherent Interference
With uniform averaging of the subarray covariance matrices the averaged subarray covariance matrix for subarray 1 is

\[ R_{xx1}' = \frac{1}{K} \sum_{n=1}^{N} R_{xx1} \]  

\[ = \frac{1}{K} \sum_{n=1}^{K} Q_i R_{xx} Q_i^T \]  

Nonuniform averaging may also be used.

\[ R_{xx1}' = \sum_{i=1}^{N} v_i R_{xx1} \]  

where

\[ \sum_{i=1}^{N} v_i = 1 \]

The cascade structure that the spatial averaging method and cascade adaptive arrays have in common suggests the investigation of the coherent interference rejection properties of cascade adaptive arrays. The cascade structure breaks the array up into smaller arrays having fewer inputs. Under certain conditions cascade arrays converge quickly. In some cases the cascade structure is the signal processing technique of choice. This is a technique for handling non-uniform arrays if the antenna elements can be partitioned into similar sets of elements that can be combined in the same subarrays.

Shan and Kailath [46] demonstrated that an array can
function in the presence of coherent interference by using spatial dither. Spatial dither can be used with any conventional algorithm. Shan and Kailath have shown that in a coherent signaling environment the covariance matrix has some zero eigenvalues. Minimization of the mean square error will force the output of the beamformer to fall to zero. Spatial dithering restores the input covariance matrix to full rank.

Conventional arrays will fail if any of the interfering signals is coherent with the desired signal. Shan and Kailath have studied only linear arrays with equally spaced identical sensors. They suggest that unequally spaced or non-collinear arrays are also worthy of investigation.

In spatial dithering, at any instant of time the total array is divided into overlapping subarrays of m samples each. These subarrays are then fed in succession into the adaptive processor, which updates an n-dimensional weight vector each time. After all the subarrays have been processed, the same procedure is repeated with the next data snapshot. All the groups for a given snapshot contain the same signal, but the interferences in different groups have different phase relations. The output of these groups is run into the adaptive processor one by one. Using different running sequences will result in different spatial smoothings that all break up the coherence between the signal and interference. This method has been used in direction finding and adaptive
beamforming. The recovered signal is sensitive to spatial smoothing rate. Signal distortion can result from using a high adaptation rate. For each snapshot the method requires considerable computation.

Su's [48] has a method of implementing spatial dither. When the desired signal is unknown but its direction is known a priori then spatial dither can be used. Su's algorithm for spatial dither has a parallel structure. At convergence the spatial averaging output is the maximum likelihood estimate of the desired signal.

Su's parallel spatial processing algorithm, Figure 5.2, is based on a group of subarrays having the same structure. The subarrays function in parallel. The algorithm requires the same number of computations as a conventional array. The N array elements are partitioned into n subarrays containing m elements each. Subarrays are allowed to have overlapping elements just as long as the overlapping is done in the same way for each of the subarrays. The total number of elements is \( N = mn \). Since each subarray is identical the subarrays have the same weights.

During the first pass the first beamformer updates its weights. Its weights are copied to the other subarrays. On the second snapshot the second array updates its weights and then they are copied to the other subarrays. The adaptation process cycles through the subarrays. When the adaptation process reaches the last subarray the whole process starts
Figure 5.2 Su's Spatial Dithering Technique
over again with the first array. Every subarray uses the same weights. The final output is the average of the subarray outputs. Shan's method requires \( N \) adaptations per pass whereas Su's parallel spatial method requires only one adaptation per pass. Spatial dithering can be done with cascade arrays.

5.1 Coherent Interference

When coherent interference exists expression (2.4.1) for \( R_{xx} \) no longer applies. In the coherent interference case the input signal at the \( n^{th} \) antenna element is

\[
x(t) e^{j(n-1)\phi_d} + \rho x(t) e^{j(n-1)\phi_I} + n_n(t)
\]  

(5.1.1)

In the coherent case the interference and the desired signal are correlated. \( \rho \) represents the relative magnitude and phase between the desired signal and the coherent interferer. Equation (5.1.1) can be rewritten as

\[
X = A_d e^{j\omega t} [U_d + \rho U_i] + X_n
\]

If a normalized input is used, the coherent interference case yields the received vector

\[
X = e_d U_d + \rho e_i U_i + X_n
\]

(5.1.1)

The normalized covariance matrix becomes

\[
R'_{xx} = E[X^*X^T]
\]
Signal cancellation is one the limitations of adaptive arrays. When signal cancellation occurs the array is not only nulling the interference but the signal also. Under certain circumstances an interferer can be devised to elude adaptive arrays whose purpose it is to eliminate interference. Thus some interferences are capable of breaking down the adaptive procedure of adaptive arrays. Widrow [53] has shown that the Griffith, Frost, Zahm, and Compton beamformers are subject to signal cancellation through interaction between the desired signal and the interference or noise. This interaction may cause the array to cancel out the desired signal. The adaptive array may be overcome by the interference through signal cancellation rather than being overwhelmed by the interference power. Considering a Frost beamformer, if the signal from the desired look direction is a sinusoid and there is an interference at the same frequency but slightly off the look direction, then minimizing the output power will cause the interference to be accepted with just the right magnitude and phase to cancel out the desired signal. In the case of narrowband signals signal cancellation results in signal loss.
Figure 5.3 Output with Signal Cancellation
In the case of wideband signals signal cancellation results in significant signal distortion.

5.2 Mathematical Analysis of Spatial Dithering

Adaptive arrays operate under the assumption that the interference is not coherent with the desired signal, so that there is no cross correlation between them. If the interference is not coherent then the input covariance matrix is not Toeplitz. Each element of the matrix is affected by the cross correlation between the coherent waves. Spatial averaging can be used to restore the input covariance matrix to Toeplitz form. The weights calculated using this modified matrix will suppress the coherent interference.

Spatial averaging of the covariance matrices of subarrays can be used. The technique divides the array into subarrays and averages the input covariance matrices of the subarrays to produce a resultant matrix whose dimensions are the same as a subarray. The size of the full array is sacrificed in order to decorrelate the incoming coherent waves. The phase relation between the desired signal and the coherent interference can be randomized by spatially moving the receiving point. When the array moves the phases of the waves at a particular element change differently according to their respective directions of arrival. Their phase relation fluctuates. The cross correlation of the desired signal and the interference is measured at several points and it tends to
be minimized by averaging. In this technique the input covariance matrix is partitioned by the subarrays.

The received input vector to succeeding subarrays is

$$X(n) = e_d \left[ e^{-j(n-1)\Phi_d} U_{d1} + e^{-j(n-1)\Phi_1} U_{i1} \right] + n_n(t) \quad (5.2.1)$$

where

$$U_{d1} = \begin{bmatrix} 1 \ e^{j\Phi} \ ... \ e^{j(M-1)\Phi} \end{bmatrix}$$

$$= Q_1 U_d$$

and

$$U_{i1} = \begin{bmatrix} 1 \ e^{j\Phi} \ ... \ e^{j(N-1)\Phi} \end{bmatrix}$$

$$U_{i1} = Q_1 U_i$$

The $i^{th}$ subarray covariance matrix is

$$R_{xxi} = E[X^*(n)X^T(n)]$$

Substituting (5.2.1) into (5.2.2) yields the normalized subarray covariance matrix.

$$R'_{xxi} = E\left[ e_d^2 [e^{j(i-1)\Phi_d} U_{d1} + \rho e^{j(i-1)\Phi_1} U_{i1}]^* [e^{j(i-1)\Phi_d} U_{d1} + \rho e^{j(i-1)\Phi_1} U_{i1}]^T \right]$$

$$+ \sigma^2 I$$

$$R'_{xxi} = e_d^2 E\left[ [e^{-j(i-1)\Phi_d} U_{d1}^* + \rho^* e^{-j(i-1)\Phi_1} U_{i1}^*] \right.\left. [e^{j(i-1)\Phi_d} U_{d1}^T + \rho e^{j(i-1)\Phi_1} U_{i1}^T] \right\} + \sigma^2 I$$

Taking the expression for the subarray covariance matrix and
substituting into (5.0.1) yields

\[
R_{xx1} = e_d^2 \langle U_{d1}^* U_{d1}^T \rangle + \rho U_{d1}^* U_{11}^T \left[ \frac{1}{K} \sum_{i=1}^{K} e^{j(i-1)\theta_i} \right] \\
+ \rho^* U_{11}^* U_{d1}^T \left[ \frac{1}{K} \sum_{i=1}^{K} e^{j(i-1)\theta_i} \right] + |\rho|^2 U_{11}^* U_{11}^T + \sigma^2 I
\]

5.2.1 Adaptively Combined Spatially Dithered Subarrays

In order to evaluate the performance of spatial dithering, first an equation for the spatially dithered subarray covariance matrix is derived. This is done by averaging the subarray covariance matrices.

\[
\hat{R} = \frac{1}{K} \sum_{i=1}^{K} R_{xxi} \quad (5.2.1.1)
\]

Substituting (5.2.1) into (5.2.1.1) yields

\[
\hat{R} = e_d^2 \langle U_{d1}^* U_{d1}^T \rangle \\
+ \left[ \frac{1}{K} \sum_{i=1}^{K} e^{j(i-1)\theta_i} \right] \rho U_{d1}^* U_{d1}^T \\
+ |\rho|^2 U_{11}^* U_{11}^T + \sigma^2 I
\]
Making the substitution

\[ c = \frac{1}{K} \sum_{i=1}^{K} e^{j(i-1)} (\phi_i \psi_i) \]

The spatially dithered subarray covariance matrix may be written as

\[ \mathbf{K} = e_d^2 \mathbf{H} \left( (c \mathbf{U}_{d1} + \rho \mathbf{U}_{ll})^* (c \mathbf{U}_{d1} + \rho \mathbf{U}_{ll}) \right) + \left( 1 - |c|^2 \right) \mathbf{U}_{d1}^* \mathbf{U}_{d1}^T + \sigma^2 \mathbf{I} \]

(5.2.1.2)

The subarray weight vector is chosen as

\[ \mathbf{W} = \mathbf{K}^{-1} \mathbf{U}_{d1}^* \]

(5.2.1.3)

Substituting (5.2.1.1) into (5.2.1.2) yields an equation for the weights of the spatially dithered subarray.

\[ \mathbf{W} = \left[ e_d^2 (c \mathbf{U}_{d1} + \rho \mathbf{U}_{ll})^* (c \mathbf{U}_{d1} + \rho \mathbf{U}_{ll}) + \sigma^2 \mathbf{I} \right]^{-1} \mathbf{U}_{d1}^* \]

(5.2.1.3)

Using the matrix inversion lemma on (5.2.1.3) yields

\[ \mathbf{W}_{\text{dithered}} = \frac{1}{\sigma^2} \left[ \mathbf{I} - \frac{e_d (c \mathbf{U}_{d1} + \rho \mathbf{U}_{ll})^* (c \mathbf{U}_{d1} + \rho \mathbf{U}_{ll})}{e_d |c \mathbf{U}_{d1} + \rho \mathbf{U}_{ll}|^2 + 1} \right] \mathbf{U}_{d1}^* \]

The signal to noise ratio for a single subarray is

\[ \text{SINR}_{\text{dithered}} = \frac{e_d^2 \mathbf{H} \left( (U_{d1} + \rho \mathbf{U}_{ll})^T \mathbf{W}_{\text{dithered}} \right)}{\sigma^2 \mathbf{W}_{\text{dithered}}^H \mathbf{W}_{\text{dithered}}} \]

(5.2.1.4)
The array gain for the desired signal is

\[ e_d W_{dithered}^T U_{d1} = M - \frac{e_d |CM + \rho U_{d1}^T U_{11}|^2}{e_d |CU_{d1} + \rho U_{11}|^2 + 1} \quad (5.2.1.5) \]

The array gain for the interference is

\[ U_{d1} W_{dithered}^T = M - \frac{e_d |CM + \rho U_{d1}^T U_{11}|^2}{e_d |CU_{d1} + \rho U_{11}|^2 + 1} \quad (5.2.1.6) \]

The noise gain for the array is

\[ \sigma^2 W M = \sigma^2 \left[ M - \frac{e_d |CM + \rho U_{d1}^T U_{11}|^2}{e_d |CU_{d1} + \rho U_{11}|^2 + 1} \right] \quad (5.2.1.7) \]

Substituting (5.2.1.5), (5.2.1.6), and (5.2.1.7) into (5.2.1.4) yields the SINR equation.

\[ SINR_{dithered} = e_d |1-c|^2 \left[ M - \frac{e_d |CM + \rho U_{d1}^T U_{11}|^2}{e_d |CU_{d1} + \rho U_{11}|^2 + 1} \right] \]

Maximum SINR is obtained when c is zero. In this case the subarray SINR is

\[ SINR_{subarray} = e_d M \]

If the K subarrays are combined in the second stage transformation then

\[ SINR = e_d^2 M K = e_d^2 N \]

If interference is present the objective is to minimize the value of c.
5.3 Performance of Adaptively Combined Spatially Dithered Subarrays

Three examples of coherent interference rejection are done here in order to demonstrate spatial dithering. The first example uses a 3 element array broken into 2 two element array. The second example uses a five element array broken into 3 element subarrays. The third and final example uses a 5 element array broken into 2 element subarrays. Spatial dithering is done with the first two stages.

Example 5.3.1 To demonstrate the effects of spatial dithering example 3.14.1 is simulated using a coherent interference. This is a three element array using two element subarrays. The simulation yields

\[ g(\theta_d) = 0.66 \]

\[ g(\theta_i) = 0.1104 \]

\[ SINR = -3.63 \text{ dB} \]

Figure 5.4 shows the resulting array pattern.
Fig. 5.4 3x2 with Coherent Interference
Example 5.3.2 In this case a five element array broken into three element subarrays is simulated using spatial dithering of the subarray and coherent interference.

\[ g(\theta_d) = 1.197 \]

\[ g(\theta_f) = 0.122 \]

\[ SINR = 3.058 \text{ dB} \]

Figure 5.5 shows the resulting array pattern.
Fig. 5.5 Coherent Interference
Example 5.3.3 In this case a five element array broken into two element subarrays is simulated using spatial dithering of the subarrays and coherent interference. The simulation yields:

\[ g(\theta_d) = 1.728 \]

\[ g(\theta_i) = 0.571 \]

\[ SINR = -2.201 \text{ dB} \]

Dithering is also done on the second stage. Figure 5.6 shows the resulting array pattern.
Fig. 5.6 5x2 with Coherent Interference
5.4 **Summary**

In this chapter the SMI method has been used with spatial averaging to break up coherent interference. The spatial dither is done on multiple layers in the last example. Formulas for the SINR using this method are also derived. In previous usage by others array aperture is sacrificed using the spatial dither method. They only use one subarray to compute the output of the array. This chapter shows that it is not necessary to sacrifice array aperture.
Chapter 6
The Cascaded LMS Algorithm

6.0 The LMS Algorithm and Cascade Arrays

An adaptive finite impulse response filter consists of both a filter structure and an algorithm for adjusting the tap weights of a linear combiner filter. The filter output is a linear combination of the past inputs, or in the case of an adaptive array, of array inputs. An FIR filter contains no feedback and is non-recursive. The LMS algorithm is useful because of its computational simplicity. The basic principle behind the operation of adaptive signal processing algorithms is the searching of a performance surface to locate the optimum settings for the weights of a linear combiner. Gradient search methods are appropriate for searching the performance surface for an optimum. Newton's method, the method of steepest descent, and random search methods may be used to search the performance surface.

The LMS algorithm is a gradient method of searching the performance surface to find the location of the performance minimum. A special estimate of the gradient that is valid for the adaptive linear combiner is used by the LMS algorithm. This estimate requires that the input vector and the desired response be available at each iteration. For a linear combiner the elements of the input vector may be interpreted as simultaneous inputs from different sources. This is the multiple input case. For a linear combiner the input vector
may be interpreted as sequential samples of the same signal source or simultaneous samples of the same wavefront.

6.1 The Complex LMS Algorithm

The output of the linear combiner can be expressed as either

\[ y = \text{Re}[W^T\mathbf{w}] = \text{Re}[W^T\mathbf{x}] \]  

(6.1.1)

or

\[ y = \text{Re}[X^H\mathbf{w}] = \text{Re}[W^H\mathbf{x}] \]

depending on the sign convention\(^1\) used for \(\mathbf{w}\). For the first case

\[ \mathbf{w} = w_i + jw_i \]

For the second case

\[ \mathbf{w} = w_i - jw_i \]

The weights are complex numbers each weight can produce a magnitude and a phase change. In an adaptive array the array weights are implemented by a three port device called a quadrature hybrid. The quadrature hybrid splits the incoming waveform into an inphase and quadrature path as seen in Figure 6.1. For the LMS algorithms the error is defined as the difference between the output of the linear combiner and the reference.

\(^1\) Both conventions are common in the literature.
Fig. 6.1 Quadrature Hybrid
The reference signal, the array output, and the error are real. A complex input signal is used to model the quadrature hybrid. The real part of output is taken to find the array response. The LMS algorithm uses the gradient of the actual error and not the actual mean square error gradient. The gradient estimate is called a stochastic or noisy gradient estimate. To obtain the gradient estimate for the MSE consider

\[ e_k = d_k - \text{Re}\left[ w_k^T x_k \right] \]  
(6.1.2)

In the LMS algorithm the gradient of \( E(\epsilon_k^2) \) is estimated as

\[ \nabla_w [\epsilon_k^2] = 2\epsilon_k \nabla_w [\epsilon_k] = 2\epsilon_k [d_k - w^T x^*] = 2\epsilon_k [X_k^*] = -2\epsilon_k X_k^* \]  
(6.1.3)

Using a steepest descent algorithm to update the weight vector at each iteration the equation to update the weight vector is

\[ w_{k+1} = w_k - \mu \nabla E(\epsilon_k^2) \]

\[ = w_k + 2\mu \epsilon_k X_k^* \]  
(6.1.4)

where the parameter \( \mu \) adjusts the speed of adaptation. This algorithm converges to an optimum where \( E(\epsilon_k^2) \) is minimized. As an alternative to the complex notation real notation may
also be used. In this case the algorithm has both inphase and quadrature phase components. In this case the algorithm is expressed as

\[ w_{k+1}^i = w_k^i + 2\mu e_k x_k^i \]  
(6.1.5a)

\[ w_{k+1}^q = w_k^q + 2\mu e_k x_k^q \]  
(6.1.5b)

\( w^i \) and \( w^q \) denote the inphase and quadrature weight vectors. \( x^i \) and \( x^q \) denote the inphase and quadrature signal vectors. The error is

\[ e_k = d_k - s_k = d_k - \sum_{j=1}^{N} W_{j,k} x_{j,k} + \sum_{j=1}^{N} W_{j,k} x_{j,k} \]

At convergence the optimum weight vector is given by the Wiener-Hopf equation. Note that convergence of the LMS algorithm is guaranteed only if the step size \( \mu \) satisfies

\[ 0 < \mu < \frac{1}{\text{Tr}[R_{xx}]} \]

When there is a large eigenvalue spread in the input covariance matrix \( R_{xx} \) the LMS algorithm converges slowly because the value of \( \mu \) must be small to insure convergence.

The LMS method operates without knowledge of the direction of arrival and the spectrum of the signal, but with no knowledge of the noise field. No direct measurements of the noise field are necessary. The LMS method is a computationally less complex method. It does not require the computation of correlation coefficients or the inversion of
matrices. The input signals are used only once, as they occur in the adaptation process, and they do not need to be stored. Only the values of the linear combiner coefficients need to be stored.

6.2 Convergence Constraints

For convergence of the LMS algorithm the constraint on the parameter \( \mu \) is

\[
0 < \mu < \frac{1}{\text{Tr}[R_{xx}]} \tag{6.2.1}
\]

however \( \lambda_{\text{max}} \) is bounded by the trace of \( R_{xx} \)

\[
\lambda_{\text{max}} \leq \sum_{i=1}^{N} \lambda_{i} = \text{Tr}[R_{xx}] \tag{6.2.2}
\]

which is in turn equal to the total power received by the array, i.e., the sum of the powers at each of the elements.

\[
P_{\text{tot}} = \sum_{i=1}^{N} \mathbb{E}[|x_i(n)|^2] \tag{6.2.3}
\]

so \( \mu \) is bounded by the reciprocal of the total input power

\[
0 < \mu < \frac{2}{\lambda_{\text{max}}} = \frac{2}{P_{\text{tot}}} \tag{6.2.4}
\]

for acceptable weight variance, i.e., small misadjustment noise a tighter bound on \( \mu \) is used.

\[
0 < \mu < \frac{1}{P_{\text{tot}}} \tag{6.2.5}
\]

The strongest interference input is suppressed by the initial
subarrays. Therefore $P_{\text{tot}}$ at the cascade input is smaller and $\mu$ for the cascaded array can be made larger than that of the initial subarrays. The larger $\mu$ of the cascade array yields faster convergence.

6.3 Weight Behavior

The general differential equation for the weights of an LMS array is [52]

$$\frac{d}{dt} W + k R_{xx} W = k P_n$$  \hspace{1cm} (6.3.1)

Specializing this to a first stage subarray yields

$$\frac{d}{dt} W_f + k Q_1 R_{xx} Q_1^T W_f = k Q_1 U_d^*$$  \hspace{1cm} (6.3.2)

where $W_f$ represents the subarray weights. The solution for the first stage weights during convergence is

$$W_f(t) = \sum_{i=1}^{N} C_i e^{-k \lambda_i t} + R_{xx} P_{x1}$$

Substituting the solution for $W_f(t)$ into the differential equation for $W_{\text{cas}}$ yields

$$\frac{d}{dt} W_{\text{cas}} + k G_1^H(t) R_{xx} G_1(t) W_{\text{cas}} = k G_1^H(t) P_x = k G_1^H(t) U_d^*$$  \hspace{1cm} (6.3.3)

this can be solved iteratively for the cascade weights. Discretizing the equation for the cascade weights yields

$$W_{\text{cas}}(n+1) = k G_1^H(n) U_d^* - k G_1^H(n) R_{xx} G_1(n) + W_{\text{cas}}(n)$$  \hspace{1cm} (6.3.4)
6.4 Simulation of LMS Algorithm

In order to establish a basis for comparison two non-cascaded LMS algorithm cases are demonstrated.

Example 6.4.1 The simulation parameters follow are the same as in Chapter 3. From the calculation of the covariance matrix in Example 3.13.1.

\[ 0 \leq \mu \leq \frac{1}{\text{Tr}[R_{xx}]} = 0.0277 \]

In the examples of this chapter \( \mu \) is selected as one tenth of its maximum. So

\[ \mu = 0.00277 \]

Next an LMS algorithm simulation is done. First a sinusoidal reference signal is generated for the antenna array. The reference signal is

\[ e^{j\omega_d t} \]

Then noisy input signals are generated for the three antenna elements. The input signal is composed of the desired signal, interference, and thermal noise. The thermal noise is modelled using a narrowband noise representation.

\[ n(t) = n_1(t)\cos\omega_d t - jn_2(t)\sin\omega_d t \]

where \( n_1(t) \) and \( n_2(t) \) are independent Gaussian processes. A noisy input signal is generated for antenna two. The appropriate phase shift is added to the signal and interference at antenna one. Also thermal noise independent of that at antenna 1 is added. Finally the noisy input
signal is generated for antenna three. The appropriate phase shift and independent thermal noise are added. The weights are adapted by the complex valued LMS algorithm, (6.1.4). The time behavior of the real and imaginary components of the array weights is displayed. Figure 6.3 shows the convergence of the real (in-phase) component of the array weights.
Figure 6.2 Noisy Inphase Signal Antenna 1

Figure 6.3 Inphase Array Weights
Figure 6.4 displays the imaginary (quadrature phase) parts of the array weights.

![Figure 6.4 Quadrature Part Array Weights](image)

The error signal is calculated using (6.1.2) and then plotted.

![Figure 6.5 Three Element Output Error](image)
The output signal is calculated using (6.1.1) and then plotted.

\[
\begin{align*}
\frac{A_d + A_1 + 3 \cdot s_k}{s_k} & \\
\frac{-A_d - A_1 - 3 \cdot s_k}{s_k}
\end{align*}
\]

Figure 6.6 Three Element Array Output
The LMS algorithm antenna pattern is plotted using the final weight values.

Fig. 6.7 Non-cascade 3 Element LMS Pattern
The gain at the desired signal and interference angles is

\[ g(\theta_d) = 0.65 \]

\[ g(\theta_i) = 0.36 \]

The gain at the interference angle is 5.13 dB down from the gain at the desired signal angle. The SINR is found using (3.9.1).

\[ SINR_{conv} = -5.49 \text{ dB} \]

**Example 6.4.2.** As a larger example for comparison the five element array of Example 3.13.2 is simulated using the LMS algorithm. The constraint on \( \mu \) the five element array is

\[ 0 \leq \mu \leq \frac{1}{\text{Tr}[R_{xx}]} = \frac{1}{60} = 0.01667 \]

So

\[ \mu = 0.001667 \]

The gains at the desired signal and interference angles for the 5 element LMS pattern are

\[ g(\theta_d) = 0.64 \]

\[ g(\theta_i) = 0.17 \]

The gain at the interference angle is 11.51 dB down from the gain at the desired signal angle. The SINR is found using (3.9.1).

\[ SINR_{conv} = 0.44 \text{ dB} \]
Figure 6.8 shows the array gain for Example 6.4.2 after 256 iterations.

Fig. 6.8 Five Element LMS Pattern
6.5 Simulation of Cascaded LMS

The two examples of this section show the behavior of the cascaded LMS algorithm.

Example 6.5.1 As cascade LMS example the five element array with three element subarrays in Example 3.14.4 is simulated. For the first stage

\[ 0 \leq \mu_x \leq \frac{1}{\text{Tr}[R_{xx1}]} = 0.02778 \]

with

\[ \mu_x = 0.002778 \]

For the second stage

\[ 0 \leq \mu_y \leq \frac{1}{\text{Tr}[R_{yy1}]} = 0.66269 \]

with

\[ \mu_y = 0.066269 \]

The final cascade weights are

\[ w_{21} = 0.582 - 0.243j \]
\[ w_{22} = -0.249 + 0.225j \]
\[ w_{23} = 0.3 - 0.555j \]

The second stage transformation is

\[ G_2 = \begin{bmatrix} w_{21} \\ w_{22} \\ w_{23} \end{bmatrix} \]

The total overall transformation is \( G_2^T G_1^T \) yielding
Figure 6.9 shows the array gain for Example 6.5.1 after 256 iterations.

Fig. 6.9 5 Elements, 3 Element Subarrays
The cascade gain at the desired signal and interference angles is
\[ g(\theta_d) = 0.297 \]
\[ g(\theta_1) = 0.188 \]

The gain at the interference angle is 3.97 dB down from the gain at the desired signal angle. The SINR is found using (3.9.1).

\[ \text{SINR}_{\text{cascade}} = -5.97 \text{ dB} \]

**Example 6.5.2.** As an additional cascade comparison the five element array with four element subarrays in Example 3.14.4 is simulated. For the first stage
\[ 0 \leq \mu_x \leq \frac{1}{\text{Tr}[R_{xx1}]} = 0.02083 \]

with
\[ \mu_x = 0.002083 \]

For the second stage
\[ 0 \leq \mu_y \leq \frac{1}{\text{Tr}[R_{yy1}]} = 0.73529 \]

with
\[
W_{eq} = \begin{bmatrix}
0.182 + 0.107j \\
-0.137 + 0.035j \\
0.014 - 0.138j \\
0.127 + 0.062j \\
-0.2 + 0.068j
\end{bmatrix}
\]
Figure 6.10 shows the array gain for Example 6.5.2 after 256 iterations.
The gains at the desired signal and interference angles are

\[ g(\theta_d) = 0.453 \]

\[ g(\theta_i) = 0.052 \]

The gain at the interference angle is 18.8 dB down from the gain at the desired signal angle. The SINR is found using (3.9.1).

\[ \text{SINR}_{\text{cascade}} = 1.316 \text{ dB} \]

6.6 **Comparison of Cascade and Non-cascade Performance**

The mean square error is just

\[ \mathbb{E}[d(t) - s(t)]^2 = \]

which when expanded becomes

\[ \text{MSE} = E[d_k^2] + W^T R_{xx} W - 2 P_n^T W \]  \hspace{1cm} (6.6.1)

For a conventional array this is

\[ \text{MSE}_{\text{con}} = \frac{A_d^2}{2} + W^T R_{xx} W - A_d U_d^T W \]  \hspace{1cm} (6.6.2)

whereas for a cascade array the MSE expression is

\[ \text{MSE}_{\text{cascade}} = \frac{A_d^2}{2} + W_{eq}^T R_{xx} W_{eq} - A_d U_d^T W_{eq} \]

6.7 **Frost Constraints and the LMS Algorithm**

The LMS–Frost algorithm can be broken down into parts. The first part is the same as the LMS algorithm with the array
output substituted for the error. In the second part of the algorithm a correction is made to assure that the gain is constrained in the desired signal direction.

\[ W_{k+\frac{1}{2}} = W_k + 2\mu y_k X_k \quad (6.7.1) \]

where \( y_k \) is the array output. Next the constraint error is found.

\[ C - [1 \ 1 \ \ldots \ \ 1] W_{k+\frac{1}{2}} \]

and the weights are adjusted in order to correct this error in the following way.

\[ \frac{1}{K} [C - [1 \ 1 \ \ldots \ \ 1] W_{k+\frac{1}{2}}] = E_{k+\frac{1}{2}} \quad (5.7.2) \]

Combining both Frost steps into one equation yields

\[ W_{k+\frac{1}{2}} = W_k + 2\mu y_k X_k + E_{k+\frac{1}{2}} \quad (6.7.3) \]

The following table provides a summary.
Table 6.1 Comparison of Non-cascaded and Cascaded LMS

<table>
<thead>
<tr>
<th></th>
<th>Non-cascaded</th>
<th>Cascaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>computations per update</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>maximum $\mu$</td>
<td>$\frac{1}{N M_1 + 1}$</td>
<td>$\frac{1}{M M_1 + 1}$</td>
</tr>
<tr>
<td>multiplications per output</td>
<td>N</td>
<td>$\frac{1}{M M_1 + 1}$</td>
</tr>
<tr>
<td>calculation</td>
<td></td>
<td>$\sum_{i=1}^{M-1} [K-i(M-1)]$</td>
</tr>
<tr>
<td>shortest time constant, $\tau_{\text{min}}$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>longest time constant, $\tau_{\text{max}}$</td>
<td>$N M_1 + 1$</td>
<td>$M M_1 + 1$</td>
</tr>
<tr>
<td>eigenvalue spread</td>
<td>$N M_1 + 1$</td>
<td>$M M_1 + 1$</td>
</tr>
<tr>
<td>misadjustment</td>
<td>$\mu T_x[R_{xx}]$</td>
<td>$\sum_{i=1}^{N-1} \mu_i T_x[R_{xx}]$</td>
</tr>
</tbody>
</table>
6.8 Summary

In this chapter the LMS algorithm convergence constraints and weight behaviour has been discussed. The LMS algorithm was applied in examples 6.5.1 and 6.5.2. It can be seen that smaller LMS arrays converge faster than larger LMS arrays. Table 6.1 is a list of performance parameters.
Chapter 7
Conclusions

The objective has been reduction in the complexity of calculating the weights for an adaptive antenna array. The technique has been to partition and cascade the inputs. Two major adaptive filtering algorithms were used as examples. These were the least mean squares (LMS) algorithm and the sample matrix inversion algorithm. These two algorithms differ widely in their computational complexity. SMI is $O(N^3)$, whereas LMS is $O(N)$. 

A framework for the discussion of cascade arrays was established in Chapter 2. An analytical method of calculating the array weights was shown. Next the eigenvalue spread was shown to be proportional to the number of array elements. An expression for calculating array gain was shown. Next the antenna array degrees of freedom is discussed, and the array pattern is shown to be equivalent to a polynomial. The number of degrees of freedom is important because if it is exceeded then the antenna array will turn itself off. The number of interferences sets the size of array or subarray necessary. If the number of degrees of freedom is overcome then a Frost constraint can be used. Finally in Chapter 2 grating nulls are discussed. Grating nulls occur when there is insufficient spatial sampling of the incoming wavefronts. The subarrays of a cascade array must overlap in order to avoid spatial aliasing or grating nulls.
Several examples of optimum cascade arrays were demonstrated in Chapter 3. The subarray covariance matrices are shown to be identical. It was shown that the subarrays at each stage have the same weight solution. A general formula for the overall equivalent weights and gain of a cascade array was derived. The equivalent weights of a cascade array approximate the weights of a fully adapted array. The approximation becoming better as the interference level is increased. A five element array was broken into subarrays of size 2, 3, and 4. The SINR performance of five elements broken into subarrays of size 2, 3, and 4 was shown to be nearly equivalent to a fully adapted five element array.

In Chapter 4 the sample matrix inversion algorithm is applied to the cascade structures of Chapter 3. Since the number of input samples necessary to estimate the covariance matrix is proportional to the size of the array, subarray have less estimation delay. The complexity of using the sample matrix inversion technique is greatly reduced by using subarrays in a cascade. Through the examples it was verified that the cascade array structures converge very quickly.

As an additional example of the application of the cascade structures spatial dithering is done in Chapter 5. In this case the subarray covariance matrices at a stage are all estimated. They are then averaged to produce a new subarray covariance matrix that is used to find the subarray weights.
This technique can be used to suppress coherent interference. Examples show this to be the case.

In Chapter 6, the LMS algorithm is applied to the same cascade array structures as in Chapter 3 and 4. The LMS algorithm convergence is shown to benefit from the cascade approach. Eigenvalue spread is reduced through the use of subarrays. Examples of the cascade LMS algorithm confirm these expectations.

In conclusion simulation software and mathematical models for non-cascade and cascade adaptive arrays were developed. Two cascaded adaptive algorithms, the least mean square LMS and sample matrix inversion SMI, were simulated and compared to that of non-cascade arrays using the same algorithms. Novel cascade array structures were proposed. Simulation software was used to spatially dither the subarrays of these structures in order to demonstrate coherent interference rejection. Various cascade configurations were evaluated.

In summary the following advantages of a cascade approach were demonstrated:

- Necessary processor complexity is lowered and a time division multiplexed (TDM) processor is possible.
- Reliability can be increased through the use of a pool of processors.
- Faster convergence with lower computational complexity is possible with the cascaded sample matrix inversion algorithm.
• Because the number of inputs in each subarray is limited, higher complexity algorithms are possible.
• The coherent interference rejection properties of spatial dithering can be exploited with the array.
• Covariance matrix estimation delay is decreased.
• Only one subarray per level need be calculated. The solution may be copied to the other subarrays on the same level.
• The LMS algorithm benefits from a eigenvalue spread reduction.
• Very large arrays may be processed using these cascade structures.
References


[58] Van Veen, B.D., "Improved Power Minimization Based Partially Adaptive Beamformer Design".


Cascade adaptive arrays that reduce processor requirements, increase reliability, and create modularity are proposed. These cascade adaptive arrays reduce processor complexity by breaking a large array up into a number of smaller subarrays that can then be processed by a smaller, more economical signal processor. Furthermore, cascade performance approaches that of a fully adapted array.

When coherent interference is present, adaptive arrays eliminate not only the interference from the output of the array, but can also suppress the desired signal. As an example of a cascade adaptive array, the spatial averaging method or the spatial dither method is used to alleviate coherent interference. This method breaks the array into subarrays. Spatial dithering is a natural by product of cascade subarrays. The subarrays are adapted and averaged. The spatial dither method breaks the interference coherence up by decorrelating it with the desired signal. The array can then differentiate between the signal and coherent interference.

Cascade adaptive array structures consisting of 2, 3, and 4 element subarrays were examined. The performance of these cascade arrays was compared to the non-cascade case. Two cascade adaptive algorithms, the least mean square LMS and sample matrix inversion SMI, were simulated and compared to that of non-cascade arrays.