Technical Report 1575
September 1992

Broadband Conventional Beamforming Incorporating Adaptive Equalization

Richard C. North

Technical Report 1575
September 1992

Broadband Conventional Beamforming Incorporating Adaptive Equalization

Richard C. North

NAVAL COMMAND, CONTROL AND OCEAN SURVEILLANCE CENTER
RDT&E DIVISION
San Diego, California 92152-5001

J. D. FONTANA, CAPT, USN
Commanding Officer

R. T. SHEARER
Executive Director

ADMINISTRATIVE INFORMATION

This work, conducted during FY 1992, was performed for the Office of Naval Technology, Office of the Chief of Naval Research, Arlington, VA 22217-5000, under program element 0602314N and work unit DN308291.

Released by
Paul Reeves, Head
Analysis and Simulation Division

Under authority of
Ed Shutters, Head
Surveillance Department

SM

EXECUTIVE SUMMARY

Energy radiated from a source usually propagates in many directions. To detect the spatial origin of the source of energy, data from an array of receiving elements (sensors) are summed coherently (in a spatial sense) to enhance the signal-to-noise ratio over a single element. This is called spatial processing or beamforming. Adaptive beamforming allows for spatial nulls to be positioned so that strong spatial interferers can be cancelled. In a multipath environment, however, multipath signals arrive at the array at an angle spatially different from the direct path as well as being time delayed (temporally different) from the direct path.

Unfortunately, the performance of most adaptive beamforming algorithms progressively deteriorates with the amount of correlation between sources, here direct and multipath. In addition, most adaptive beamforming algorithms attempt to null *all* spatially different sources and thus cannot make use of the multipath information.

This report introduces methods which attempt to coherently recombine the spatially and temporally different multipaths with the direct path to further enhance the array gain. These methods are based on utilizing a broadband conventional beamformer incorporating adaptive equalization to recombine multipath information. Preliminary results are compared with two well-known narrowband, purely spatial processing techniques: the Bartlett beamformer (also called a frequency-domain conventional beamformer) and the minimum variance distortionless response beamformer. While preliminary results are promising, several important research issues are outlined which remain to be addressed.

CONTENTS

FIGURES

TABLES

SYMBOLS

A^i, A^r, A^t	magnitude of the incident, reflected, and transmitted signals
c	speed of propagating wave (m/s)
d	array element spacing (m)
f	frequency (Hz)
M	number of elements
N	number of FIR filter taps or weights
P_y^i, P_y	instantaneous array output power and average array output power
$\boldsymbol{R}(f) = E\left[\boldsymbol{X}(f)\, \boldsymbol{X}(f)^H \right]$	spatial spectral correlation matrix
$\boldsymbol{R}(\vec{u}) = E\left[\boldsymbol{X}^a(k,\vec{u})\, \boldsymbol{X}^a(k,\vec{u})^H \right]$	spatial-temporal correlation matrix (or steered covariance matrix)
T	number of time samples
$\tau_m(\vec{u}) = \frac{\vec{x}_m \cdot \vec{u}}{c}$	required steering delay at the $x_m(t)$ element to look at direction $\vec{u}$
$\vec{u}$	array steering directional unit vector
$\boldsymbol{W}$	array element weight vector
$x_m(t)$	received signal at the m^{th} array element (includes source, interferers, and noise)
$\boldsymbol{X}^a(k,\vec{u})$	aligned (also called steered) array element data matrix
$y(k,\vec{u}),\ Y(f,\vec{u})$	time-domain array output and frequency-domain output for look direction $\vec{u}$

1. INTRODUCTION

The fundamental aims of an array processing system [9] are to

1. detect the presence of propagating energy
2. determine the location of the source of propagating energy
3. classify the source from its radiation waveform

The difficulty in designing signal processing algorithms to achieve these fundamental aims resides in the nature of the propagating energy. This energy is composed of *spatial and temporal* components. While joint spatial−temporal processing has been proposed in [15], for tractability, the assumption is usually made that the two components are separable, i.e., $x(t,\vec{u}) = y(t)\ z(\vec{u})$. Some sequence of temporal and spatial processing algorithms is therefore developed to meet the design objective. A typical sequence might be (1) temporal processing (element data conditioning, i.e., automatic gain control, low pass filter, FFT), (2) temporal processing (time averaging to form an estimate of the spatial spectral covariance matrix), and (3) spatial processing (narrowband beamforming). Even though in some applications the assumption ot spatial−temporal separation might be valid (or valid enough), there are numerous examples when such an assumption is not valid. For instance, when an attempt is made to detect a quiet source in a multipath environment, useful information can be present in the multipaths. However, the multipath incident energy is both *spatially and temporally* different from the direct path. In such a case, to coherently add the multipaths to the direct path signal, spatial processing cannot be separated from temporal processing.

The aim of this report is (1) to clarify the distinction between spatial and temporal processing in array processing and (2) to introduce several array processing algorithms suitable for real-time applications which combine both spatial and temporal processing. These algorithms are based on utilizing a broadband conventional time-domain beamformer incorporating adaptive equalization. This is a preliminary report. The merits and pitfalls of these algorithms are discussed and demonstrated in simulations in the hope of generating ideas and improvements.

1.1. SPATIAL PROCESSING (BEAMFORMING)

Beamforming refers to spatial signal processing algorithms used to focus an array of spatially distributed elements (also called sensors) to increase the signal-to-interference-plus-noise ratio (SINR) over that received by a single element [1]. It can be used to accomplish the first and second fundamental aims discussed above. Focusing is accomplished by

coherently (i.e., in-phase) summing the element outputs. Coherency is achieved by correcting for the appropriate phase delay in each element (created by their spatial separation) by making reasonable assumptions about the propagating wave's characteristics. Any a priori information, such as sound speed profiles or source range, can be used to obtain a more accurate estimate of the phase delay between elements in sensing the propagating energy wavefront.

Beamforming can be classified in two broad categories: conventional (or nonadaptive) and adaptive. In conventional beamforming (CBF), each element output of the array is weighted, delayed, and summed to align an incident wavefront coming from a particular direction. The resulting directivity or spatial filtering of the array is *data independent*; that is, the weights and delays are predetermined constants. Adaptive beamforming, on the other hand, adjusts its weights and delays *to the observed data*. For instance, a null can be steered automatically in the direction of a strong interferer to suppress its effects on the array output. If sufficient observations are available, adaptive beamforming can usually outperform conventional beamforming in achieving its fundamental aims. Only linear adaptive algorithms will be discussed in this text. Nonlinear adaptive algorithms, like Volterra filters or neural networks, are left to other sources.

Figure 1.1 shows schematically one way of relating several of the more popular beamforming algorithms (see [31] for a more comprehensive comparison of frequency-domain narrowband beamformers). The first classifier is the domain in which the beamforming

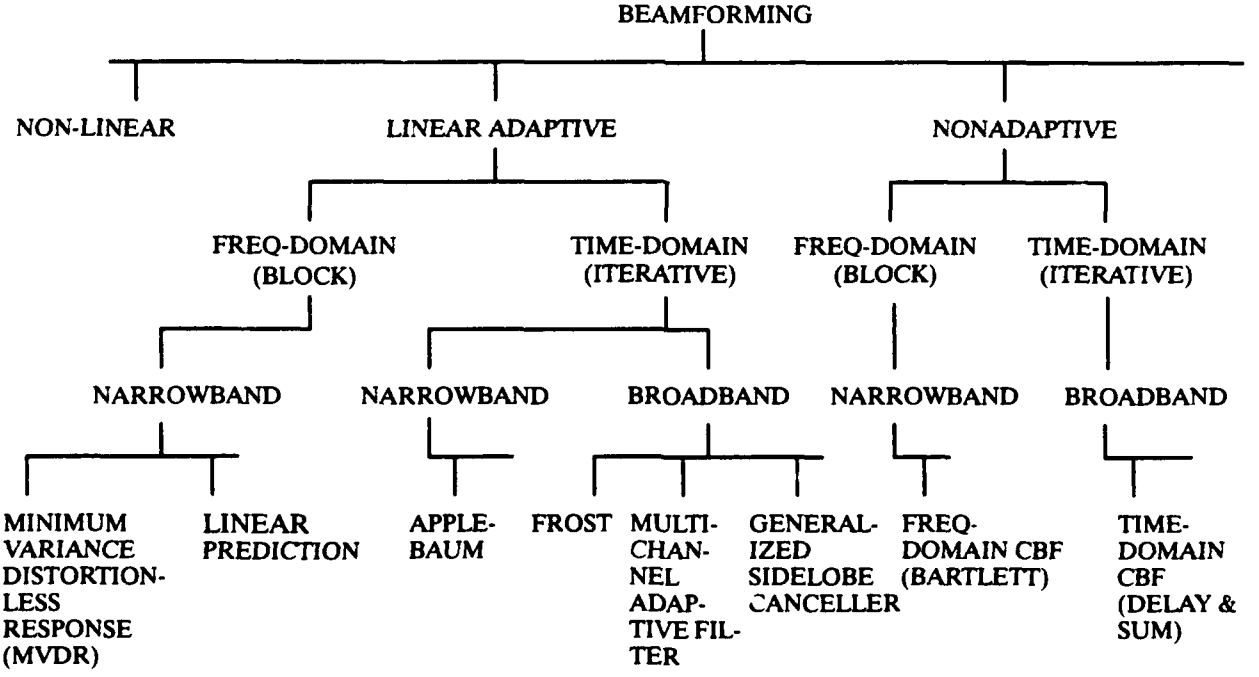


Figure 1.1. Relationship between some popular beamforming algorithms.

algorithm is computed, frequency (via the DFT) or time. The block/iterative classifier refers to whether the processing is computed on a block of data or with each new sample. Narrowband beamforming searches for the direction of an incident signal with frequency equal to that at which the processing is being performed. Broadband beamforming searches for the direction of an incident energy source (sum of all frequencies). A signal is usually defined as narrowband when the bandwidth of the incident sources is much less than the reciprocal of the propagation time of the wavefront across the array. More will be said on the use of narrowband beamforming with broadband data in Section 2.2.

1.2. TEMPORAL PROCESSING (CHANNEL EQUALIZATION)

Temporal processing refers to any operation which can be performed on a single channel of data, say from a single element. A channel includes everything between the transmitter, i.e., the source of energy, and the receiver, i.e., a single element of an array. The distinction between temporal processing and spatial processing is sometimes confusing because many of the standard temporal processing techniques (anti-alias filtering, quantization, Fourier transforms, discrete filtering, etc.) also carry over into spatial processing. In this text, we will be primarily concerned with the temporal processing concept of channel equalization.

Most real-world channels are nonideal in that their frequency response does not have constant amplitude and linear phase. A channel equalizer attempts to "invert" the frequency characteristics of the communications channel so that the frequency response of the cascaded channel-equalizer combination is ideal [12],[16]. When the channel is not known a priori or is time varying, it must be equalized by an adaptive equalizer. An adaptive equalizer automatically varies its impulse response in accordance with the temporal variations of the channel characteristics.

In a multipath channel, the received signal consists of the sum of a number of individual components that have traversed paths of differing lengths between the transmitter and the receiver. Depending on the phase relationships between these components, they can either add constructively or destructively at the receiver. The channel equalizer must sort out the multipaths from the direct path and coherently recombine all components into a single output.

1.3. SIGNAL MODEL

In this text, it will be assumed that L sources (which include interferers as well as the target source), $s_l(t)$ $l = 0, 1, 2, \ldots, L-1$, each generate a stochastic plane-wave signal which

propagates in an isovelocity medium through M spatially separated array elements, $x_m(t)$ $m = 0, 1, 2, \ldots, M-1$. A *source* refers to any incident plane-wave signal; thus, a single discrete energy source may actually generate multiple sources, a direct path plus any multipaths due to reflected or refracted paths. The plane-wave approximation is valid in an isovelocity medium for sources located at a distance from the array equal to roughly 60 or more times the aperture of the array (the so called "far-field" solution) [9]. An isovelocity medium implies that a wave travels at the same speed of propagation throughout all space. Note that a far-field source can generate a curved wavefront in a multivelocity medium (see [40] or [18]). It is assumed in this text that the speed of propagation and the exact element location is known.

Each source $s_l(t)$ produces a plane-wave response at the $x_m(t)$ element equal to

$$x_m(t) = s_l\left(t + \frac{\vec{x}_m \cdot \vec{s}_l}{c}\right) + n_m(t) \tag{1.1}$$

where

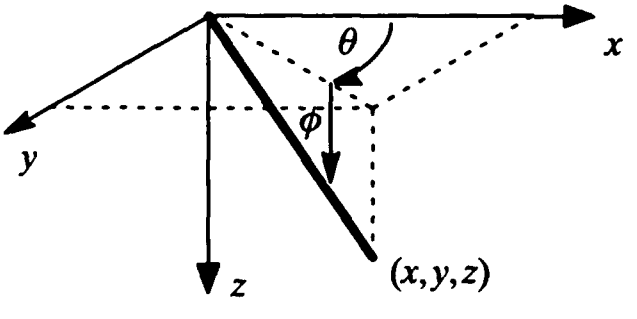
$\vec{x}_m$ = position vector for the $x_m(t)$ element

$\vec{s}_l$ = negative directional unit vector for the $s_l(t)$ source

c = speed of propagation of the acoustic plane wave

$n_m(t)$ = noise at the $x_m(t)$ element (assumed to be spatially uncorrelated)

$\left(\frac{\vec{x}_m \cdot \vec{s}_l}{c}\right)$ is the difference between the arrival time of the plane wave from $s_l(t)$ at the m^{th} element, $x_m(t)$, and the arrival time of the same wave at the origin. Thus, for $\vec{x}_m \perp \vec{s}_l$ or $\vec{x}_m = (0, 0, 0)$, this time difference will be zero. Figure 1.2 displays the coordinate system used throughout this text.



ϕ = ELEVATION ANGLE (+DOWN, − UP)

θ = AZIMUTH ANGLE (OR BEARING) (0 − NORTH)

$x = r \cos\phi \cos\theta$

$y = r \cos\phi \sin\theta$

$z = r \sin\phi$

Figure 1.2. Coordinate system.

2. GENERAL BEAMFORMING ISSUES

This section discusses several adaptive beamforming issues relevant to the development of the beamforming structures in Section 5. In particular, it briefly discusses the performance of the adaptive array with respect to the number of elements, the signal bandwidth, and a multipath environment.

2.1.PERFORMANCE VERSUS ARRAY SIZE

Array gain is the improvement in the SINR due to beamforming. One definition commonly used is [9]

$$array\ gain = \frac{SINR_{output}}{SINR_{input}} \tag{2.1}$$

(in Ch. 4.2.4 of [9], the noise term implies interferers plus noise as explicitly stated here; see also [8],[11]). Note that if a large interferer is cancelled by a beamformer, the array gain given by Eq. (2.1) can be quite large. The array gain at the output of an ideal M element array for the case of no interferers and spatially white, isotropic noise is equal to

$$array\ gain = 10\ \log_{10}(M)\ \ \text{dB} \tag{2.2}$$

regardless of the arrival direction (see, for example, p. 37 of [11]). As the interference-to-noise ratio (INR) increases, however, the array gain will change depending on (among other things) the number of interferers, the direction of each interferer, and the power in each interferer. For instance, if a 10-element array effectively cancels a 30-dB interferer, then the array gain given by Eq. (2.1) can be as large as 40 dB [= 10 log(10) dB + 30 dB]. For more on array gain see [5].

Any adaptive array with M independently controlled elements has M degrees of freedom, which are typically distributed as

$$M = [\text{no. of constraints}] + [\text{no. of steer directions}] + [\text{no. of interference nulls}] \tag{2.3}$$

For example, the minimum variance distortionless response (MVDR) has 1 constraint of constant magnitude in the single steer direction and so has a capability of generating $M-2$ interference nulls in its array response. When more interferers are incident on an array than the array is capable of cancelling, the adaptive weights will collapse (to zero) as the incident INRs increase and the array gain will deteriorate [8].

2.2. EFFECTS OF SOURCE SIGNAL BANDWIDTH

Broadband signals create (at least) three separate problems in array processing. The first deals with the fact that the interelement phase shift is a function of frequency. To see this, consider the interelement phase shift for a broadband signal

$$inter-element\ phase\ shift\ = 2\pi f\left[\tau_m(\vec{u}) - \tau_{m-1}(\vec{u})\right]$$

where $\tau_m(\vec{u}) = \frac{\vec{x}_m \cdot \vec{u}}{c}$ is the time it takes a plane-wave coming from direction $\vec{u}$ to propagate from the $x_m(t)$ element to the origin. The interelement phase shift is seen to be linearly dependent on the frequency of the signal. We can investigate the ramifications of this dependence further by considering the uniform linear array (ULA) with element spacing d. For this case and a signal incident to the array from angle ϕ, we find

$$inter-element\ phase\ shift = 2\pi f\left[\frac{d}{c}\ sin(\phi)\right] \qquad \text{(ULA)} \qquad (2.4)$$

From Eq. (2.4), it is apparent that a single signal with f_{source} arriving from angle ϕ_{source} can be perceived by the array as having a different frequency content, f_{array}, and arriving from a different angle, ϕ_{array}, as long as the interelement phase shift is unchanged, or $f_{array}\ sin(\phi_{array}) = f_{source}\ sin(\phi_{source})$. This can happen in narrowband beamforming whenever $f_{array} \neq f_{source}$. In these cases, the source appears to be incident from an angle

$$\phi_{array} = \sin^{-1}\left[\frac{f_{array}}{f_{source}}\ sin(\phi_{source})\right] \qquad \text{(ULA)} \qquad (2.5)$$

For narrowband frequency-domain beamforming, two situations can arise where $f_{array} \neq f_{source}$: (1) when the temporal DFT performed on each element's time data is not of adequate size so that spectral leakage between frequency bins is significant, and (2) when the spatial spectral correlation matrix, $\boldsymbol{R(f_1)}$, and the steering vector, $\boldsymbol{E(f_2,\bar{u})}$ (defined in Section 4), are formed at different frequencies.

Figure 2.1 demonstrates the effect of an inadequate size of temporal FFT, with the MVDR adaptive beamformer processed at 15.625 Hz (bin centered for a 1024-point FFT). The incident signal on the 5-element vertical ULA with d=30-m spacing is a narrowband signal at 25.0 Hz coming in at −14 degrees (c=1500 m/s) and sampled at 1000 Hz (all examples in this text are sampled at 1000 Hz). One would expect the MVDR processor to yield zero output when processing is done at any frequency other than 25.0 Hz. However, figure 2.1 shows that for FFT sizes of 32, 64, and 256 points, the spectral leakage from the 25.0-Hz source into the 15.625-Hz bin creates an apparent arrival angle of −22.8 degrees, as described by Eq. (25). As the frequency bin leakage is decreased with larger FFTs, the desired solution is obtained. In figure 2.2, the MVDR beampattern* is plotted for the same vertical ULA and 25.0 Hz source for $f_{array} = 1, 2, 3, \ldots, 60$ Hz. An FFT size of 32 points was used to clearly illuminate the arc sin dependence of Eq. (2.5) (clearly $f_{array} \neq f_{source}$). Figure 2.3 plots the MVDR beampattern results for an FFT size of 1024 (with 50% overlap). The figure shows the correct frequency and

* The beampattern is defined in this report as the array's output energy ($\int P_y(f,\vec{u})\ df$) as the direction of look is varied. It is a real valued quantity.

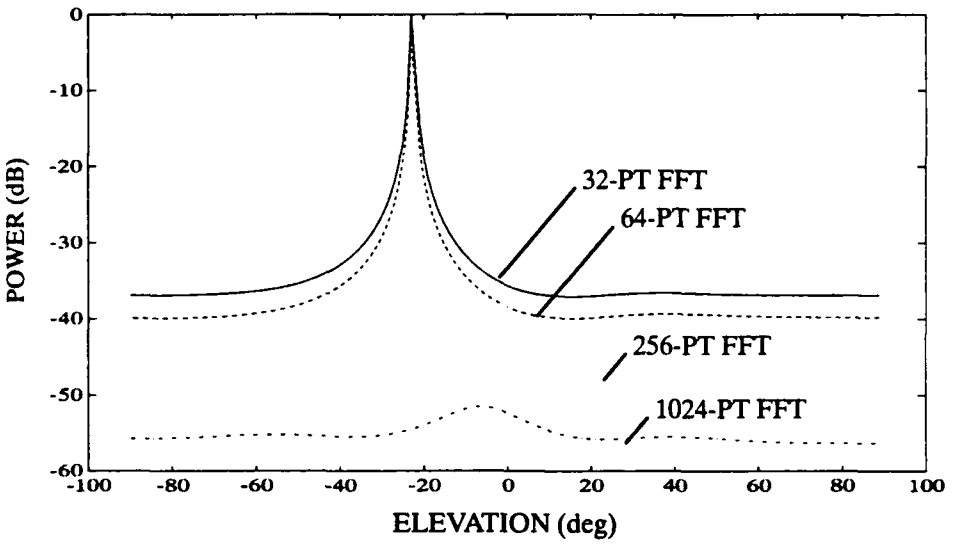


Figure 2.1. MVDR beampattern for single CW at 25.0 Hz/−14 deg; FFT sizes of 32, 64, 256, and 1024 points.

angle of the incident signal. Unless otherwise stated, a 1024-point FFT with 50% overlap will be used in all narrowband frequency-domain beamforming.

The second problem associated with broadband signals is spatial aliasing, which creates a directional ambiguity in the beampattern. Expressions for spatial aliasing are based on the sampling theorem NO TAG and are a function of the incident angle of the signals. The worst case for spatial aliasing in the ULA with element spacing d is when the incident angle is +/− 90 degrees (end-fire). Under these conditions, spatial aliasing will occur when

$$\frac{d}{c} <= \frac{1}{2}\frac{1}{f_{MAX}} \tag{2.6}$$

where f_{MAX} is the largest temporal frequency of the incident signals. The best case for spatial aliasing in the ULA with element spacing d is when the incident angle is 0 degrees (broadside), where spatial aliasing will occur whenever

$$\frac{d}{c} <= \frac{1}{f_{MAX}} \tag{2.7}$$

In the previous example, $d = 30$ m, $c = 1500$ m/s; we can expect the onset of spatial aliasing to appear when the frequency of the incident signal is between 25.0 Hz and 50.0 Hz, depending on the incident angle of the signal (see Appendix B for more discussion on grating lobes and time-domain conventional beamforming (TD-CBF) with a ULA). Spatial aliasing can be viewed clearly as grating lobes in the array pattern.* The grating lobes can be clearly seen in figure 2.4, where the array pattern of the 5-element vertical ULA is plotted for a steer direction

* The array pattern is defined in this report as the complex valued weighting given to an incident signal as its direction is varied when the array's direction of look is fixed. For equally spaced elements, it reduces to the spatial Fourier transform of the array weights.

of -14 degrees. Figure 2.4 is plotted from -180 degrees to $+180$ degrees for a complete view of the grating lobe structure. Note that because the array is linear along the Z axis, its array pattern is azimuthally independent. Thus, the array pattern actually forms a main cone and a grating cone when Eq. (2.6) is satisfied. It is straightforward to show that the true angle (versus that defined by the grating lobes) for $f > 1/2\, f_{MAX}$ lies between

$$\sin^{-1}\left[-\frac{1}{2}\frac{1}{f_{source}}\frac{c}{d}\right] \leq \phi_{array} \leq \sin^{-1}\left[\frac{1}{2}\frac{1}{f_{source}}\frac{c}{d}\right] \tag{2.8}$$

Figure 2.5 plots the MVDR beampattern for a broadband signal incident on the 5-element vertical ULA from -14 degrees. The grating lobes are readily apparent for $f > 40$ Hz.

The last problem associated with broadband signals is that the array pattern's main beam width is a function of frequency (as well as a function of the steer direction). This can also be seen in figure 2.4, where the main beam gets wider at lower frequencies, and thus the resolvability between two closely spaced sources becomes more difficult at lower frequencies. A closed-form approximation for the main beam width of a uniform linear array is found in Appendix B. Broadband beamformers which integrate across frequencies (either implicitly or explicitly) will have to take this issue into effect, especially if significant amounts of energy exist at the lower frequencies. The simplest method is to filter out the low frequencies (as well as the high frequencies for reducing spatial aliasing) to retain the required beam width for adequate resolution.

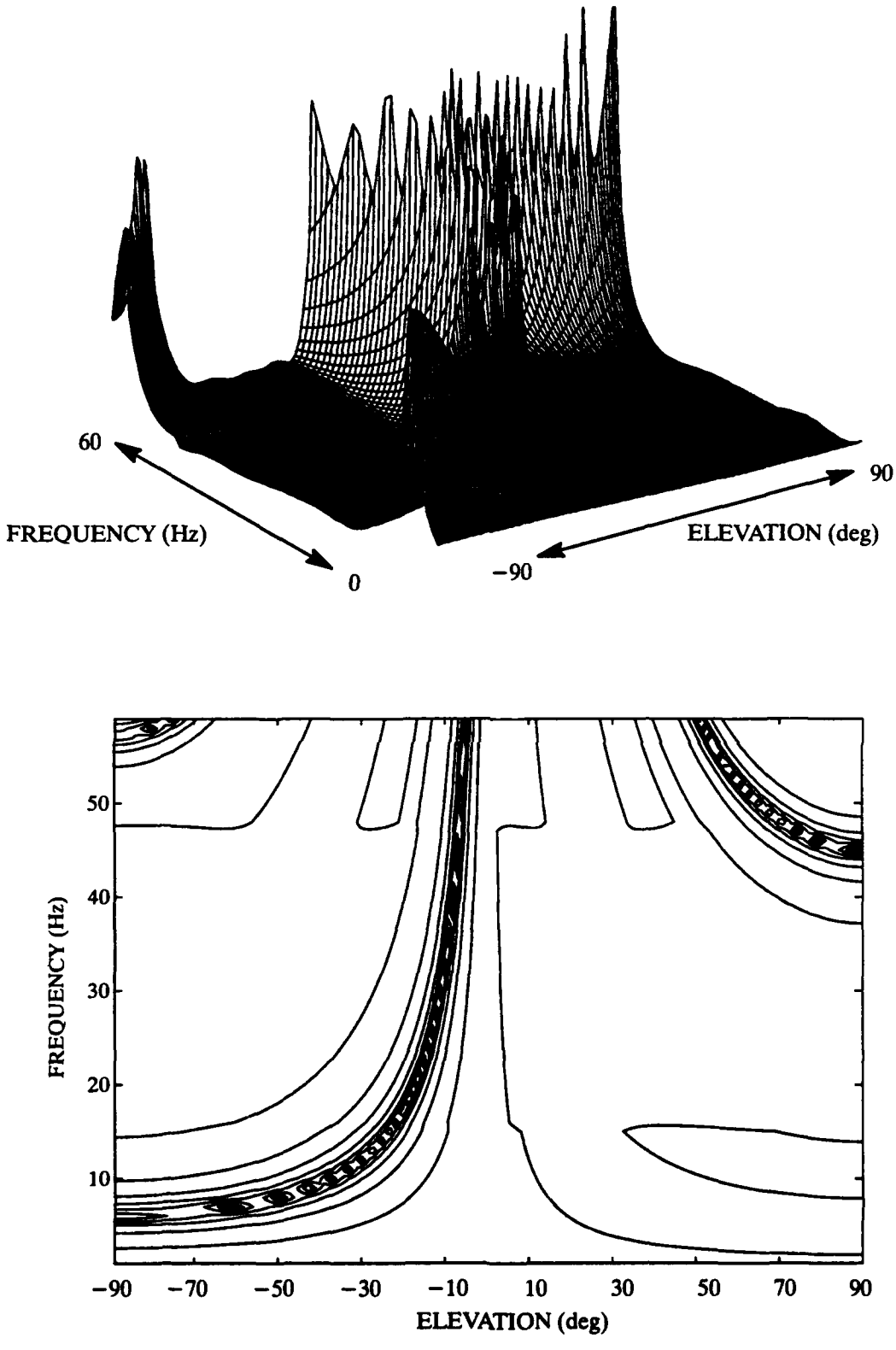


Figure 2.2. MVDR beampattern for single CW at 25.0 Hz/−14 deg; FFT size of 32 points.

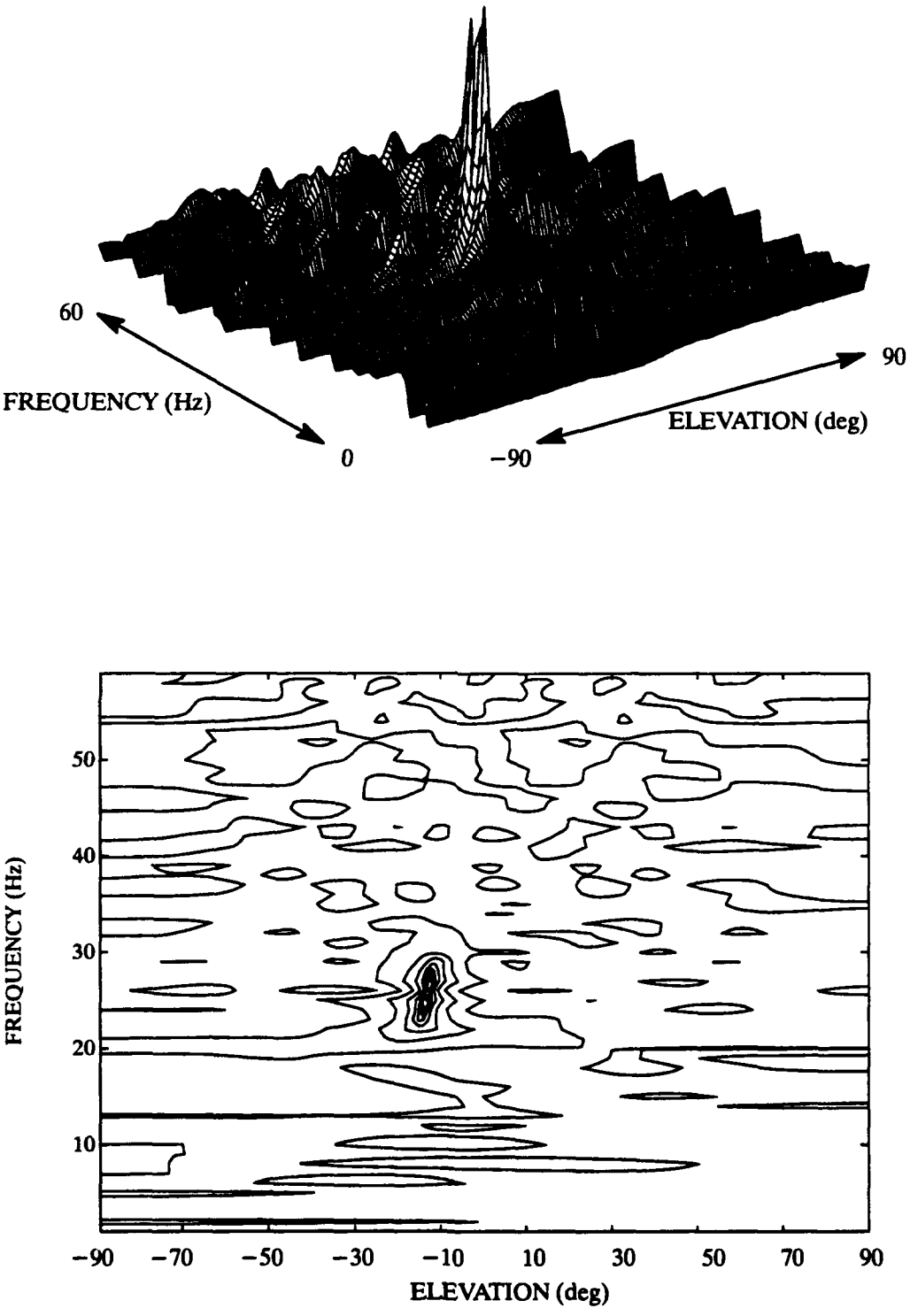


Figure 2.3. MVDR beampattern for single CW at 25.0 Hz/−14 deg; FFT size of 1024 points.

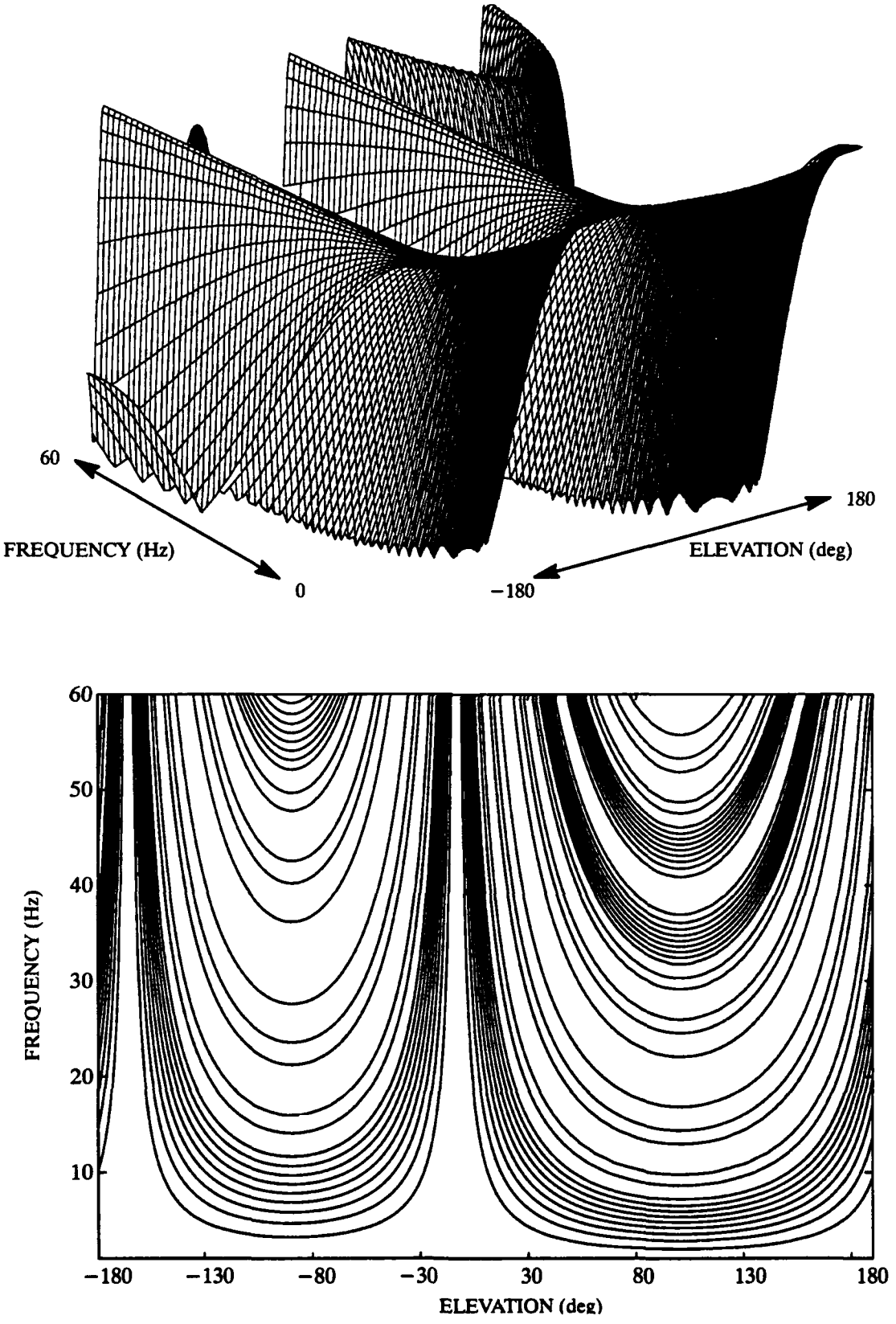


Figure 2.4. Array pattern for 5-element ULA pointed at −14 degrees.

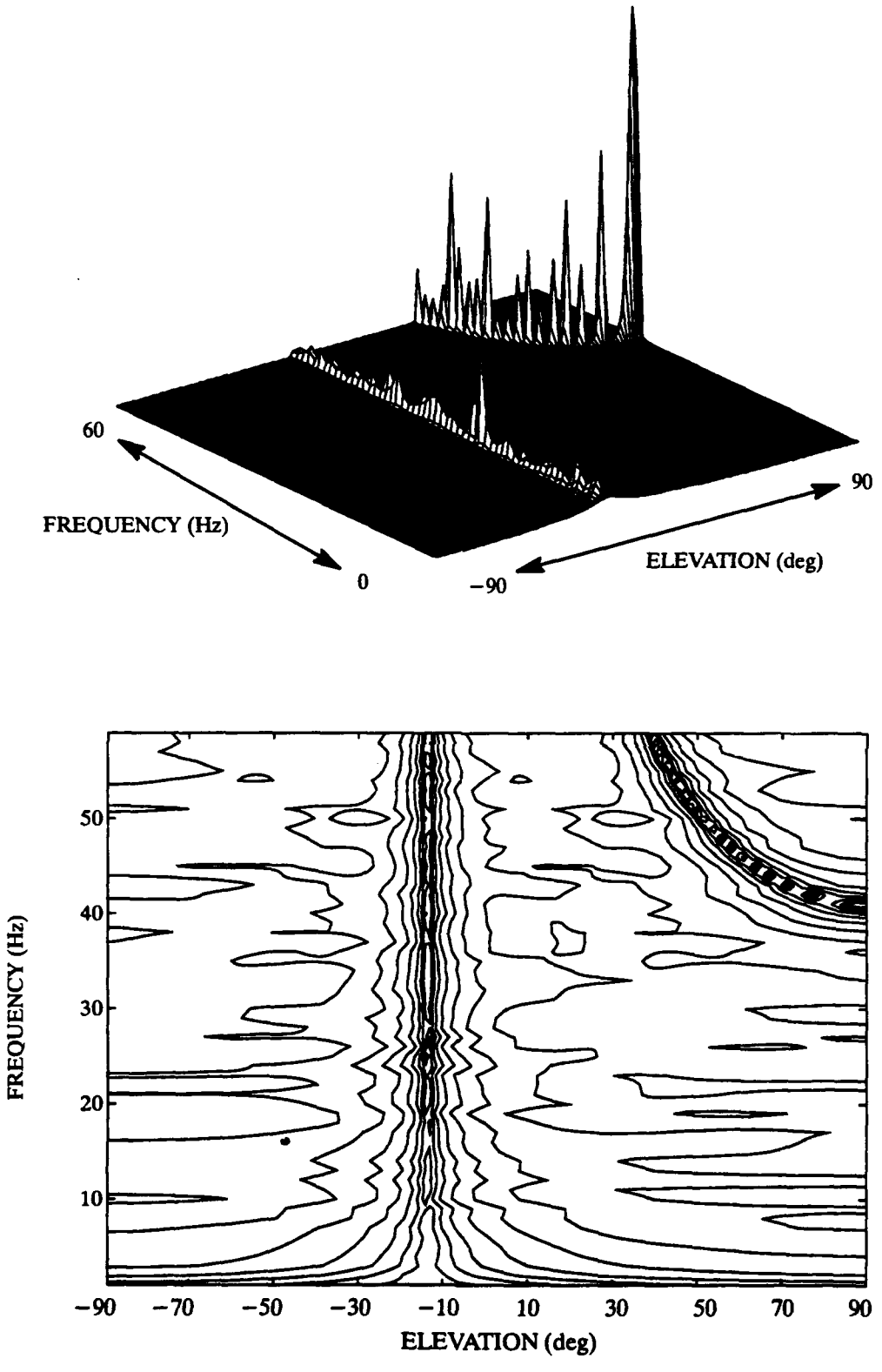


Figure 2.5. MVDR beampattern for single broadband signal incident at −14 deg; FFT size of 1024 points.

This section has introduced some of the effects of signal bandwidth on spatial processing. Insight here is crucial to understanding the beampattern output and the operation of various beamformers. Figures 2.6 and 2.7 show simplified adaptive arrays for narrowband signals and broadband signals, respectively [28]. A single complex weight is capable of supplying the required gain and phase adjustment at each element for narrowband signals, as seen in figure 2.6. The tapped-delay line in figure 2.7 permits the adjustment of the gain and phase at a number of different temporal frequencies, as required for the array processing of broadband signals. Signal preconditioning has been left off both figures for clarity.

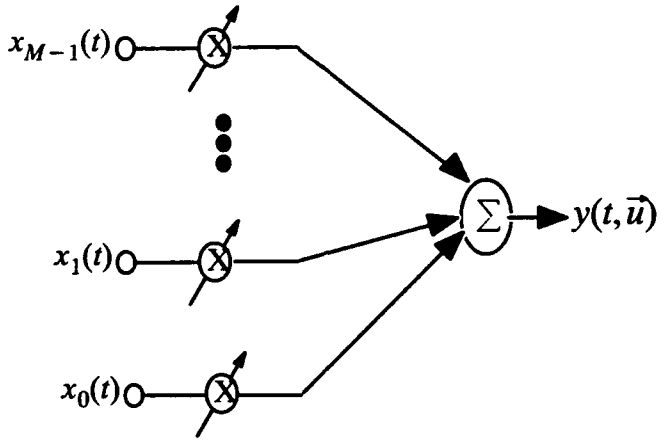


Figure 2.6. Adaptive array for narrowband signals (data and weights complex).

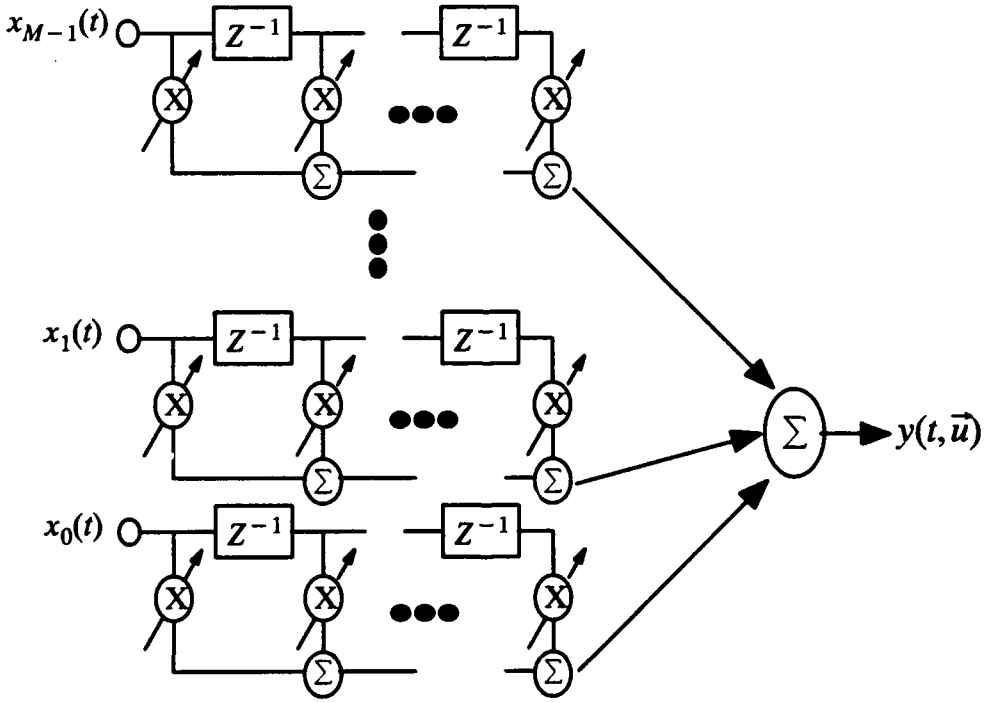


Figure 2.7. Adaptive array for broadband signals (data and weights complex).

2.3. SOURCE-TO-ELEMENT PROPAGATION CHANNEL

Two identities from ray theory can be used to generate a more complete picture of the source (including interferences) to element propagation channel in a nonhomogeneous medium. Consider figure 2.8 where an incident wave is colliding with a single abrupt discontinuity. We find that [7]

1. Incident angle is equal to the negative of the reflected angle

$$\phi_i = -\phi_r \tag{2.9}$$

2. Snell's Law:

$$\frac{cos(\phi_i)}{c_1} = \frac{cos(\phi_t)}{c_2} \tag{2.10}$$

where c_1 and c_2 are the speed of propagation of a plane wave in medium 1 and medium 2, respectively. From Eq. (2.10) it can be seen that there exists a critical incident angle where if $\phi_i \leq \phi_c$, no transmission results. The critical angle is

$$\phi_c = \cos^{-1}\left(\frac{c_1}{c_2}\right) \tag{2.11}$$

The ratio of the sound speeds in the two mediums, $\frac{c_1}{c_2}$, is called the index of refraction.

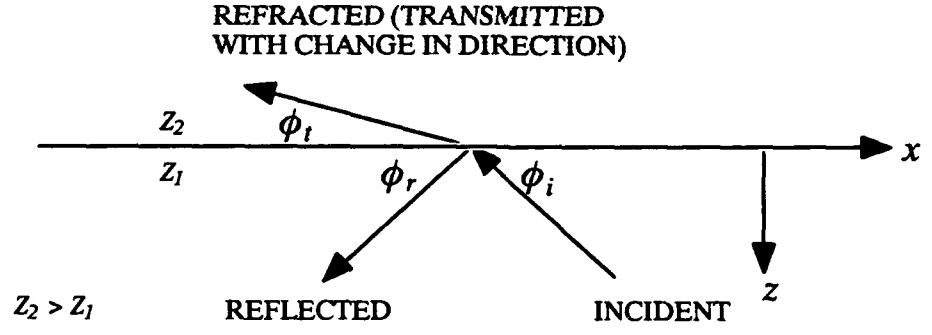


Figure 2.8. Effects of a single discontinuity in the propagation medium.

The magnitude of the reflected and transmitted waves depends on the characteristic acoustic impedance, $Z = \varrho\, c$, where ϱ is the medium density. For instance, it is straightforward to show that the magnitude of the reflected wave is related to the magnitude of the incident wave by [7]

$$A^r = A^i \left[\frac{Z_2\, sin(\phi_i) - Z_1\, sin(\phi_t)}{Z_2\, sin(\phi_i) + Z_1\, sin(\phi_t)}\right] \tag{2.12}$$

and that the magnitude of the transmitted wave is [7]

$$A^t = A^i \left[\frac{2\, Z_2\, sin(\phi_i)}{Z_2\, sin(\phi_i) + Z_1\, sin(\phi_t)}\right] \tag{2.13}$$

Using Eq. (2.9)−(2.13), we can compute the magnitude and direction of the reflected and transmitted waves. For example, assume that a source is 2000 m from a uniform linear vertical array and 250 m from an abrupt impedance boundary, with $Z_2 = 1.5/1.45\, Z_1 = 1.035\, Z_1$, as shown in figure 2.9. Table 2.1 summarizes the multipath and direct path data for various element depths. The data show quite clearly that the magnitude of the multipath, A^r, is different for many of the elements. For a large array, each element will most likely see a different propagation channel. In fact, it is possible for an element to see no signal, direct or multipath, from the source or to see only a transmitted multipath and no direct path (see, for instance, the top two elements in Z_2 in figure 2.9). Thus, for an algorithm to use the multipath information to increase the array gain, each signal-to-element channel must be treated individually.

The underlying difficulty with including the multipath data is that it is temporally delayed and deformed (due to nonideal propagation such as spreading, absorption, ducting and nonideal boundary interactions) as well as spatially different from the direct path. Most adaptive beamformers treat incident energy which is spatially different from the steer direction as interference to be nulled out. This is equivalent to making the implicit assumption of *all source-to-element propagation channels being equal and no multipaths existing.*

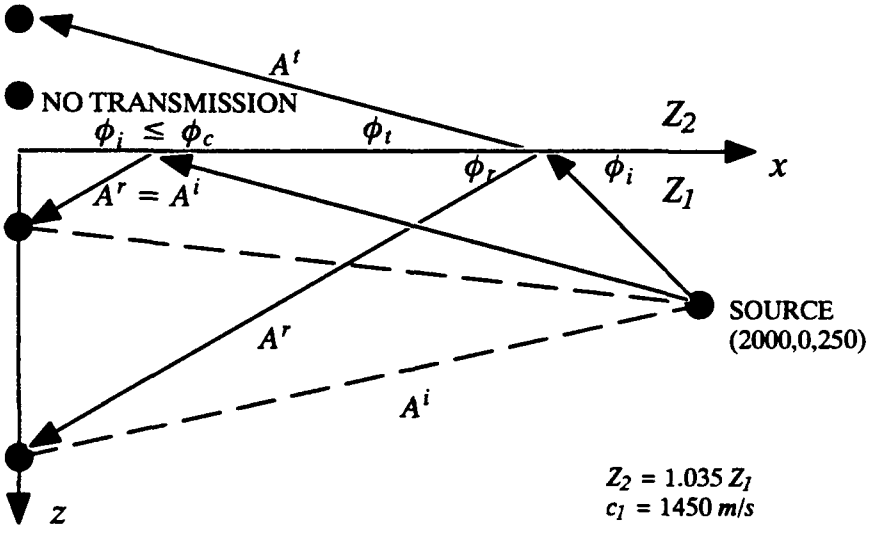


Figure 2.9. Effects of a single discontinuity in the propagation medium.

Table. 2.1. Direct and multipath data for the uniform linear vertical array in figure 2.9.

Element Depth (m)	A^r	ϕ_t (deg)	$\phi_r = \phi_i$ (deg)	x Reflect (m)	Direct Path (m)	Multi-Path (m)	Diff (ms)
190	1.0	–	12.4	863	2000.9	2047.8	32.3
220	1.0	–	13.2	936	2000.2	2054.5	37.4
250	1.0	–	14.1	1000	2000.0	2061.6	42.5
280	0.94	0.5	14.8	1056	2000.2	2069.6	48.0
310	0.52	5.0	15.6	1107	2000.9	2076.9	53.0
340	0.4	7.2	16.4	1153	2002.0	2085.2	58.7
370	0.33	8.9	17.2	1194	2003.6	2093.9	64.8

2.4. SOURCE-TO-ELEMENT PROPAGATION CHANNEL MODEL

Each source-to-element propagation channel can be modeled as a FIR filter with impulse response

$$ih_m(n) = \delta(n - n_m^{direct}) + \sum_{i=1}^{I} A_{m,i}^r \, \delta(n - n_{m,i}^{multi}) \tag{2.14}$$

for the m^{th} element and I multipaths [12]. By aligning the received element data to point towards the direct path, we have a channel impulse response

$$\begin{aligned} h_m(n) &= \delta(n) + \sum_{i=1}^{I} A_{m,i}^r \, \delta(n - n_{m,i}^{multi} + n_m^{direct}) \\ &= \delta(n) + \sum_{i=1}^{I} A_{m,i}^r \, \delta(n - n_{m,i}^0) \end{aligned} \tag{2.15}$$

We can take the Z transform of Eq. (2.15) for a single multipath (I=1) to find the channel transfer function between the source and the m^{th} element

$$H_m(z) = \sum_{n=0}^{\infty} h_m(n) \, z^{-n} = 1 + A_m^r \, z^{-n_m^0} \tag{2.16}$$

The power spectral density of the m^{th} element is now related to the power spectral density of the source by

$$PSD_{x_m}(f) = |H_m(f)|^2 \, PSD_s(f) \tag{2.17}$$

where

$$|H_m(f)|^2 = 1 + (A_m^r)^2 + 2 A_m^r \cos(2 \pi f n_m^0) \tag{2.18}$$

The spectral nulls in Eq. (2.18) are characteristic of a multipath channel and can be seen in figure 2.10 for $A_m{}^r = 0.8$, and $n_m{}^0 = 5$. It is seen from Eq. (2.18) that the depth of the spectral

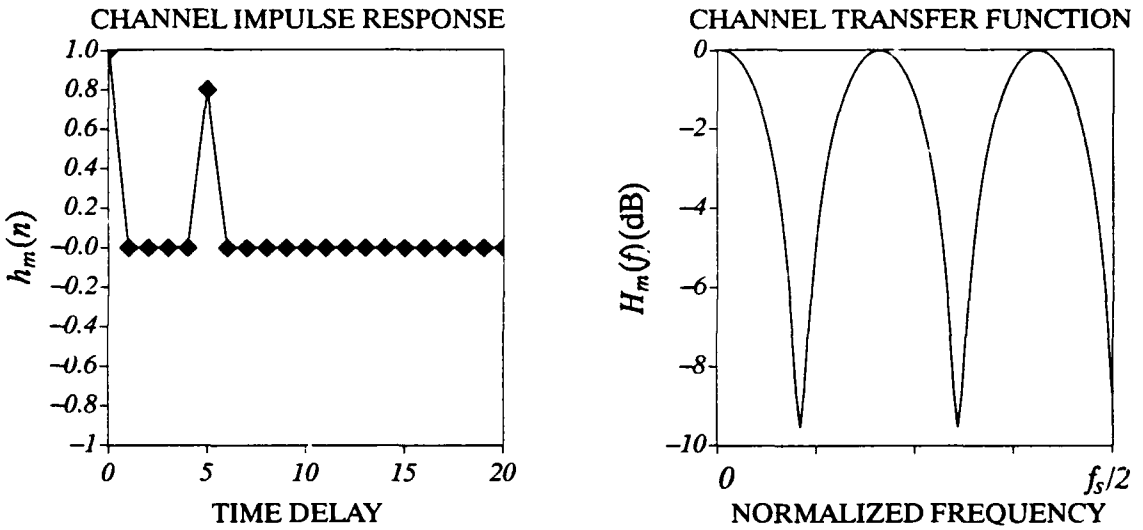


Figure 2.10. Channel impulse response and transfer function.

null in the channel's transfer function increases with the amplitude of the multipath ($A_m{}^r$) and that the distance between spectral nulls decreases as the delay ($n_m{}^0$) between the direct and multipaths increases.

2.5. SOURCE-TO-ELEMENT PROPAGATION CHANNEL EQUALIZATION

Even though the multipath can be distorted by the propagation channel and arrives at the array from a direction other than that of the direct path, it contains energy coherent with the signal of interest. If each source-to-element channel can be equalized, i.e., the effects of the multipath are negated, then the SINR of each element and thus the array can be enhanced. (It should be noted that two broadband signals are fully correlated if they are amplitude-scaled and temporally delayed replicas, while two narrowband signals are fully correlated if they have a fixed phase difference relative to each other. Multipath signals are usually partially correlated to the direct path due to nonideal reflections or refractions.)

Continuing under the premise that the inverse filter can be found, we can find that the transfer function of the ideal equalizer or inverse filter is

$$H_m^{inv}(z) = \frac{1}{H_m(z)} \tag{2.19}$$

For the single multipath example above

$$H_m^{inv}(z) = \frac{1}{1 + A_m^r\, z^{-n_m^0}} = \sum_{i=0}^{\infty} (-A_m^r)^i\, z^{-n_m^0 i} \tag{2.20}$$

using the geometric series. Thus, the ideal inverse filter impulse response is

$$\begin{aligned} h_m^{inv}(n) &= (-A_m^r)^{i/n_m^0} \quad \text{for } i = 0,\ n_m^0,\ 2n_m^0, \ldots \\ h_m^{inv}(n) &= 0 \qquad\qquad\qquad\quad \text{otherwise} \end{aligned} \tag{2.21}$$

Note that $h_m{}^{inv}(n)$ is stable if and only if $|A_m{}^r| < 1$. Figure 2.11 plots the first 20 taps of the ideal inverse channel impulse response (Eq. (2.21)) and the ideal channel transfer function. It is apparent from comparing figures 2.10 and 2.11 that the cascaded channel–inverse filter results in a distortionless transmission albeit delayed in time. This is the aim of channel equalization.

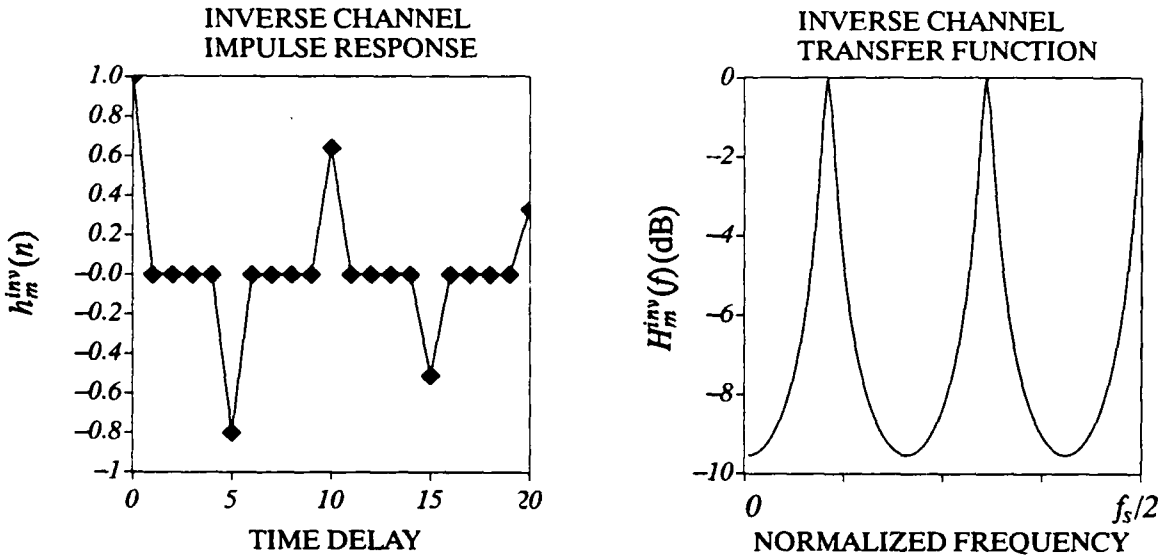


Figure 2.11. Inverse channel impulse response and transfer function.

3. TIME-DOMAIN BEAMFORMING

Time-domain beamforming techniques are used in communication, in transient signal analysis, or for wideband sources. This section briefly introduces time-domain beamforming and its discrete implementation. It introduces notation and concepts used later in this text.

3.1. CONTINUOUS TIME-DOMAIN CBF

To coherently add the outputs of all elements in the time-domain requires the proper delay of each element before summing. The delay is determined by the desired "steer" (or "look") direction. The summation can be written as

$$y(t,\vec{u}) = \sum_{m=0}^{M-1} w_m^* \, x_m(t - \tau_m(\vec{u})) \tag{3.1}$$

where

$\vec{u}$ = steering directional unit vector for the $(\theta,\phi)^{th}$ direction

$\tau_m(\vec{u}) = \frac{\vec{x}_m \cdot \vec{u}}{c}$ =required steering delay at the $x_m(t)$ element to look at direction $\vec{u}$

w_m = weight (or shading coefficient) for the $x_m(t)$ element

Figure 3.1 is a direct implementation of Eq. (3.1). It represents pure spatial processing, since the temporal delay operation has a distortionless transfer function (constant amplitude and linear phase).

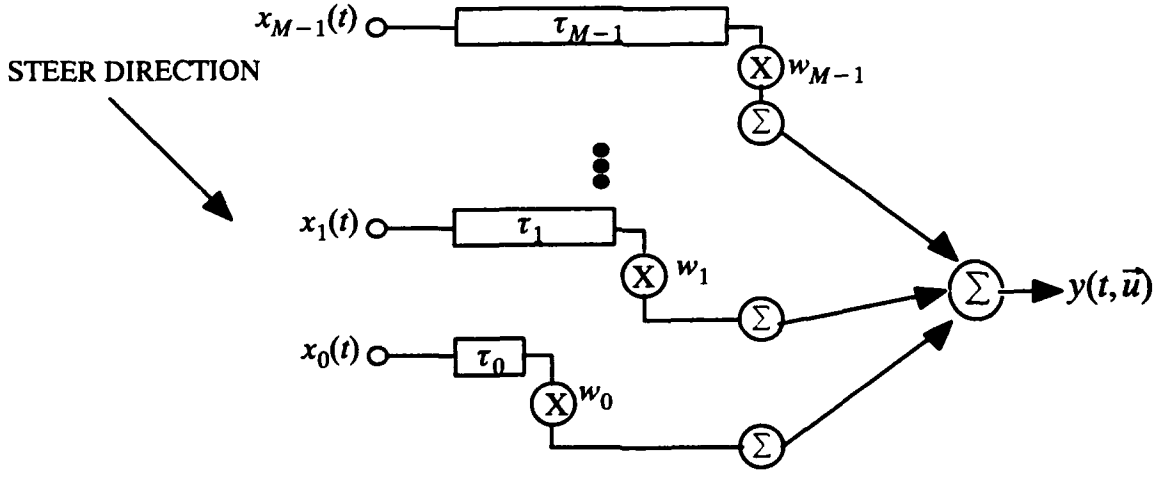


Figure 3.1. Continuous time time-domain conventional beamforming.

The instantaneous power in the beamformer output, $P_y(t,\vec{u}) = |y(t,\vec{u})|^2$, will display a peak whenever the steer direction equals a source direction. (Recall that $|y(t,\vec{u})|^2 = y(t,\vec{u})\, y^*(t,\vec{u})$, where $y^*(t,\vec{u})$ is the complex conjugate of $y(t,\vec{u})$.) In low SNR

environments, $P_y(t,\vec{u})$ must be time-averaged to adequately resolve a source direction. Since Eq. (3.1) is a summation of all energy coming from a particular steer direction, it can be used to detect broadband sources as well as narrowband sources. To classify the source, the spectrum of the beamformer output must be analyzed.

3.2. DISCRETE IMPLEMENTATION OF TIME-DOMAIN CBF

To implement any algorithm in hardware requires that the continuous input data be sampled so that $t \rightarrow k\,\Delta t$, where k is a positive integer and $\Delta t = \frac{1}{f_s}$ is the sampling interval. Sampling Eq. (3.1) and normalizing each iteration to the sampling interval Δt, we find the beamformer output can be written as

$$y(k,\vec{u}) = \boldsymbol{W}^H \boldsymbol{X}^a(k,\vec{u}) \tag{3.2}$$

where $\boldsymbol{W}$ is the Mx1 weight vector

$$\boldsymbol{W}^T = \left[w_0, w_1, w_2, ..., w_{M-1}\right] \tag{3.3}$$

and $\boldsymbol{X}^a(k,\vec{u})$ is the Mx1 perfectly aligned data element vector (or snapshot) at time k

$$\begin{aligned}\boldsymbol{X}^a(k,\vec{u})^T &= \left[x_0(k-\tau_0(\vec{u})),\ x_1(k-\tau_1(\vec{u})),\ x_2(k-\tau_2(\vec{u})),\ ...,\ x_{M-1}(k-\tau_{M-1}(\vec{u}))\right] \\ &= \left[x_0^a(k),\ x_1^a(k),\ x_2^a(k),\ ...,\ x_{M-1}^a(k)\right]\end{aligned} \tag{3.4}$$

The alignment of the element data essentially creates a broadside steer direction. With these vectors defined, the instantaneous power in the beamformer output can be rewritten as

$$P_y^i(k,\vec{u}) = y(k,\vec{u})\ y^*(k,\vec{u}) = \boldsymbol{W}^H \boldsymbol{X}^a(k,\vec{u})\ \boldsymbol{X}^a(k,\vec{u})^H\ \boldsymbol{W} \tag{3.5}$$

Figure 3.2 shows the new TD-CBF structure and notation. From the figure it can be seen that *time-domain CBF is very simple to implement. Simply retain a time-history of each element's* input data, and then sum the properly delayed data from each element in the array. The average power output is found by taking the expectation of both sides of Eq. (3.5) to give

$$P_y(\vec{u}) = \boldsymbol{W}^H \boldsymbol{R}(\vec{u})\ \boldsymbol{W} \tag{3.6}$$

where

$$\boldsymbol{R}(\vec{u}) = E\left[\ \boldsymbol{X}^a(k,\vec{u})\ \boldsymbol{X}^a(k,\vec{u})^H\ \right] \tag{3.7}$$

is the spatial temporal correlation matrix [9] (also called the steered covariance matrix in [41]). If the input process is ergodic, then a time average will approach Eq. (3.6) (see, for example, Ch. 2 of [14]).

Unfortunately, the element delays are rounded to the nearest integer in the discrete implementation. Thus, the discrete time element delays are given by

$$\tau_m(\vec{u}) = round\left[\frac{\vec{x}_m \cdot (\vec{s}_l - \vec{u})}{c} f_s\right] \tag{3.8}$$

where *round*[x] rounds x to the nearest integer. (The round operation is inherent in the sampling process.) It is apparent from Eq. (3.8) that the steering delays will be rounded to integer multiples of the sampling period. Thus, the higher the sampling frequency, the more steer directions are possible and the better the delineation of the array beam pattern.

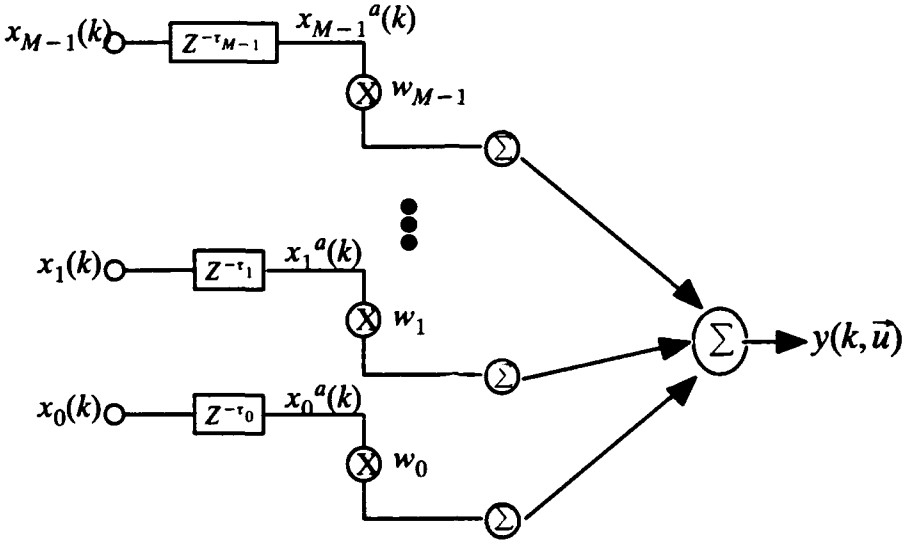


Figure 3.2. Discrete time time-domain conventional beamforming.

Often interpolation is required to obtain the desired delineation of the array pattern. Two possible techniques for data interpolation are (1) conventional time series interpolation, which effectively resamples each element's data at a higher sampling rate [9] and (2) data interpolation by a linear 2-tap FIR filter or a quadratic 3-tap FIR filter to interpolate between data values while retaining the same sampling rate (see, for example, Ch. 3.1 of [4]).

Figure 3.3 shows the most intuitive means of implementing conventional interpolation, called low-pass prebeamforming interpolation [26]. Each element is resampled at a higher sampling frequency, allowing for more precise control over the steering delays. Enough data memory from each element must be retained to account for +/− maximum steer delay (in number of samples), which is a function of the resampled sampling frequency, the speed of propagation, the element location (with respect to the origin), and the steer direction. When the number of steer directions is greater than the number of array elements, a computationally more efficient method of interpolation can be used, called low-pass postbeamforming interpolation [26]. The interpolation operation can be separated into two operations: (1) $I-1$

zeros added between each successive data sample and (2) interpolation filtering. I determines the amount of interpolation and is assumed to be an integer greater than 1. In postbeamforming interpolation, the zero adding is done before the delay and sum beamforming computation, while the interpolation filtering is performed after the delay and sum beamforming. Both techniques result in similar improvements in determining the angle of arrival of the source.

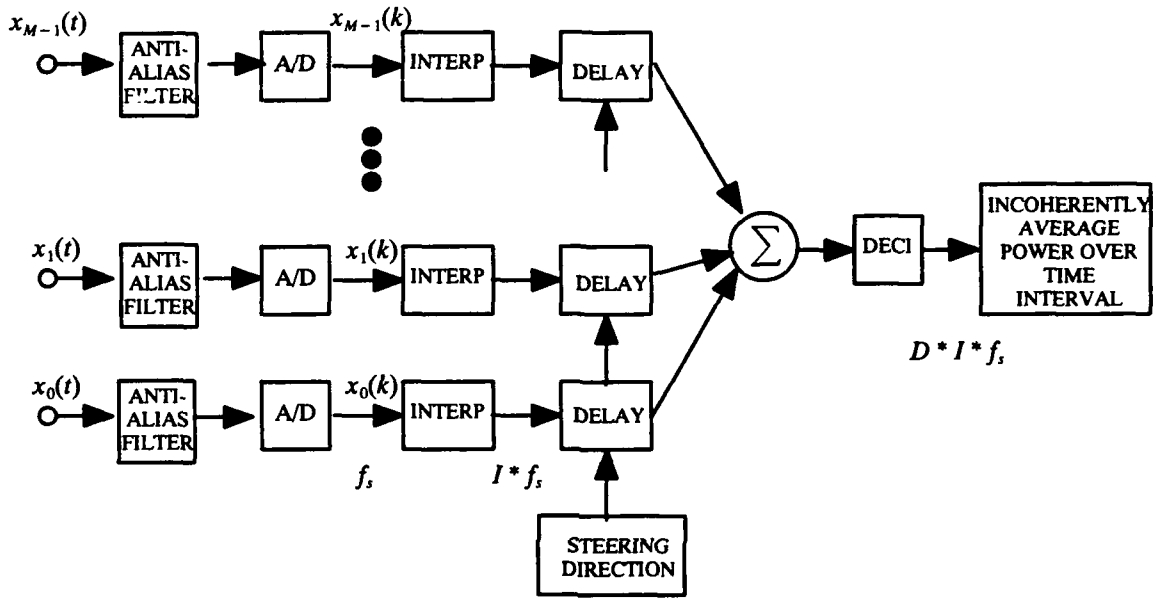


Figure 3.3. Prebeamforming-interpolation time-domain beamforming.

A second method of data interpolation which does not increase the data rate is shown in figure 3.4 for a single element. First, the rounded-down integer delay is used for coarse data alignment. Then a small FIR interpolation filter (either 2 taps for a linear interpolation or 3 taps for a quadratic interpolation) is used to compensate for the difference between the actual delay and the rounded-down integer delay. Essentially, the interpolation filter provides for fine steering. In most cases, the generation of the aligned data matrix will require some method of data interpolation for adequate beampattern delineation.

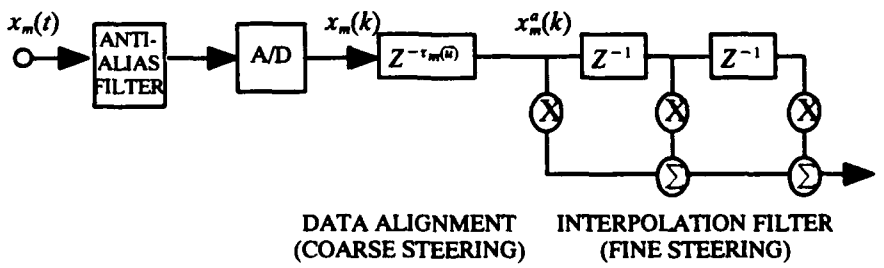


Figure 3.4. Data interpolation of a single element for time-domain beamforming.

4. FREQUENCY-DOMAIN BEAMFORMING

Frequency-domain spatial processing is an especially efficient technique when the stochastic signals are narrowband and sustained rather than transient [9]. This section briefly introduces two of the more popular frequency-domain beamformers, Bartlett and MVDR. References are provided for more detailed information.

4.1. FREQUENCY-DOMAIN CBF (BARTLETT)

The array output can be analyzed in the frequency domain as well as the time domain. In the frequency domain, time-delays transform into frequency-dependent phase shifts. Thus, taking the Fourier Transform of Eq. (3.1), we obtain

$$\mathcal{F}\{y(t,\vec{u})\} = \mathcal{F}\left\{\sum_{m=0}^{M-1} w_m^* \, x_m(t - \tau_m(\vec{u}))\right\}$$

$$Y(f,\vec{u}) = \sum_{m=0}^{M-1} w_m^* \, e^{-j2\pi f \tau_m(\vec{u})} \, X_m(f) \tag{4.1}$$

where $X_m(f)$ is the single element (also called along-channel) temporal Discrete Fourier Transform of $x_m(t)$ in a discrete implementation. Eq. (4.1) can be rewritten in terms of linear matrixes

$$Y(f,\vec{u}) = \boldsymbol{E}(f,\vec{u})^H \, \boldsymbol{W}^H \, \boldsymbol{X}(f) \tag{4.2}$$

where

$$\boldsymbol{E}(f,\vec{u})^T = \left[e^{+j2\pi f \tau_0(\vec{u})}, \; e^{+j2\pi f \tau_1(\vec{u})}, \; \ldots, \; e^{+j2\pi f \tau_{M-1}(\vec{u})}\right] \tag{4.3}$$

$$\boldsymbol{W}^T = \text{diag}\left[w_0, \; w_1, \; \ldots, \; w_{M-1}\right] \tag{4.4}$$

$$\boldsymbol{X}(f)^T = \left[X_0(f), \; X_1(f), \; \ldots, \; X_{N-1}(f)\right] \tag{4.5}$$

$\boldsymbol{E}(f,\bar{u})$ is the plane-wave steering vector which corrects for the phase delays at frequency f between elements when the array is steered in direction $\bar{u}$. The FFTs are usually performed with a 50%−75% overlap, weighted with a good FFT window (e.g., the Hanning window), and are of an adequate length (e.g., 1024−8192) for good frequency resolution NO TAG. Padding the along-channel data with zeros for good frequency resolution is equivalent to the data interpolation discussed in the previous section. $\boldsymbol{W}^T$ is a diagonal matrix containing the element weights (shading coefficients). For unity shading, $\boldsymbol{W}^T$ reduces to the identity and can be dropped out of the equations. The instantaneous power in the array output is given by

$$P_y^i(f,\vec{u}) = Y(f,\vec{u}) \; Y^*(f,\vec{u}) = \boldsymbol{E}(f,\vec{u})^H \, \boldsymbol{W}^H \, \boldsymbol{X}(f) \, \boldsymbol{X}(f)^H \, \boldsymbol{W} \, \boldsymbol{E}(f,\vec{u}) \tag{4.6}$$

Or taking the expected value of both sides, we find the average power output as

$$P_y(f,\vec{u}) = E(f,\vec{u})^H \; W^H \; R(f) \; W \; E(f,\vec{u}) \tag{4.7}$$

where

$$R(f) = E\left[\; X(f) \; X(f)^H \; \right]$$

is the spatial spectral correlation matrix NO TAG (also called the cross-spectral density matrix [6]). In practice, only an estimate of $R(f)$ can be computed:

$$R(f) \approx \hat{R}(f) = \frac{1}{N} \sum_{i=1}^{N} X(f) \; X(f)^H \tag{4.8}$$

The efficiency of Eq. (4.7) is that once an estimate of $R(f)$ is formed, any new direction can be searched by simply generating a new plane-wave steering vector, $E(f)$. For broadband signals, Eq. (4.7) is computed for multiple frequencies within the band of interest and then summed across frequency.

4.2. MVDR

MVDR is an example of a linearly constrained adaptive beamforming algorithm. It has the ability to adapt to the spatial interferer environment and place spatial nulls wherever strong interferers exist. The MVDR method determines the weight vector, $W(f,\vec{u})$, which minimizes the average array output power:

$$P_y(f,\vec{u}) = W(f,\vec{u})^H \; R(f) \; W(f,\vec{u}) \tag{4.9}$$

subject to the constraint that the steer direction has distortionless response (constant amplitude and linear phase), i.e., $W(f,\vec{u})^H \; E(f,\vec{u}) = 1$. It was shown in [9] (see also [31]–[40] for more on MVDR) that the array element weight vector satisfying these constraints is

$$W(f,\vec{u}) = \frac{R(f)^{-1} \; E(f,\vec{u})}{E(f,\vec{u})^H \; R(f)^{-1} \; E(f,\vec{u})} \tag{4.10}$$

Using Eq. (4.10) in Eq. (4.9), we find the average output power to be

$$P_y(f,\vec{u}) = \frac{1}{E(f,\vec{u})^H \; R(f)^{-1} \; E(f,\vec{u})} \tag{4.11}$$

Once an estimate of $R(f)$ is formed, its inverse must be computed. This requires $O(M^3)$ operations in general and can be a numerically unstable operation if the condition number of $R(f)$ is large. However, once this inverse is computed, any direction can be searched by just generating a new plane-wave steering vector, $E(f)$. Since the spectra of MVDR are constrained in the direction of look and because of the narrowband processing, MVDR is not capable of channel equalization.

5. BROADBAND BEAMFORMING INCORPORATING ADAPTIVE EQUALIZATION

This section discusses adaptive beamforming algorithms capable of channel equalization. The complexity of the underlying problem is depicted in figure 5.1 and is described in more depth in [13] and [15]. Each source signal is convolved with its local source impulse response (transducer, ship hull, etc.), $S_i(z)$, before being transmitted into the propagation medium. As described in Sections 2.4. and 2.5, each source-to-element is modeled as a separate transmission channel (which includes multipaths, distortion, propagation losses, etc.), $H_{i,j}(z)$. Upon reception at each array element, the receiving data undergo another convolution with the element's impulse response (transducer), $E_j(z)$. To simplify the problem, it will be assumed that each element is equalized and that the problem is to equalize the channel such that $s_i(t)$ is recovered. The channel models are assumed to be linear and slowly time varying.

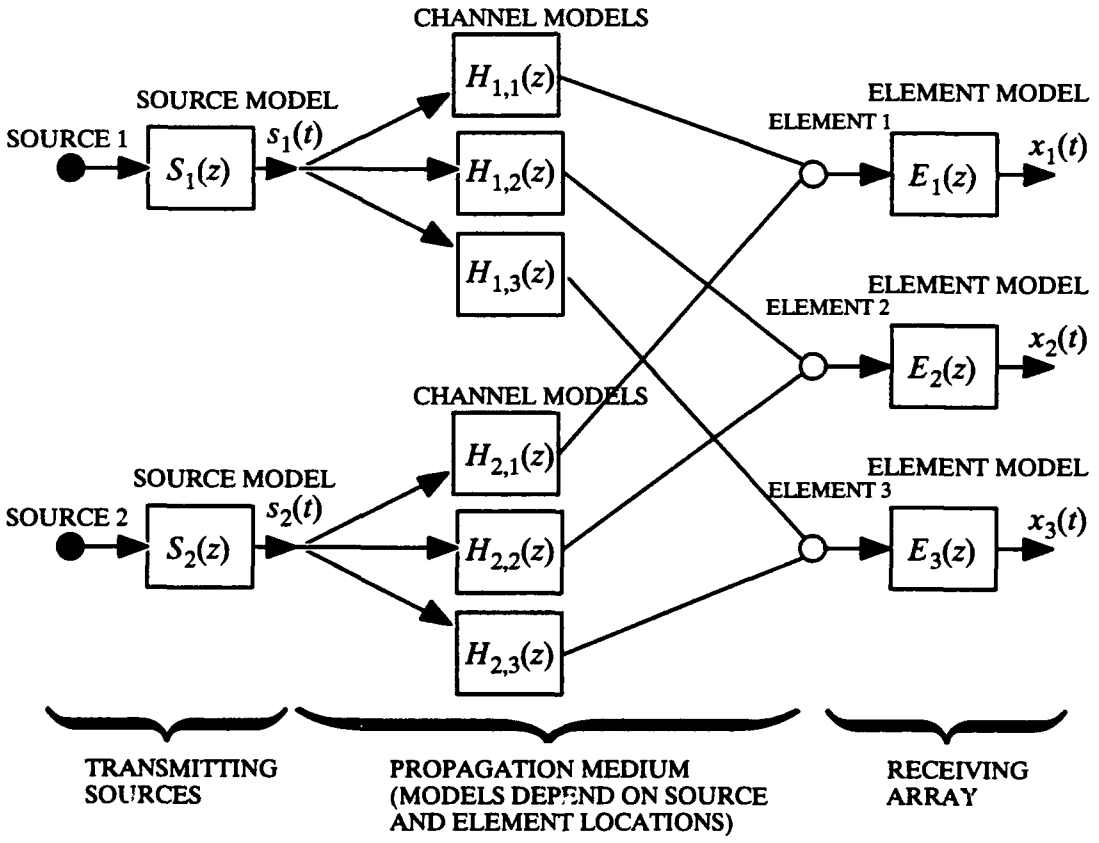


Figure 5.1. Channel models for two sources and three elements.

5.1. MULTICHANNEL ADAPTIVE FILTERING

Two of the earliest detailed descriptions of the multichannel adaptive filter for use in adaptive arrays can be found in [28] and [23]. The multichannel adaptive filter, shown in figure 5.2, adapts all of its weights to minimize the mean square error between the multichannel filter output and a desired signal. Figure 5.2 is an adaptive filter which can adapt to both the temporal and spatial characteristics of the input data and is thus capable of channel equalization and interferer cancellation. The weaknesses of this multichannel adaptive filter are that (1) source localization is *not* available from the filtered output as was true with the previous beamformers (although localization information might by using the converged weights values similar to the techniques used by frequency domain linear prediction beamforming [6] and [9]), and (2) it requires a desired signal, d_i or at least an approximate desired signal. Although these characteristics might eliminate the multichannel adaptive filter from applications in passive surveillance, a desired sequence does exist in communication systems (via a training sequence) and active surveillance (transmitted pulse).

To investigate the multichannel adaptive filter, consider the i^{th} (single) channel with $N = N_1 + N_2 + 1$ taps. The time-domain filtered output is given at time k by

$$y_i(k,\vec{u}) = W_i(k)^H X_i^a(k) \tag{5.1}$$

where

$$W_i(k)^T = \left[w_{i,-N_2}(k)\ ,\ \ldots\ ,\ w_{i,0}(k)\ ,\ \ldots\ ,\ w_{i,N_1}(k)\right] \quad (1\text{x}N) \text{ weight vector} \tag{5.2}$$

$$X_i^a(k)^T = \left[x_i^a(k+N_2)\ ,\ \ldots\ ,\ x_i^a(k)\ ,\ \ldots\ ,\ x_i^a(k-N_1)\right] \quad (1\text{x}N) \text{ aligned data vector} \tag{5.3}$$

The M-element multichannel filtered output is the sum of the M single-channel filtered outputs, or in matrix formulation:

$$\begin{aligned} y(k,\vec{u}) &= \sum_{i=0}^{M-1} y_i(k,\vec{u}) \\ &= W(k)^H X^a(k) \end{aligned} \tag{5.4}$$

where

$$W(k)^T = \left[W_0(k)^T\ ,\ W_1(k)^T\ ,\ \ldots\ ,\ W_{M-1}(k)^T\right] \quad (1\text{x}MN) \text{ vector} \tag{5.5}$$

$$X^a(k)^T = \left[X_0^a(k)^T\ ,\ X_1^a(k)^T\ ,\ \ldots\ ,\ X_{M-1}^a(k)^T\right] \quad (1\text{x}MN) \text{ vector} \tag{5.6}$$

Written in this manner, it is easy to see that the optimum Wiener–Hopf solution which minimizes the MSE, $E[e(k)e^*(k)]$, where

$$e(k) = d(k) - y(k,\vec{u}) \tag{5.7}$$

is found in [28] to be

$$W_{opt} = R(\vec{u})^{-1} P(\vec{u}) \tag{5.8}$$

or in [14] for detailed single-channel adaptive filter theory. In Eq. (5.8), $\boldsymbol{R}(\vec{u})$ is the multi-channel, spatial temporal correlation matrix and $\boldsymbol{P}(\vec{u})$ is the multichannel, cross-correlation matrix given by

$$\boldsymbol{R}(\vec{u}) = E\left[\, \boldsymbol{X}^a(k)\, \boldsymbol{X}^a(k)^H \,\right] \qquad (MN\text{x}MN) \text{ vector} \qquad (5.9)$$

$$\boldsymbol{P}(\vec{u}) = E\left[\, \boldsymbol{X}^a(k)\, d^*(k) \,\right] \qquad (MN\text{x}1) \text{ vector} \qquad (5.10)$$

Numerous methods of solving Eq. (5.8) without actually inverting the $MNxMN$ matrix (which requires $O(M^3N^3)$ operations in general) have been explored in the past 30 years. The recursive least squares (RLS) solution (also called Kalman Filter) requires $O(M^2N^2)$ operations [22]. While the shifting property does not exist between channels as would be required for (super) fast RLS solutions (i.e., $O(MN)$), it does exist within a single channel, as shown in figure 5.2. This property has been exploited by [19] and [25] to derive two different multichannel RLS lattice algorithms with $O(M^2N)$. The application of stochastic solutions like least mean squares (LMS) to the problem does not require matrix inversions. This results in $O(MN)$ algorithms, but the performance of these algorithms is dependent on the data [28].

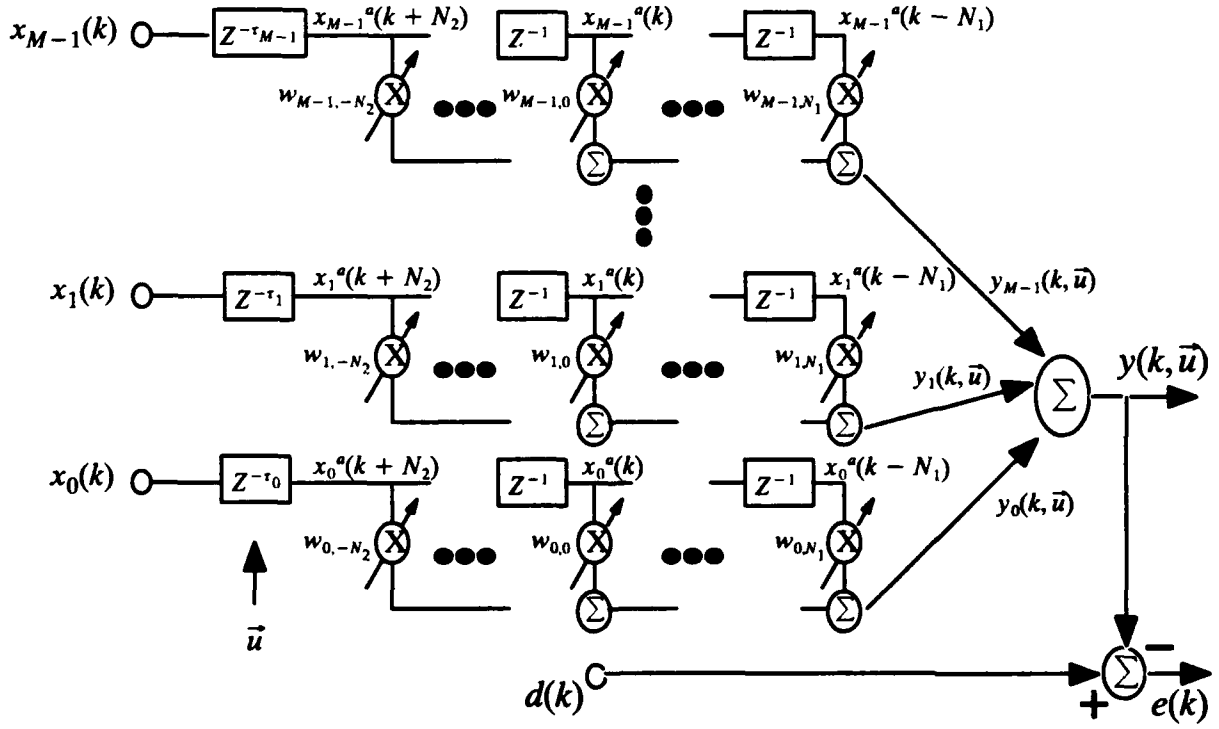


Figure 5.2. Multichannel adaptive filter for broadband signals.

The data alignment helps to minimize the required length of the single-channel FIR filter, N. By interpreting Eq. (3.8), we see that the length of the filter N (as well as other parameters) determines the possible steering delays, which in turn determine the angles at

which the multipath can arrive and still be equalized. Thus, by aligning the data toward a particular direction, a range of angles is defined for which the filter can equalize multipaths. For example, the uniform linear array with M elements can equalize multipaths as long as they arrive within $-\sin^{-1}\left[\frac{N_2}{2}\frac{c}{d}\frac{1}{M-1}\frac{1}{f_s}\right] < \phi < +\sin^{-1}\left[\frac{N_1}{2}\frac{c}{d}\frac{1}{M-1}\frac{1}{f_s}\right]$ degrees of the steered direction, $\vec{u}$.

However, since the adaptation is based on minimizing the output error power, if the same desired sequence is used for all steer directions, the weights will always attempt to equalize the same signal (as long as it is in the range of possible angles defined by N above). The generalized sidelobe canceller to be discussed in the next section will modify figure 5.2 to use a TD-CBF to generate the desired signal. It is dependent on the steer direction and will thus provide more directionality information.

5.2. GENERALIZED SIDELOBE CANCELLER

The generalized sidelobe canceller (GSC) was developed in [24] where it was shown that the linear constrained adaptive beamformer (like the Frost algorithm [21], MVDR, and [20] and [29]) can be implemented as a combination of a nonadaptive beamformer and an unconstrained adaptive beamformer. Figure 5.3 shows a block diagram of the GSC, with the spatial filters are derived from an array of elements. The GSC is essentially a spatial version of the single-channel adaptive noise canceller. The main array generates the desired signal (determined by its steer direction), which can be corrupted by interferers that bleed in through the sidelobes of the nonadaptive main array. Each auxiliary array spatially filters one interferer (determined by the auxiliary array's steer direction); then its output is adaptively weighted and subtracted from the main array output, resulting in the signal only (ideally). For broadband interferers, the adaptive weighting must consist of more than one weight, as discussed in Section 2.2.

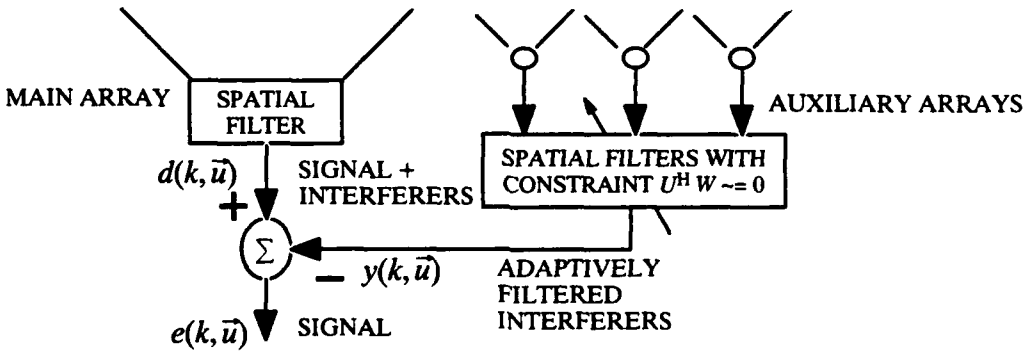


Figure 5.3. Generalized sidelobe canceller.

For the GSC to keep from cancelling the signal, it is imperative that the auxiliary arrays *output only interferers*; that is, that they place a spatial–spectral null in the direction in which the main array is steered. For fixed arrays with regular geometries (like the ULA), steer directions can be determined for each auxiliary array which place a spatial null (of the auxiliary array's array pattern) exactly in the direction in which the main array is steered. For example, the M–element linear auxiliary array will have a spatial null in the ϕ_{main} direction when the auxiliary array is steered in a direction given by [27]:

$$\phi_{aux} = \sin^{-1}\left(\sin(\phi_{main}) + \frac{2}{M}i\right) \quad \text{for } i = integer \neq 0 \tag{5.11}$$

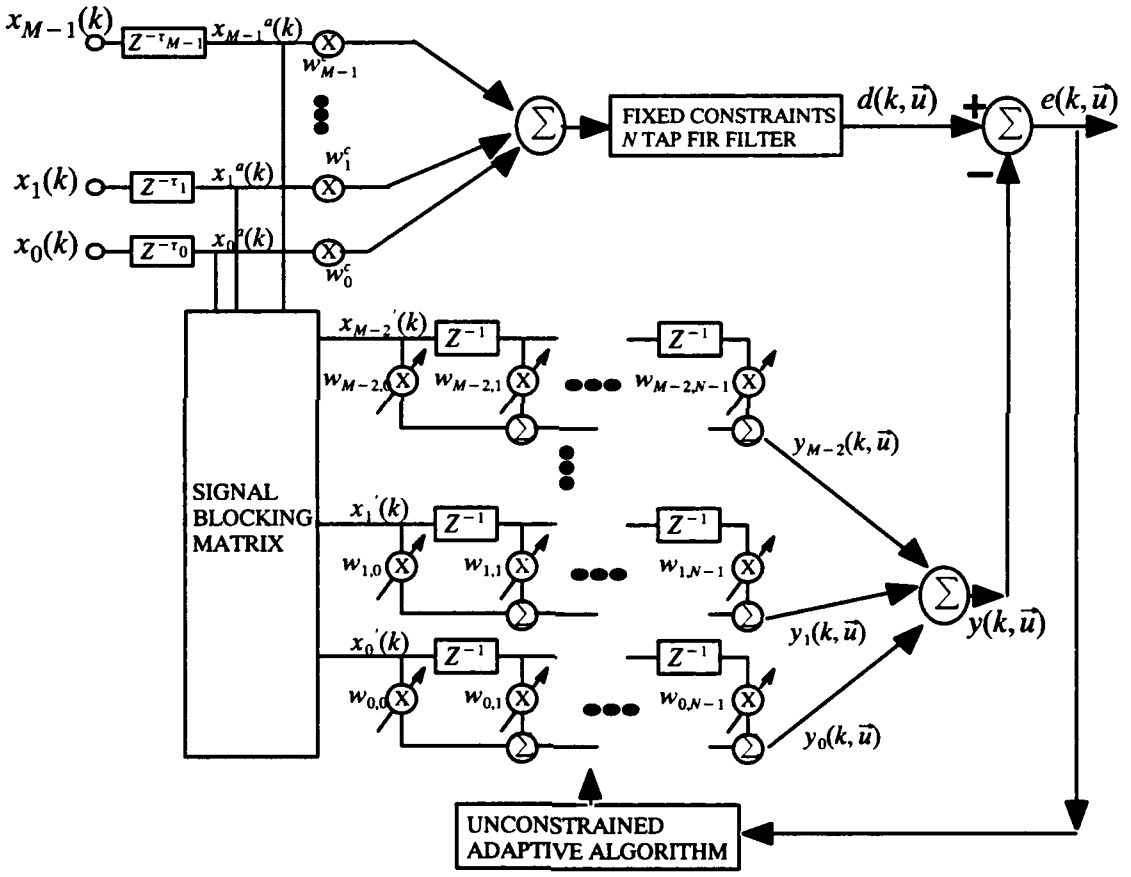


Figure 5.4. Generalized sidelobe canceller implementation of the linear constrained adaptive beamformer for broadband sources [24].

M-1 different directions exist which satisfy Eq. (5.11). In addition, [27] presents a technique for reusing elements of the main array to form the auxiliary arrays with minimal GSC degradation due to signal correlation between the auxiliary and main arrays. In general, however, a data-independent blocking matrix must be used, as seen in figure 5.4, which guarantees that only spatially different sources (assumed to be *uncorrelated* interferers!) are passed to the multichannel adaptive filter. Since the main array is nonadaptive, no channel equalization is

possible with the GSC. The performance of the GSC can be expected to deteriorate in a multipath environment because the multipath allows correlated energy to enter in the auxiliary array.

5.3. BROADBAND CBF INCORPORATING ADAPTIVE EQUALIZATION

One problem with the multichannel adaptive filter presented in Section 5.1. is that for a significant number of elements, the equalization tends to be very noisy due to the weight misadjustment (a function of the total number of weights, MN). By treating each element as a separate, single channel, as in figure 5.5, the number of weights in the update loop can be reduced by M and the equalization performance enhanced due to the reduced misadjustment noise, but now the spatial cancellation properties have been reduced to that of a nonadaptive TD-CBF. This technique without the presteering has been successfully used recently in an underwater acoustic communication channel [17]. In communication channels, it is imperative to equalize each element's data before summation. Like figure 5.2, figure 5.5 relies on a desired signal which could be supplied by a TD-CBF to form a variation of the GSC presented in the previous section.

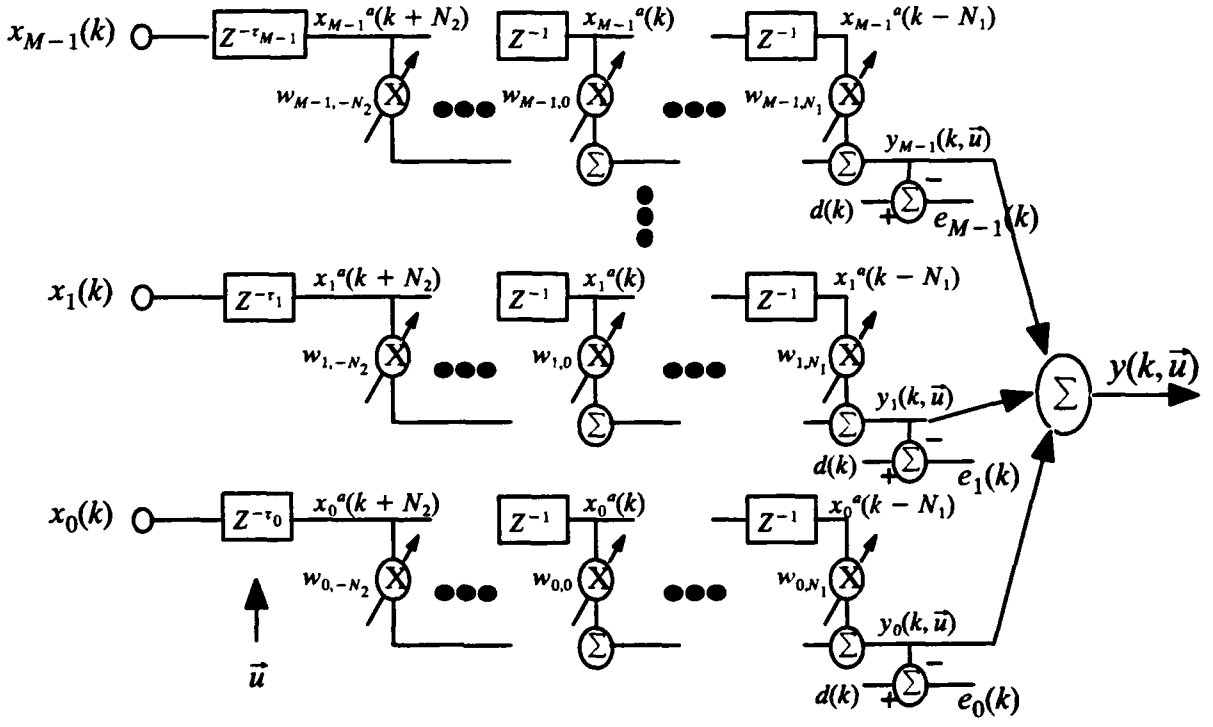


Figure 5.5. Time-domain CBF with adaptive channel equalization.

6. SIMULATIONS

Two simulations were conducted to compare the performance of the presented beamforming algorithms. The first simulation considers the case of multiple independent narrowband sources while the second simulation considers a single broadband source in a single multipath environment. Both simulations consider large SNRs only.

6.1. THREE NARROWBAND SOURCES WITH LARGE SNRS

The first simulation compares the TD-CBF, GSC, Bartlett, and MVDR beamformers for the case of three separate narrowband sources incident on a 10-element linear vertical array with large SNRs. The MATLAB generation file listed in figure 6.1 shows the details of the simulation as well as the details of the beamforming and plot generation. The power in the three narrowband signals differs by 6 dB and all sources are incident at different angles, as described in table 6.1. The array is "cut" for a 25-Hz signal (i.e., the frequency at which the distance between elements is half the incident signal's wavelength). Appendix A describes the MATLAB files and their relationships to those generated in [6]. The additive gaussian noise has a variance of 0.0001. (The adaptive techniques presented here will actually work better with more random noise because the noise helps to condition the data covariance matrix.)

Table 6.1. Source parameters for first simulation.

Source	Azimuth (deg)	Elevation (deg)	Frequency (Hz)	Amplitude	Phase (deg)
1	90	0.0	25.0	1	0
2	90	−14.0	45.0	2	45
3	90	30.0	15.0	4	90

Figure 6.2 summarizes the two time-domain beamforming algorithms, TD-CBF and GSC, while figure 6.3 summarizes the two frequency-domain beamforming algorithms, Bartlett and MVDR. Figure 6.2 shows the effects of the anti-(spatial)aliasing filter with the TD-CBF. Recall that any incident energy at a frequency greater than that which satisfies the half-wavelength spacing of the array, here 25 Hz, shows up as a directional ambiguity due to the existence of grating lobes. When not completely filtered out, the 45-Hz/−14-degree signal creates havoc in the output beampattern. The antialiasing filter used here is a 29-tap band-pass filter (5−40 Hz) with linear phase. The apparent weak signal incident at roughly +60 degrees is from the 45-Hz/−14-degree signal leaking through a grating lobe. The output can be improved by using a better antialiasing filter. Figure 6.4 plots the power spectral density (PSD) for the TD-CBF and GSC output when they are pointed in the three directions of the incident signals. The figure clearly shows that, as expected, the GSC is better able to spatially filter out the signals not in the steering direction.

Figures 6.5−6.8 plot the beamforming characteristics for the MVDR and Bartlett frequency-domain beamforming algorithms. A grating lobe is clearly defined by the 45-Hz/−14-degree signal in figure 6.5. The 15-Hz/+30-degree signal creates the sine inverse function in figure 6.5 described by Eq. (2.5). It is due to spectral leakage (even though a 1024-point FFT with 50% overlap is used) from the 15-Hz/+30-degree signal, since the background noise is negligible. Essentially, MVDR is magnifying this minimal energy. When the same simulation was rerun with a noise variance of 0.01, MVDR displayed spectral peaks, without the sine inverse effects, as desired.

MVDR seems to work particularly well only at the ideal operating frequency of the array. For instance, even though the 15-Hz/+30-degree signal has the most power of all incident sources, the MVDR algorithm does not display the +30 degree direction with the most output power in figure 6.6. On the other hand, the Bartlett algorithm gives a truer output value in figure 6.8.

Figure 6.3 plots the sum of frequencies from 5−45 Hz for both frequency-domain beamformers for comparison with the time-domain beamformer outputs plotted in figure 6.2. Note the similarities between the TD-CBF and the FD-CBF outputs. When the frequency-domain beamformers were summed over 5−25 Hz, the 15-Hz/+30-degree and 25-Hz/0-degree signals are displayed cleanly without the grating lobe interference centered at +60 degree and without the −14 degree peak.

```
%=================================================================
% NB_ULVA
% Linear array of 10 elements spaced 30 meters apart assumed to be
% arranged along the z-axis (vertical) with mulitple incident narrowband
% sources
%      Source  Azimuth  Freq   Elevation   Amp  Phase
%        1.      90      25       0        1.0    0
%        2.      90      45      -14       2.0    45
%        3.      90      15       30       4.0    90
%
%=================================================================
%-----------------
% initialize parameters
%-----------------
 fs = 1000;                                        % Sample increment (1/1000Hz) (sec)
 dt = 1/(fs);
 T = 4096;                                         % Number of samples
 c = 1500;                                         % Speed of sound (m/sec)
 M = 10;                                           % Number of elements
 % Array of element positions along z axis (meters)
 P = (30) * [ zeros(1,M);zeros(1,M);0:(M-1)];
%-----------------
% generate narrowband complex signals
%-----------------
 f_sig = [ 25 45 15];                              % Signal's frequency (Hz)
 amp_sig = [ 1.0 2.0 4.0];          % Signal's Amplitude
 phase_sig = [ 0.0 45.0 90.0];      % Signal's Phase (degrees)
 az_sig = [ 90 90 90];              % Signals Azimuth angle (degrees)
 el_sig = [ 0 -14 30];              % Signals Elevation angle (degrees)
 v = 0.0)*ones(1,M);                % Noise vector
 % Generate complex element array data
 dgr = (pi/180);                                   % conversion from degrees to radians
 L = length(f_sig);                 % number of signals
 X=zeros(M,T);
 for l=1:L-1
   u = [ cos(az_sig(l)*dgr)*cos(el_sig(l)*dgr), ...
         sin(az_sig(l)*dgr)*cos(el_sig(l)*dgr), ...
         sin(el_sig(l)*dgr)];
  % no noise
 X = X+nb_sig_gen(f_sig(l),P,u,amp_sig(l),phase_sig(l),dt,T,c,v*0);
 end
 u = [ cos(az_sig(L)*dgr)*cos(el_sig(L)*dgr), ...
       sin(az_sig(L)*dgr)*cos(el_sig(L)*dgr), ...
       sin(el_sig(L)*dgr)];
 % including noise
 X = X+nb_sig_gen(f_sig(L),P,u,amp_sig(L),phase_sig(L),dt,T,c,v);
%-----------------
% Beamforming
%-----------------
 % search directions
 az=[90];
 el=[-90:89];
 %*****************************************************************
% freq-domain beamforming
 %*****************************************************************
% beamforming frequencies must be exactly equal to
%  temporal FFT discrete frequencies
F=[1:59]*fs/1024;
Py=zeros(length(el),length(F));
for i=1:length(F)
  R = sfecsd( X , F(i), P , c, dt, 'OverlapFFT: 1024');
  PyB(:,i)=bfclass( X, R, F(i), az, el, P, dt, c);
  PyM(:,i)=mvdr( X, R, F(i), az, el, P, dt, c);
end
%mesh(flipud(db(Py)'))
%contour(flipud(db(Py)'))
%plot(el,db([Py(:,26),Py(:,15),Py(:,46)])) %25.3,14.6,44.9 Hz
%plot(F,db([Py(77,:);Py(91,:);Py(121,:)])') % -14deg,0deg,30deg
%plot(el,db([sum(PyB(:,5:46'));sum(PyM(:,5:46'))])') % sum over 5-45Hz
 %*****************************************************************
% time-domain beamforming
 %*****************************************************************
N = 10;
mu = 0.0005;      % normalized lms parameter
% Linear Phase BPF characteristics required for TD Beamforming
% (note 1: with a 20 tap FIR filter, the 0-5Hz is really not attenuated)
% (note 2: group delay = ~ 10 iterations )
 a_bpf = [1];
b_bpf = fir1(29,[5/(fs/2) 40/(fs/2)]);             % fbegin = 5 Hz  - fstop = 40 Hz
 for m=1:M
    X(m,:)=filter(b_bpf,a_bpf,X(m,:));
 end
Py=zeros(length(el),1);
B=toeplitz([1 zeros(1,M-3)],[1 -2 1 zeros(1,M-3)]);
for i=1:30:180
  % conventional time-domain beamforming
  [PyC(i:i+29),y] = td_cbf(X,az,el(i:i+29),P,dt,c);
  % Generalized Sidelobe Canceller
  [PyG(i:i+29),y] = gsc(X,az,el(i:i+29),P,dt,c,N,mu,B);
end
```

Figure 6.1. MATLAB generation file for first simulation.

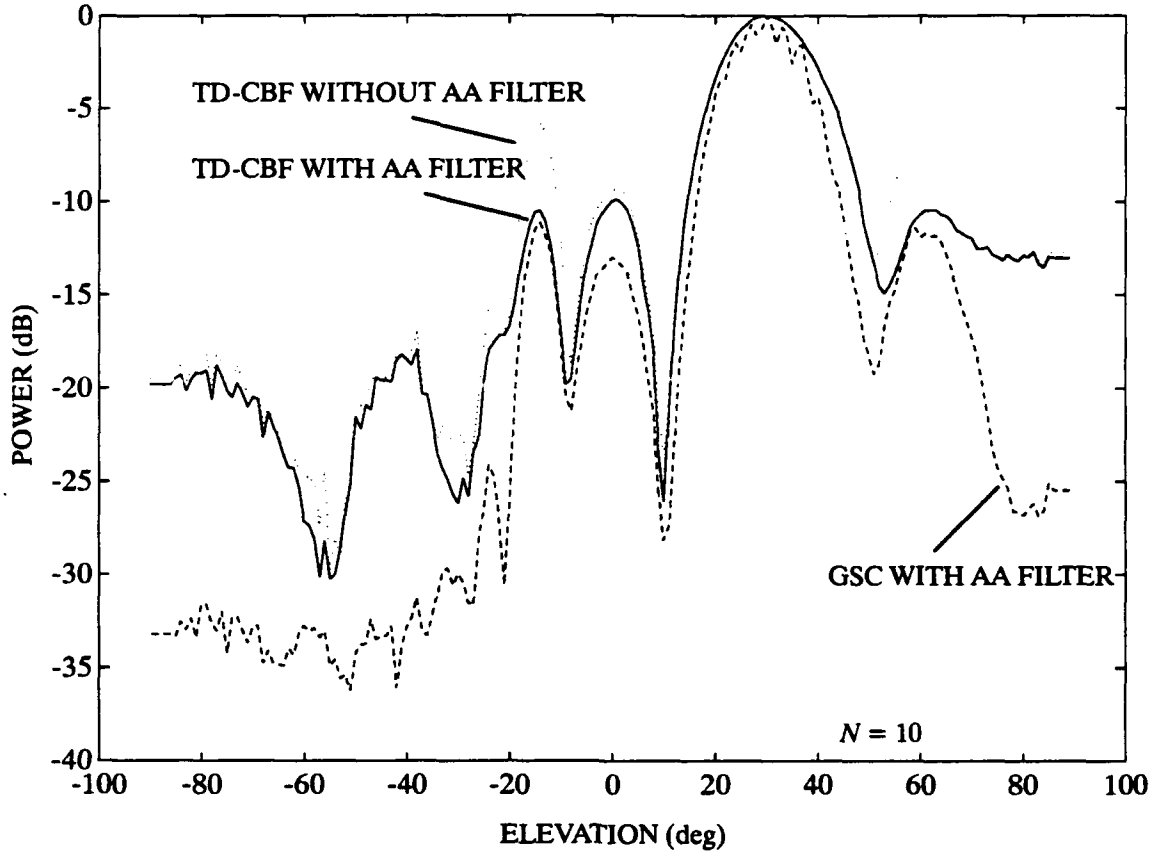


Figure 6.2. TD-CBF and GSC beamforming for three narrowband sources.

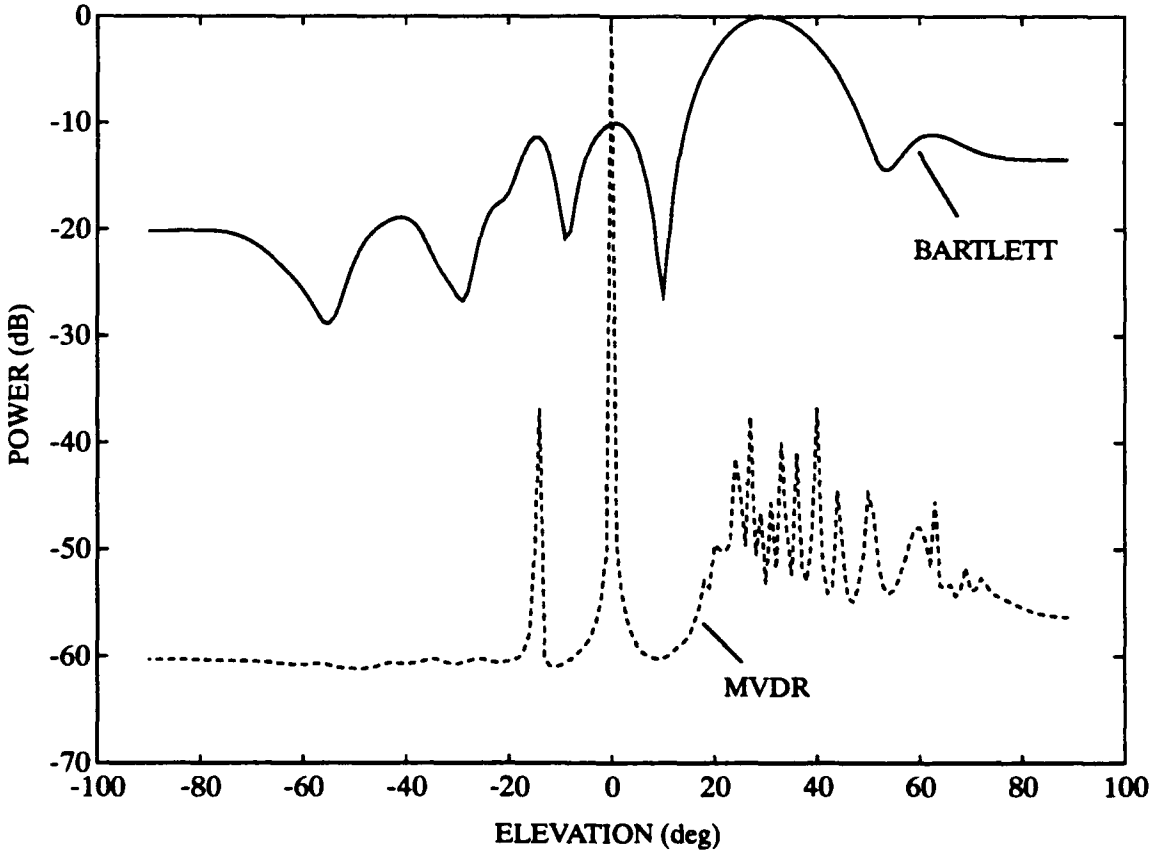


Figure 6.3. Bartlett and MVDR beamforming summed from 5 to 45 Hz for three narrowband sources.

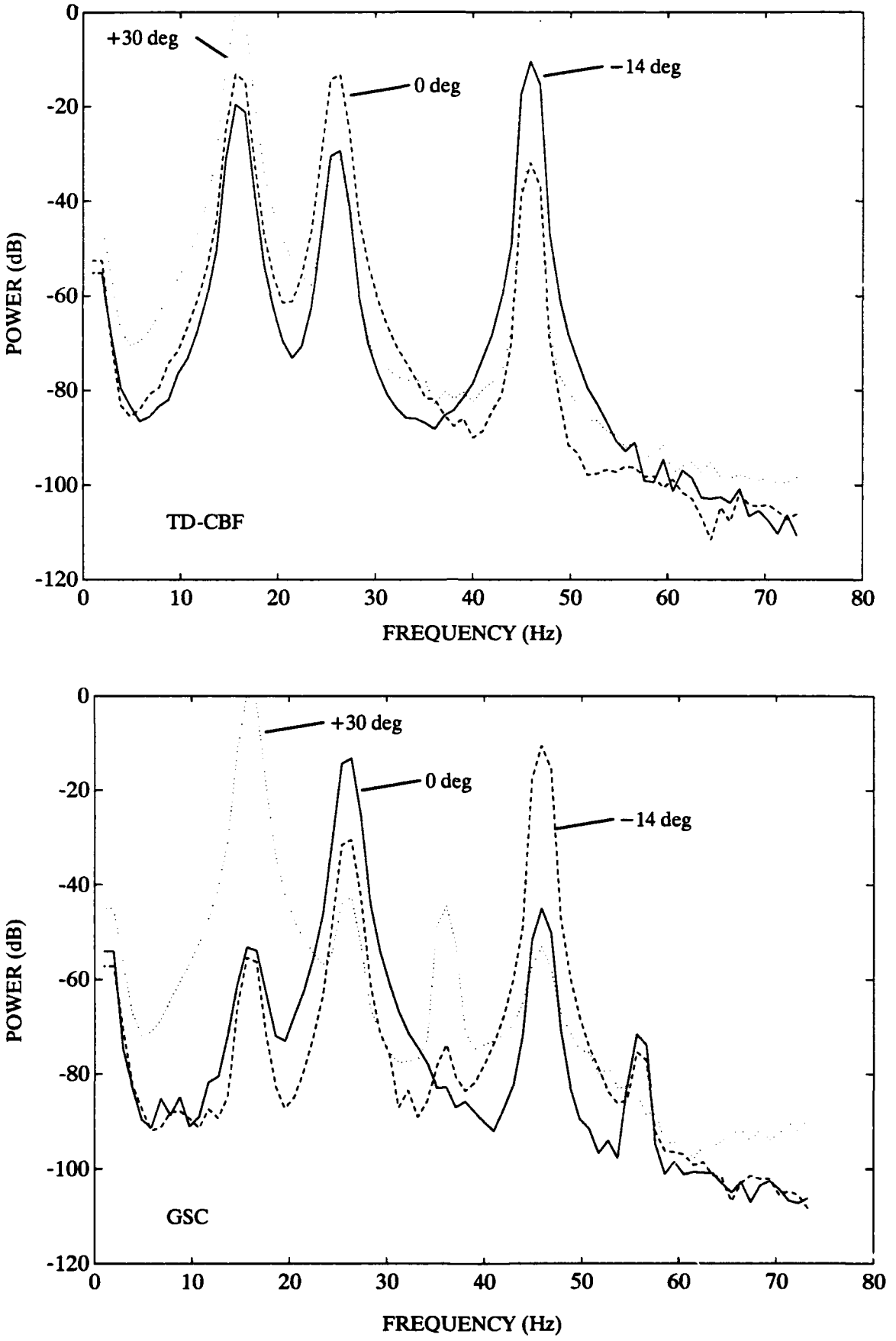


Figure 6.4. TD-CBF and GSC PSDs along −14-, 0-, and +30-degree steering directions for three narrowband sources.

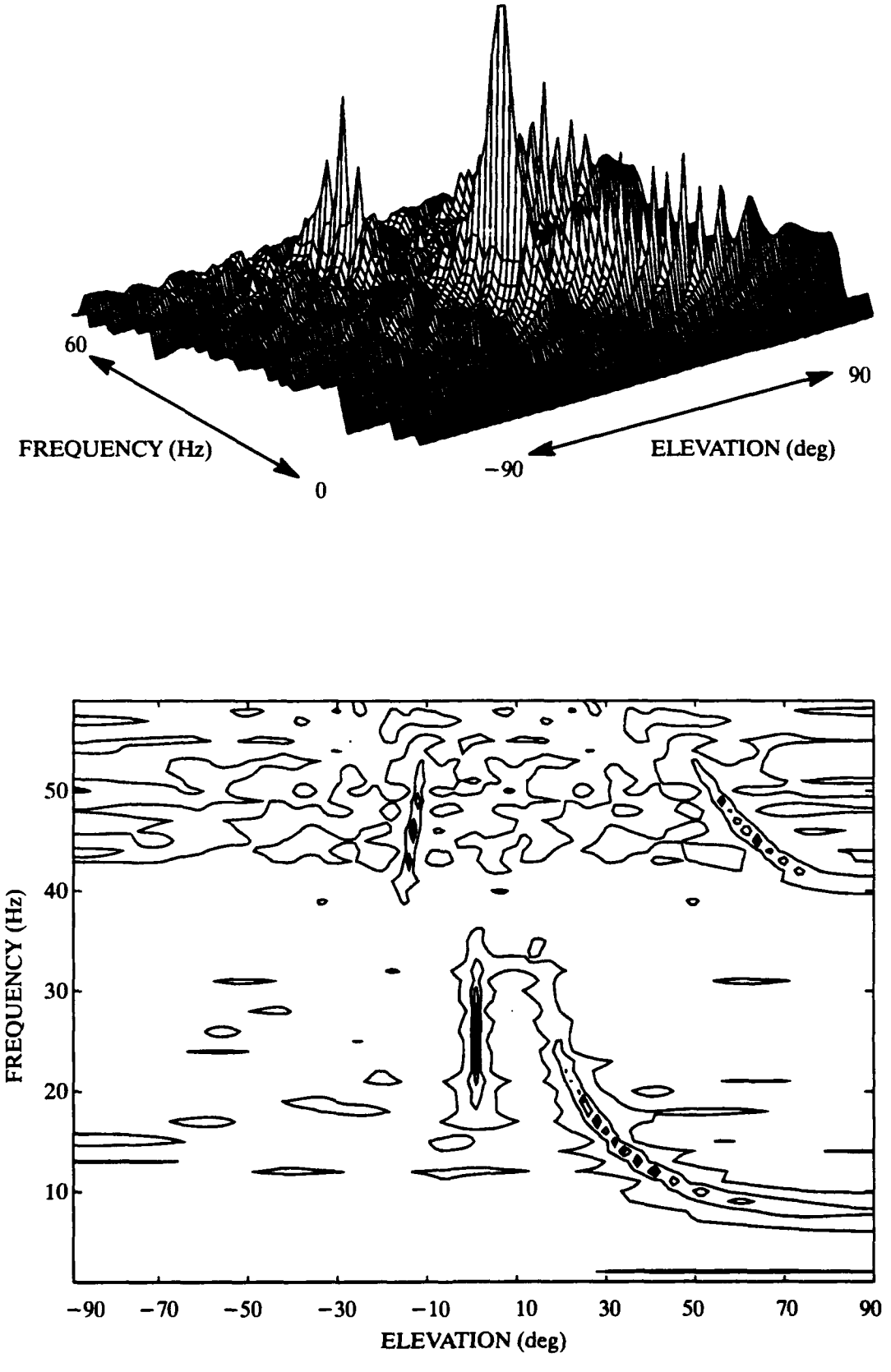


Figure 6.5. MVDR beamforming for three narrowband sources; FFT size of 1024 points.

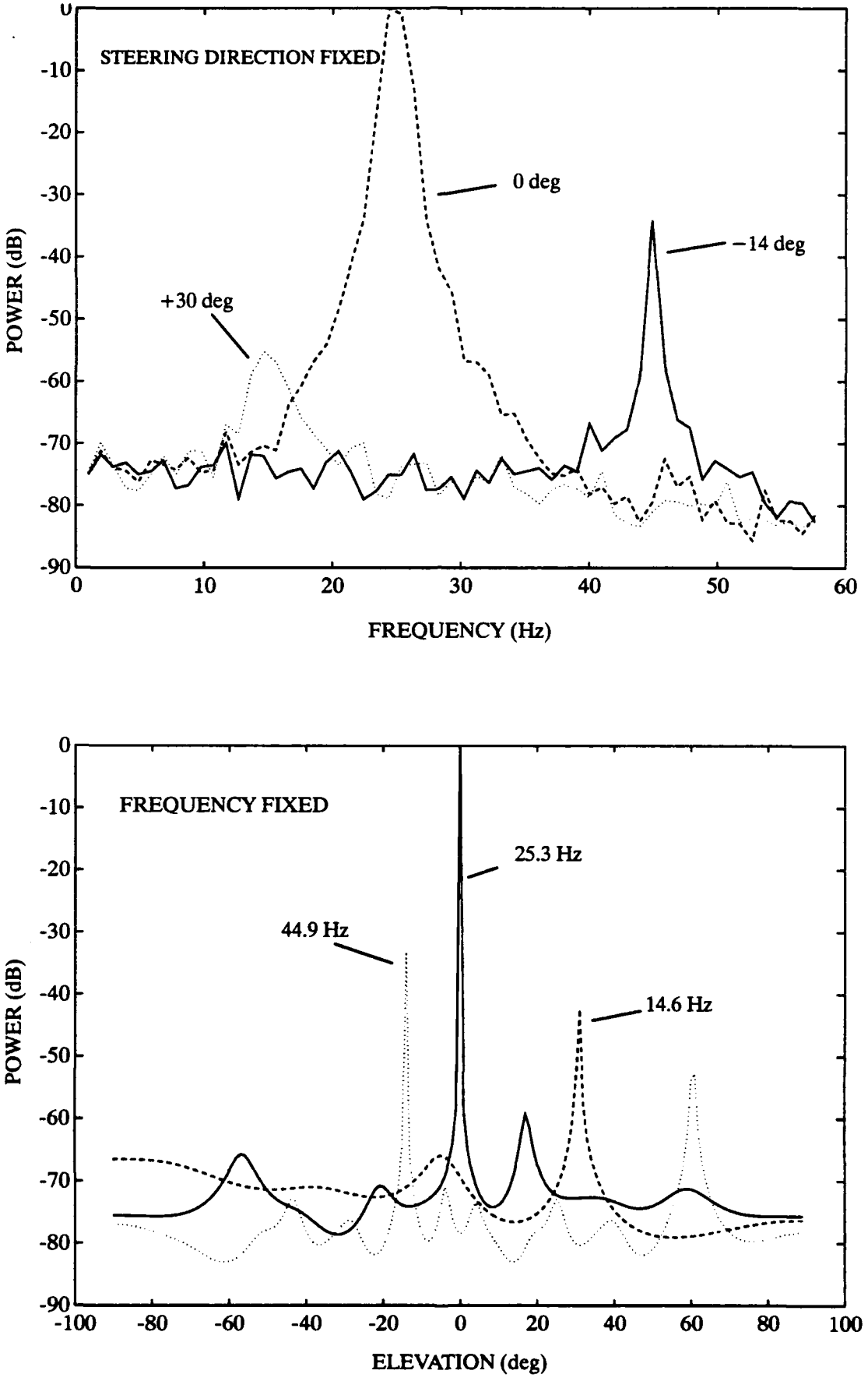
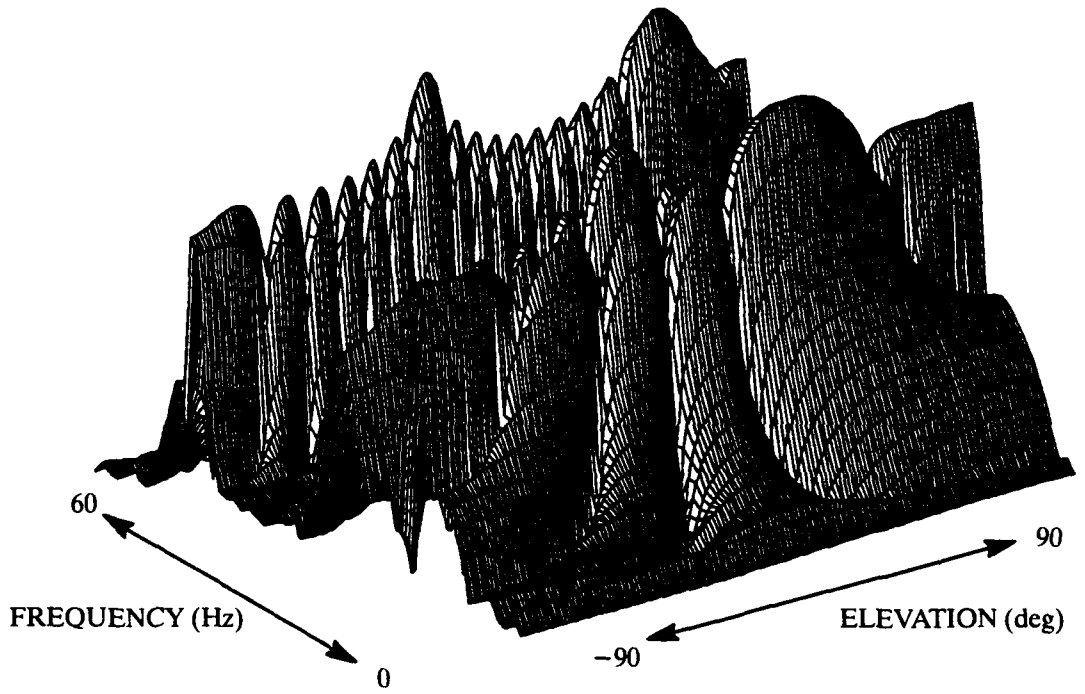


Figure 6.6. MVDR beamformer output for three narrowband sources along fixed steering directions (top) and along fixed processing frequencies (bottom).



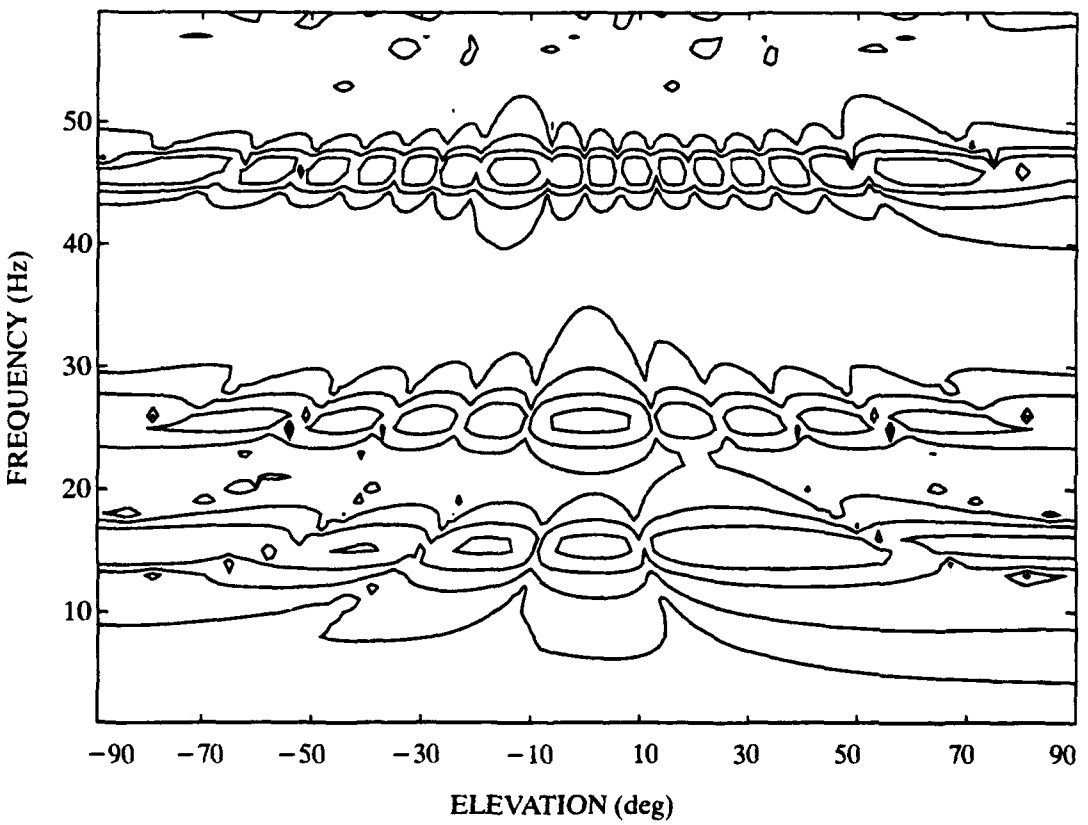
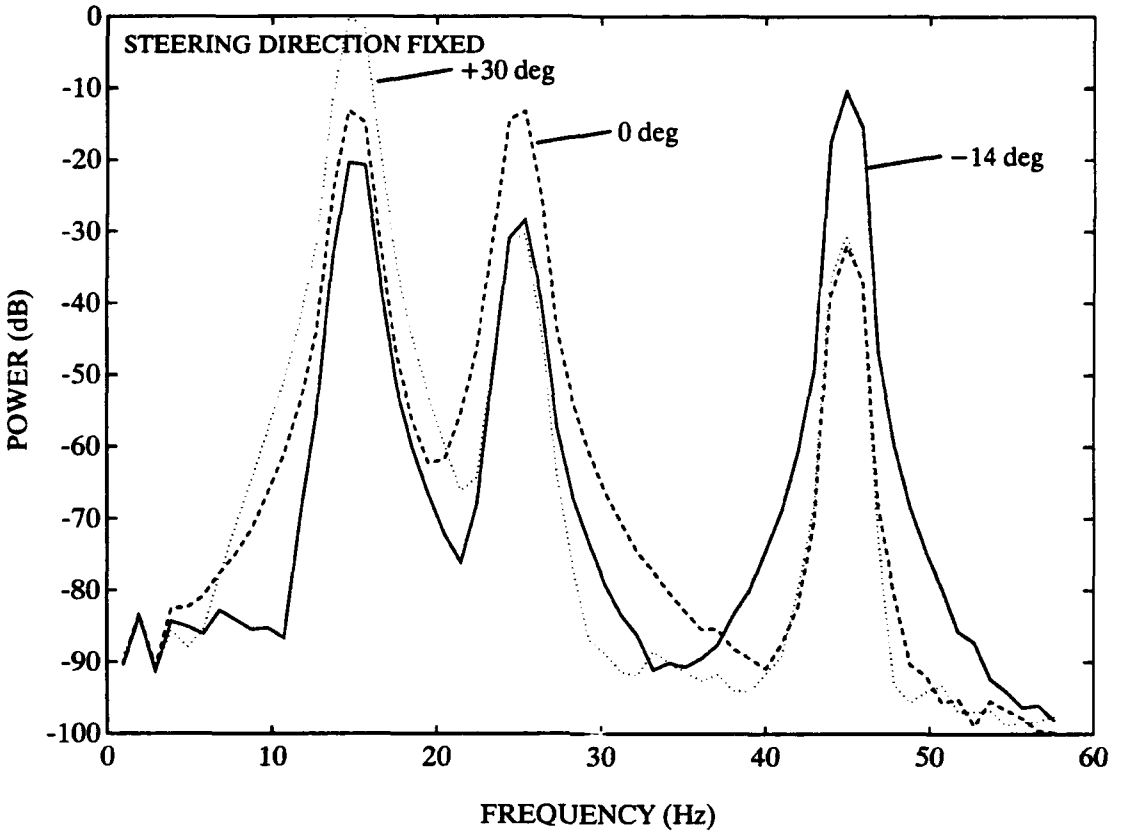


Figure 6.7. Bartlett beamforming for three narrowband sources; FFT size of 1024 points.



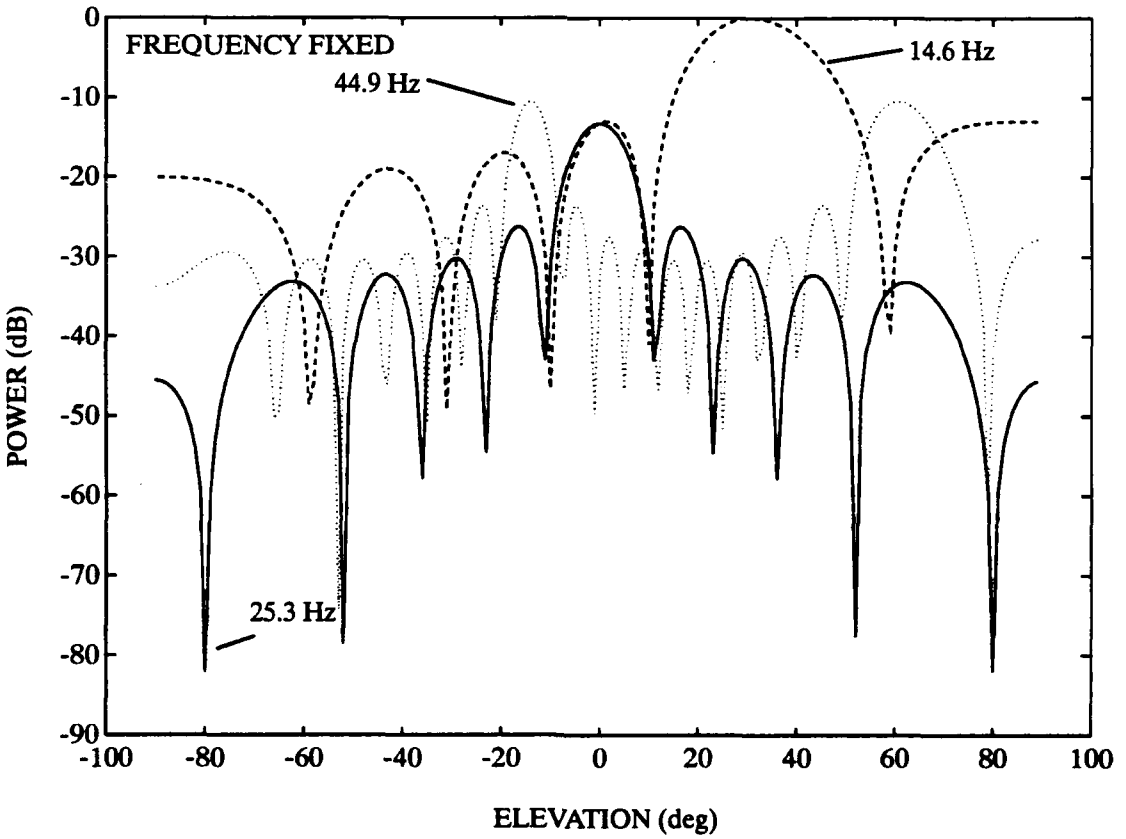


Figure 6.8. Bartlett beamformer output for three narrowband sources along fixed steering directions (top) and along fixed processing frequencies (bottom).

6.2. SINGLE BROADBAND SOURCE WITH SINGLE MULTIPATH

The second simulation compares the TD-CBF, GSC, Bartlett, and MVDR beamformers and the multichannel adaptive filters for the case of a single broadband source incident at 0 degrees and its single multipath incident at −14 degrees on a 5-element linear array. The MATLAB generation file listed in figure 6.9 shows the details of the simulation as well as the details of the beamforming and plot generation. Figure 6.10 shows the PSD of the broadband signal and the received data at element 1 and element 5. The broadband signal generated by the MATLAB file "bb_sig_gen.m" (see Appendix A) is composed of four separate components: (1) 15−25 Hz band-pass filtered noise which is amplitude-modulated at 2 Hz, (2) exponentially low-pass filtered noise, (3) a narrowband signal at 18 Hz, and (4) three odd harmonics of the narrowband signal at 24.17 Hz. The effects of the multipath are clearly seen as spectral notches (refer to Section 2.4) in the received data's PSDs. The channel is modeled after the simulation outlined in Section 2.3. The FIR filter coefficients for each source-to-element channel are defined as

```
b(1,:)=[ 0.9 zeros(1,31) 0.9 0 0 0 0 0.0 0 0 0 0 0.0 0 0 0 0 0.00 0 0 0 0 0.00];
b(2,:)=[ 0.9 zeros(1,31) 0.0 0 0 0 0 0.9 0 0 0 0 0.0 0 0 0 0 0.00 0 0 0 0 0.00];
b(3,:)=[ 0.9 zeros(1,31) 0.0 0 0 0 0 0.0 0 0 0 0 0.9 0 0 0 0 0.00 0 0 0 0 0.00];
b(4,:)=[ 0.9 zeros(1,31) 0.0 0 0 0 0 0.0 0 0 0 0 0.0 0 0 0 0 0.85 0 0 0 0 0.00];
b(5,:)=[ 0.9 zeros(1,31) 0.0 0 0 0 0 0.0 0 0 0 0 0.0 0 0 0 0 0.00 0 0 0 0 0.47];
```

which defines the direct path incident at 0 degrees (interelement delay equals 0) and the multipath incident at −14 degrees (interelement delay equals 5). The total power in the multipath is roughly 0.8 times that in the direct path.

Figures 6.11 and 6.13 summarize the results of the two time-domain beamformers, and figures 6.12, 6.14−6.17 summarize the results from the two frequency-domain beamformers. In figure 6.11, the TD-CBF displays both direct and multipath incident directions. However, the GSC displays no multipath at −14 degrees because the direct path is adaptively filtered by the auxiliary arrays and subtracted from the main array's multipath output. As can be seen in figure 6.13, neither beamformer eliminates the spectral notch created by the multipath.

In figures 6.14 and 6.15, MVDR is seen to effectively lock in on the 24.17-Hz narrowband signal, giving excellent directional data at 24 Hz for both the direct and multipath signals. However, the multipath signal is displayed at a power level significantly lower than the actual incident power level, presumably because of the strong correlation with the direct path source. The Bartlett beamformer, plotted in figures 6.16 and 6.17, has significant difficulty in any directionality, presumably because of the wide beamwidths of the 5-element array. However, it is readily apparent from the PSDs along the 0-degree and −14-degree direction in figure 6.17, that the two signals are related. This was not the case for the MVDR algorithm in figure 6.15.

Figure 6.12 plots the su: .ie frequency-domain beamformer outputs from 5−45 Hz for comparison with the time-domain beamformer outputs in figure 6.11. The MVDR beamformer displays peaks at the correct directions corresponding to the incident direct and multipath signals, but the relative power between the signals is significantly different than the actual 0.8. It is interesting how much better the TD-CBF (using the same antialiasing filter as before) did with respect to the Bartlett beamformer. One might have expected similar results between the two, as was the case for narrowband signals in the previous simulation. This might be due to energy at frequencies greater than 45 Hz or to differences in the spatial-temporal correlation matrix and the spatial-spectral correlation matrix as suggested by [41].

The weights of the multichannel adaptive filter described in Section 5.1 are plotted in figure 6.18 for presteering angles of 0 degrees and −14 degrees and $N_1 = N_2 = 75$. The weights show that in both cases, the signal with more power (0-degree incident signal) is interpreted as the direct path by the filter. Thus, as long as both direct path and multipath lie within the multichannel adaptive filter's spatial window defined by N (refer to Section 5.1), the output power contains no directional information. However, the weights could be further processed to obtain directional information. Figure 6.19 shows that the PSD of the multichannel adaptive filter output has been equalized since no spatial nulls exist. Figures 6.20 and 6.21 plot the weights of the multiple single−channel adaptive filters discussed in Section 5.3 for $N_1 = N_2 = 75$ and $N_1 = 150$, $N_2 = 0$. The weights are seen to take on values described by Eq. 2.20.

```
%*****************************************************************
% MPATH_ULVA
% This script file generates the Uniform Linear Verical Array
% element time-series inputs for a multipath environment.  The multipath
% is generated by propagating a signal (broad or narrow band) through
% separate channels (modeled as an ARMA linear filter) between each sensor
% and a single source.
%
% written by: Rich North 7-1-92
%*****************************************************************
%---------------
% initialize parameters
%---------------
 T = 8192;                          % Number of samples(2^13)
 fs = 1000;                         % Sample frequency
 dt = 1/(fs);                       % Sample increment (sec)
 c = 1450;                          % Speed of sound (m/sec)
 M = 5;                             % Number of sensors in array
 X=zeros(M,T);                      % array element time-series data
 f0=24.17;                          % primary CW wave (d = lambda/2)
 % Array of sf_sig_gen positions along z axis (meters)
 P = (30) * [ zeros(1,M);zeros(1,M);0:(M-1)];
%---------------
% generate signal
%---------------
 % (real) broad band signal
 s=bbsig_gen(T,25.0,15.0,f0,18,fs);
%---------------
% propagate signal through ARMA channel
%---------------
 rand('normal');
 nvar = 0.01;
 anoise = sqrt(nvar);
 u = zeros(1,T);
 %source-sensor#1-M channel
 a = [ 1 ];
 Lfir=53;
 b=zeros(M,Lfir);
 b(1,:)=[ 0.9 zeros(1,31) 0.9 0 0 0 0 0 0.0 0 0 0 0 0 0 0.0 0 0 0 0 0 0 0.0 0 0 0 0 0 0];
 b(2,:)=[ 0.9 zeros(1,31) 0.0 0 0 0 0 0 0.9 0 0 0 0 0 0 0.0 0 0 0 0 0 0 0.0 0 0 0 0 0 0];
 b(3,:)=[ 0.9 zeros(1,31) 0.0 0 0 0 0 0 0.0 0 0 0 0 0 0 0.9 0 0 0 0 0 0 0.0 0 0 0 0 0 0];
 b(4,:)=[ 0.9 zeros(1,31) 0.0 0 0 0 0 0 0.0 0 0 0 0 0 0 0.0 0 0 0 0 0 0 0.85 0 0 0 0 0];
 b(5,:)=[ 0.9 zeros(1,31) 0.0 0 0 0 0 0 0.0 0 0 0 0 0 0 0.0 0 0 0 0 0 0 0.0 0 0 0 0 0 0.47];
 for m=1:M
   u=filter(b(m,:),a,s)+anoise*rand(u);
   X(m,:)=u;
 end
%---------------
% Beamforming
%---------------
 % search directions
 az=[90];
 el=[-90:89];
 %*****************************************************************
 % freq-domain beamforming
 %*****************************************************************
 % beamforming frequencies must be exactly equal to
 %   temporal FFT discrete frequencies
 %F=[1:59]*fs/1024;
 %Py=zeros(length(el),length(F));
 %for i=1:length(F)
 %  R = sfecsd( X , F(i), P , c, dt, 'OverlapFFT: 1024');
 %  Py(:,i)=bfclass( X, R, F(i), az, el, P, dt, c);
 %  Py(:,i)=mvdr( X, R, F(i), az, el, P, dt, c);
 %end
 %plot(el,db([Py(:,25),Py(:,10),Py(:,19)])) %24.4,9.8,18.5 Hz
%plot(F,db([Py(77,:);Py(91,:)])')          % -14deg,0deg
 %plot(el,db(sum(PyB(1:180,5:45)')),el,db(sum(PyM(1:180,5:45)')))
%*****************************************************************
% time-domain beamforming
%*****************************************************************
 % LMS filter parameters
 N = 150;
 mu = 0.05;                         % nlms parameter
 % Linear Phase BPF characteristics required for TD Beamforming
 a_bpf = [1];
 b_bpf = fir1(29,[5/(fs/2) 40/(fs/2)]);                  % fbegin = 5 Hz  - fstop = 40 Hz
 for m=1:M
    X(m,:)=filter(b_bpf,a_bpf,X(m,:));
 end
 %Py=zeros(length(el),1);
 %B=toeplitz([1 zeros(1,M-3)],[1 -2 1 zeros(1,M-3)]);
 %for i=1:30:180
  % conventional time-domain beamforming
  %[Py(i:i+29),y] = td_cbf(X,az,el(i:i+29),P,dt,c);
  % conventional time-domain beamforming w/ single channel equalization
  %[Py(i:i+29),y] = sdeq1(X,az,el(i:i+29),P,dt,c,N,mu,s);
  % Generalized Sidelobe Canceller
  %[Py(i:i+29),y] = gsc(X,az,el(i:i+29),P,dt,c,N,mu,B);
  % Multichannel Adaptive Filter
  %[Py(i:i+29),y] = mch_af(X,az,el(i:i+29),P,dt,c,N,mu,s);
 %end
 %[Py(77),y] = gsc(X,az,el(77),P,dt,c,N,mu,B);
 %Pxx1 = spectrum(y,1024,75);
 %[Py(91),y] = gsc(X,az,el(91),P,dt,c,N,mu,B);
 %Pxx2 = spectrum(y,1024,75);
%plot(F,db([Pxx1(1:59,1)';Pxx2(1:59,1)']))          % -14deg,0deg
```

Figure 6.9. MATLAB generation file for second simulation.

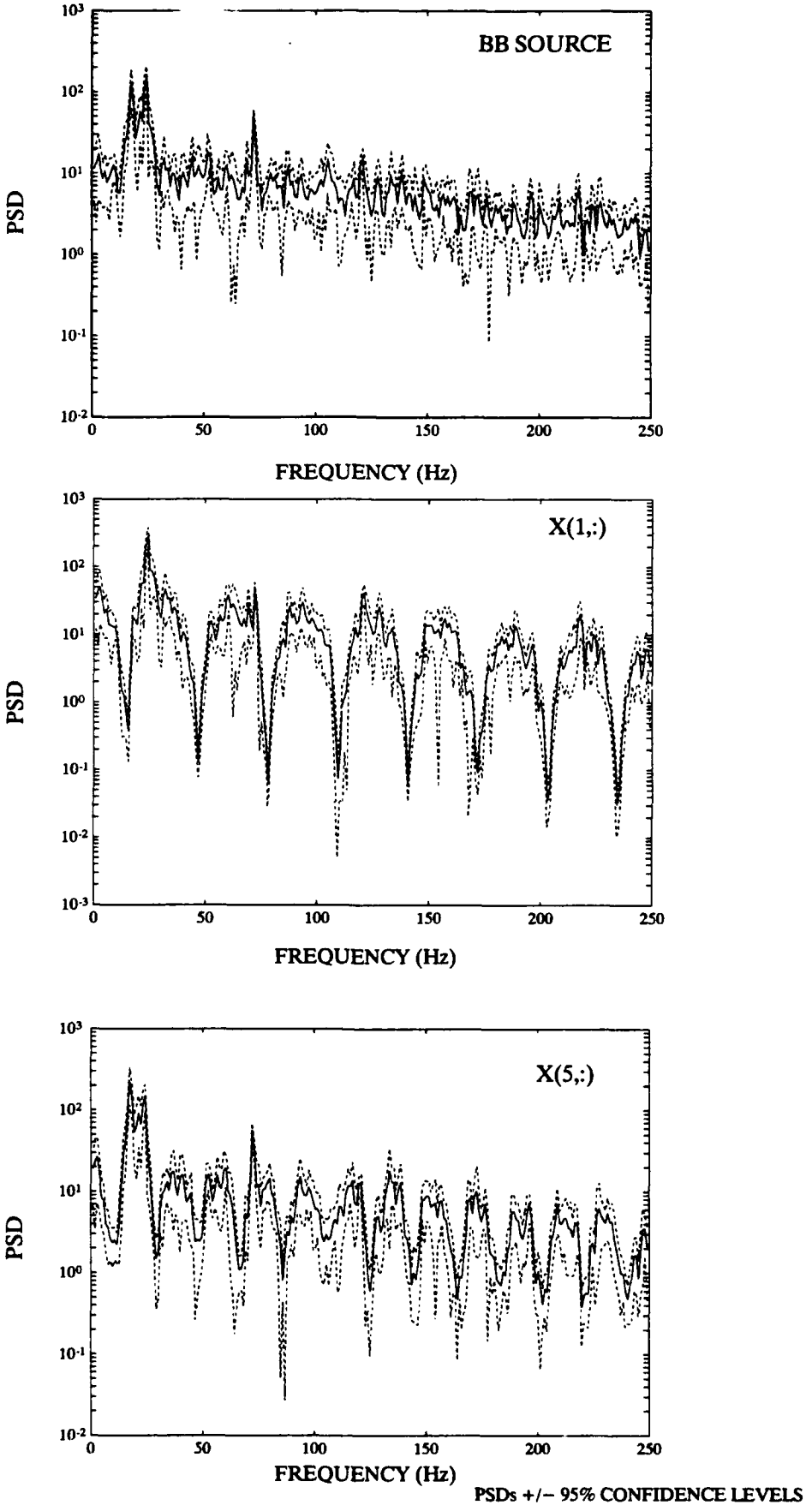


Figure 6.10. PSD of transmitted broadband source and received data from element 1 and 5.

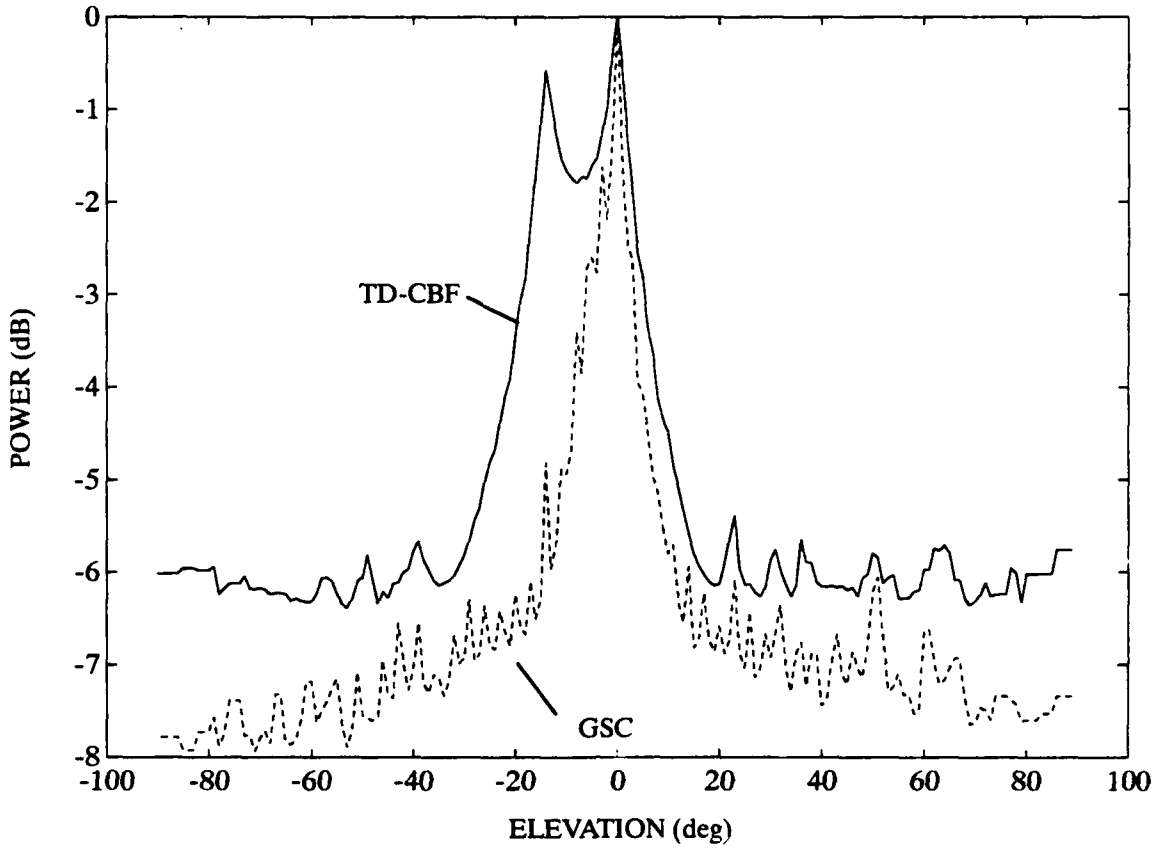


Figure 6.11. TD-CBF and GSC beamforming for a single multipath.

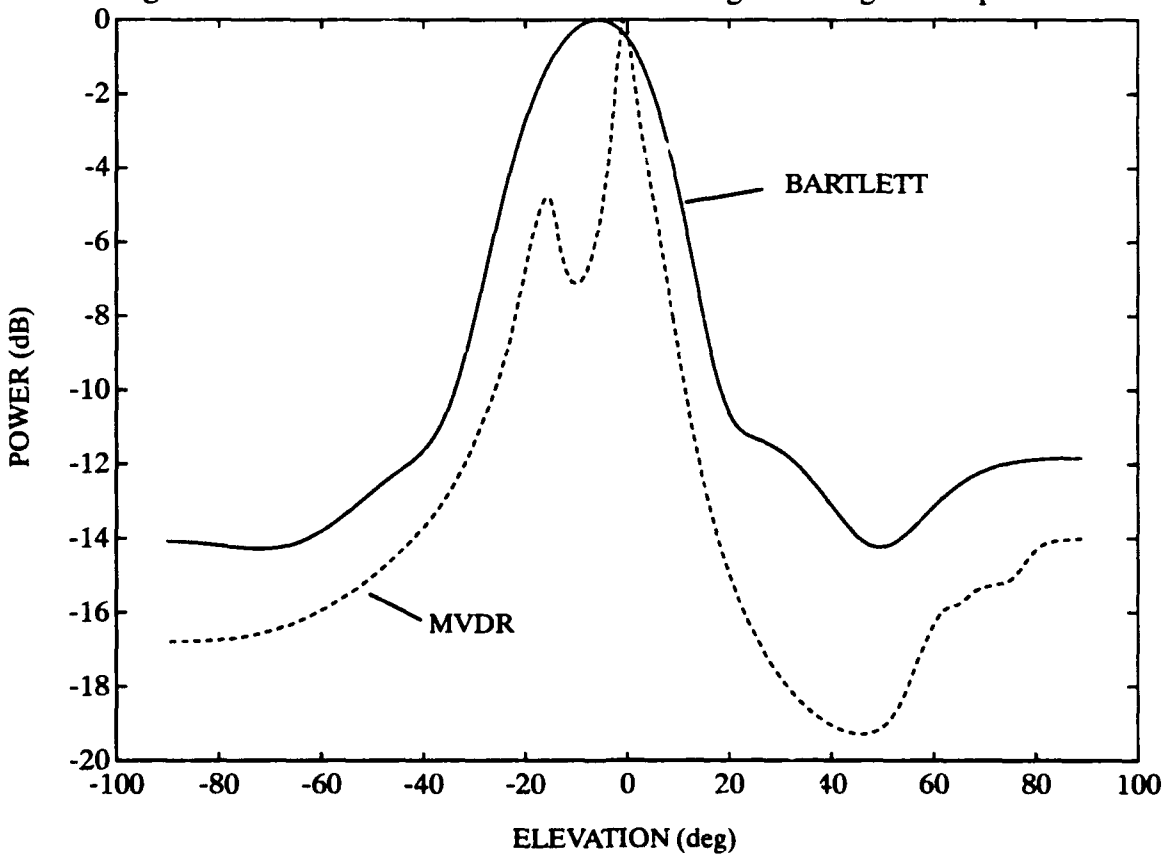
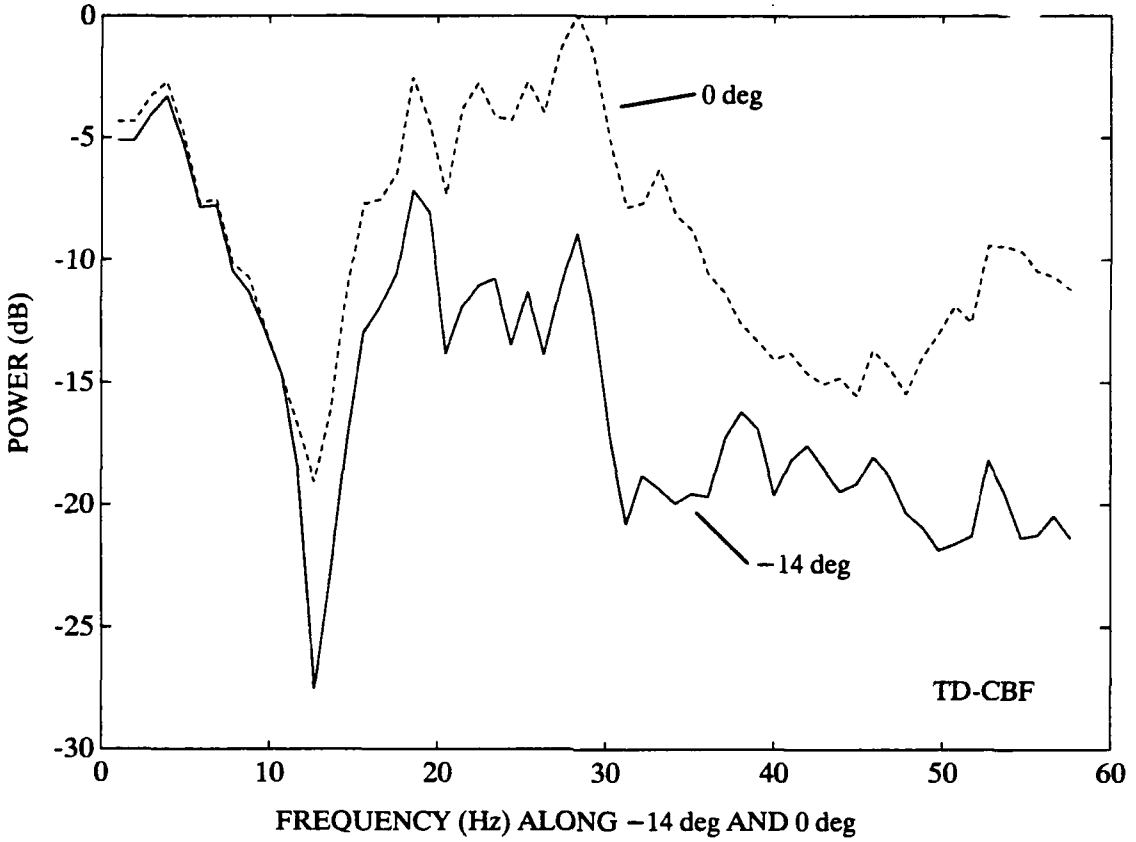


Figure 6.12. Bartlett and MVDR beamforming summed from 5 to 45 Hz for a single multipath.



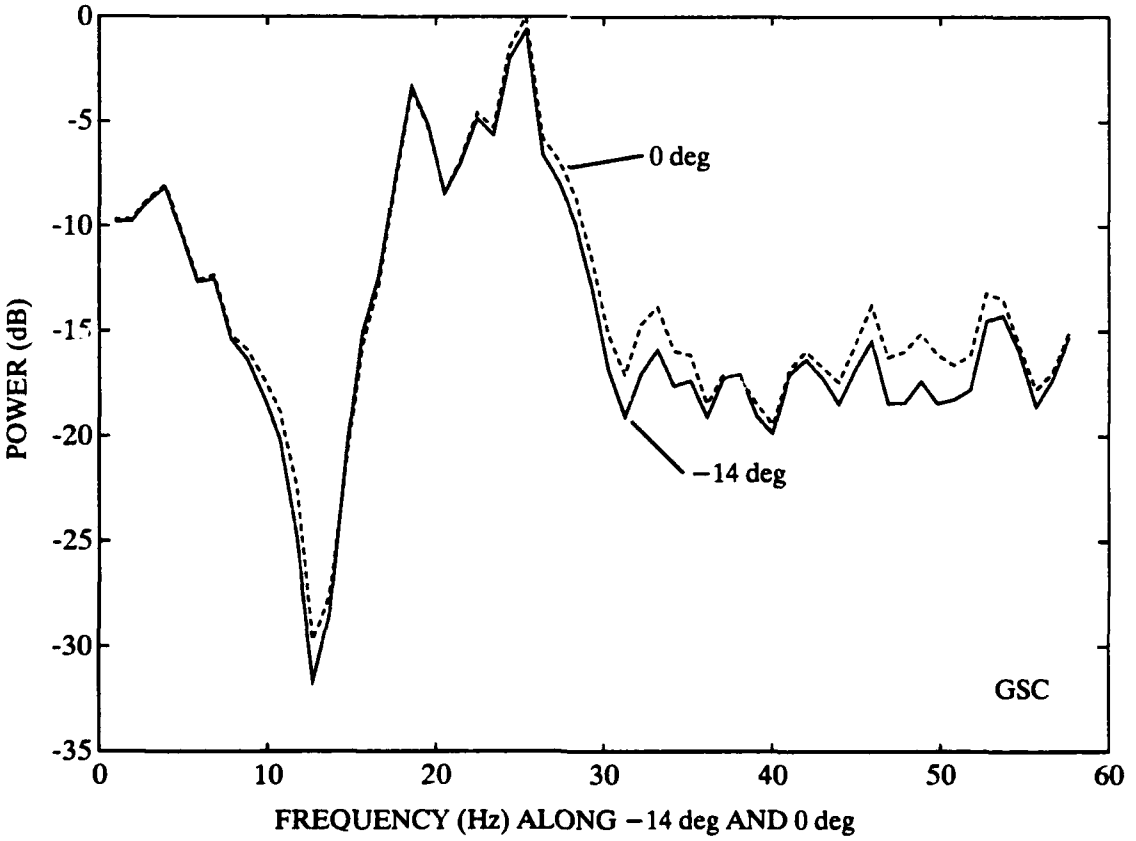


Figure 6.13. TD-CBF and GSC PSDs along −14- and 0-degree steering directions for a single multipath.

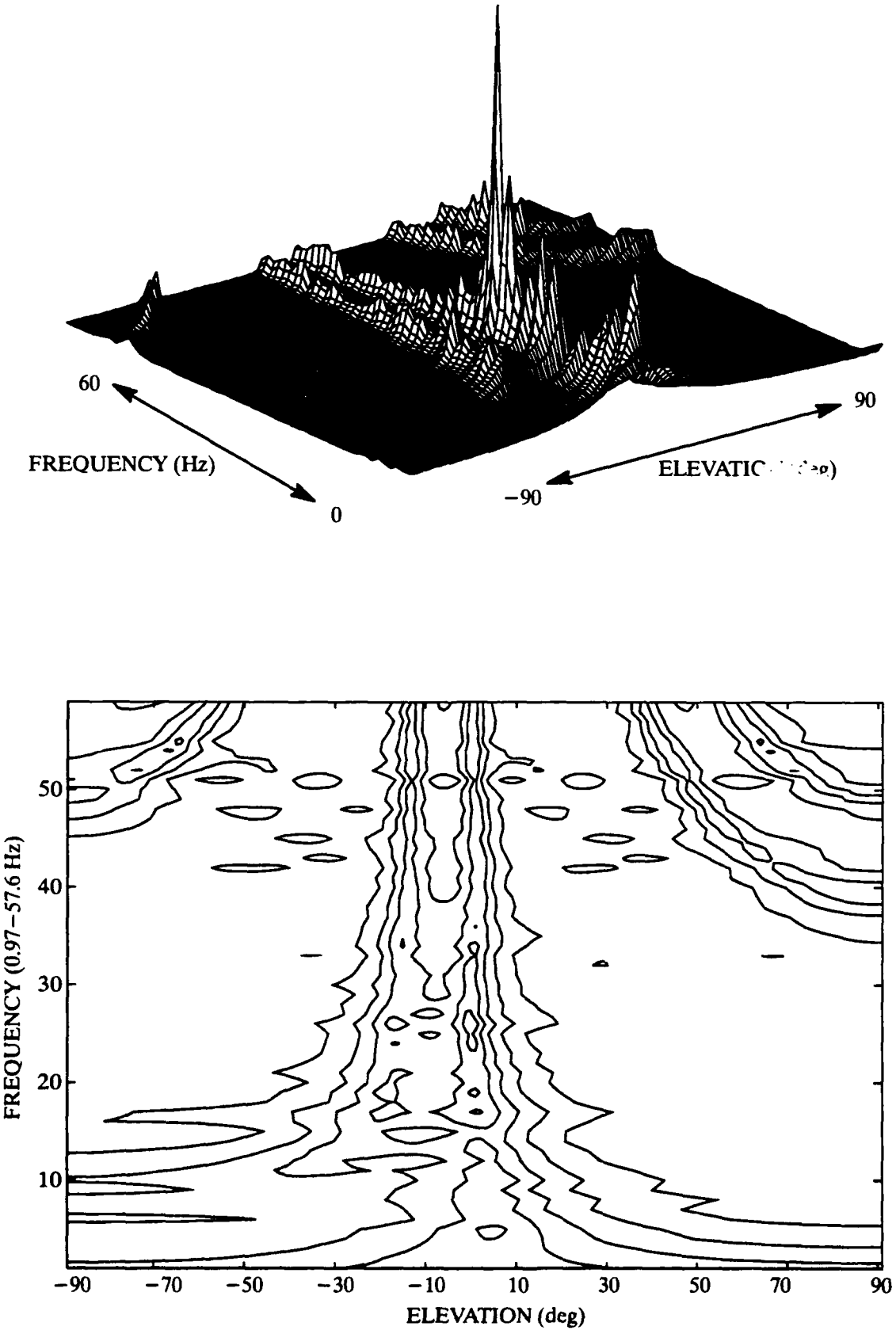
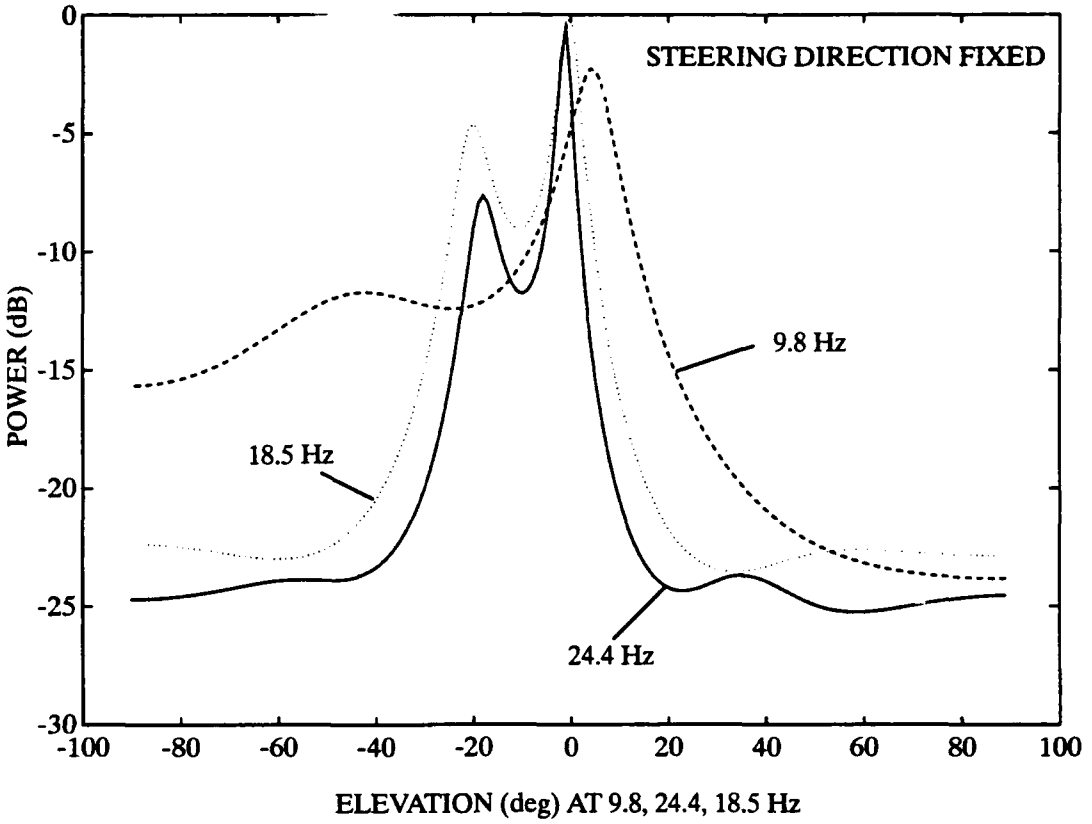


Figure 6.14. MVDR beamforming for a single multipath; FFT size of 1024 points.



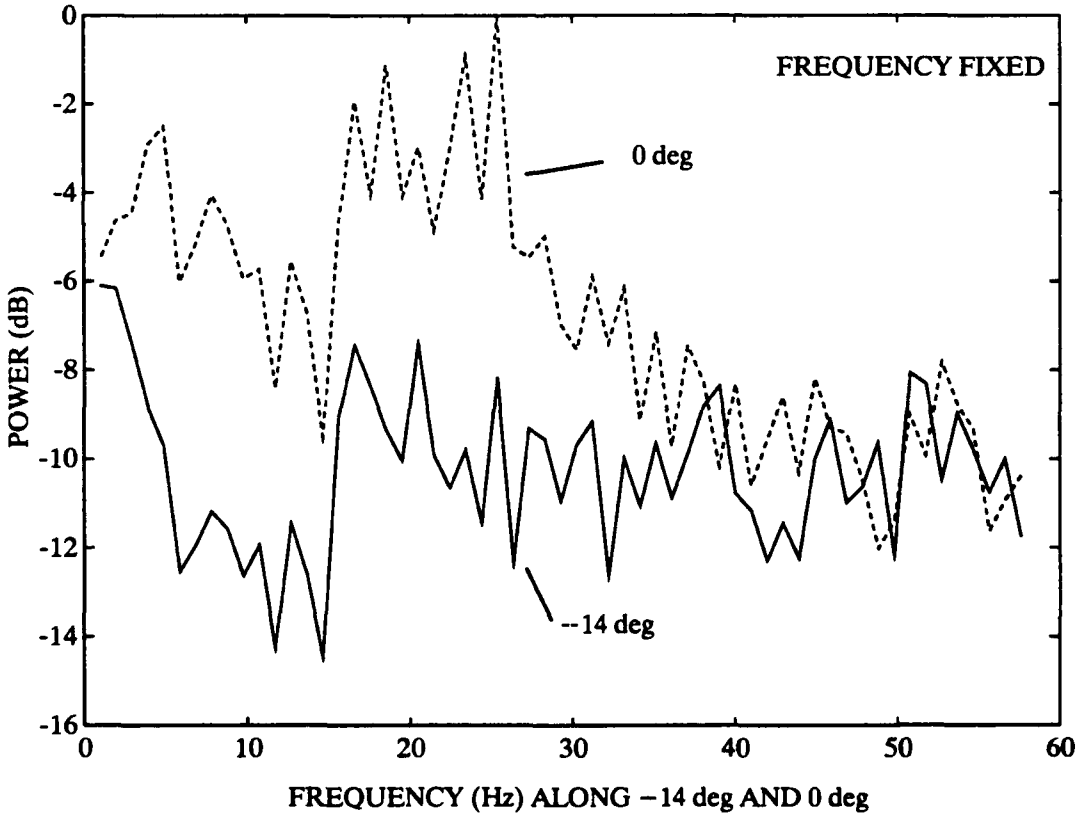


Figure 6.15. MVDR beamformer output for a single multipath along fixed steering directions (top) and along fixed processing frequencies (bottom).

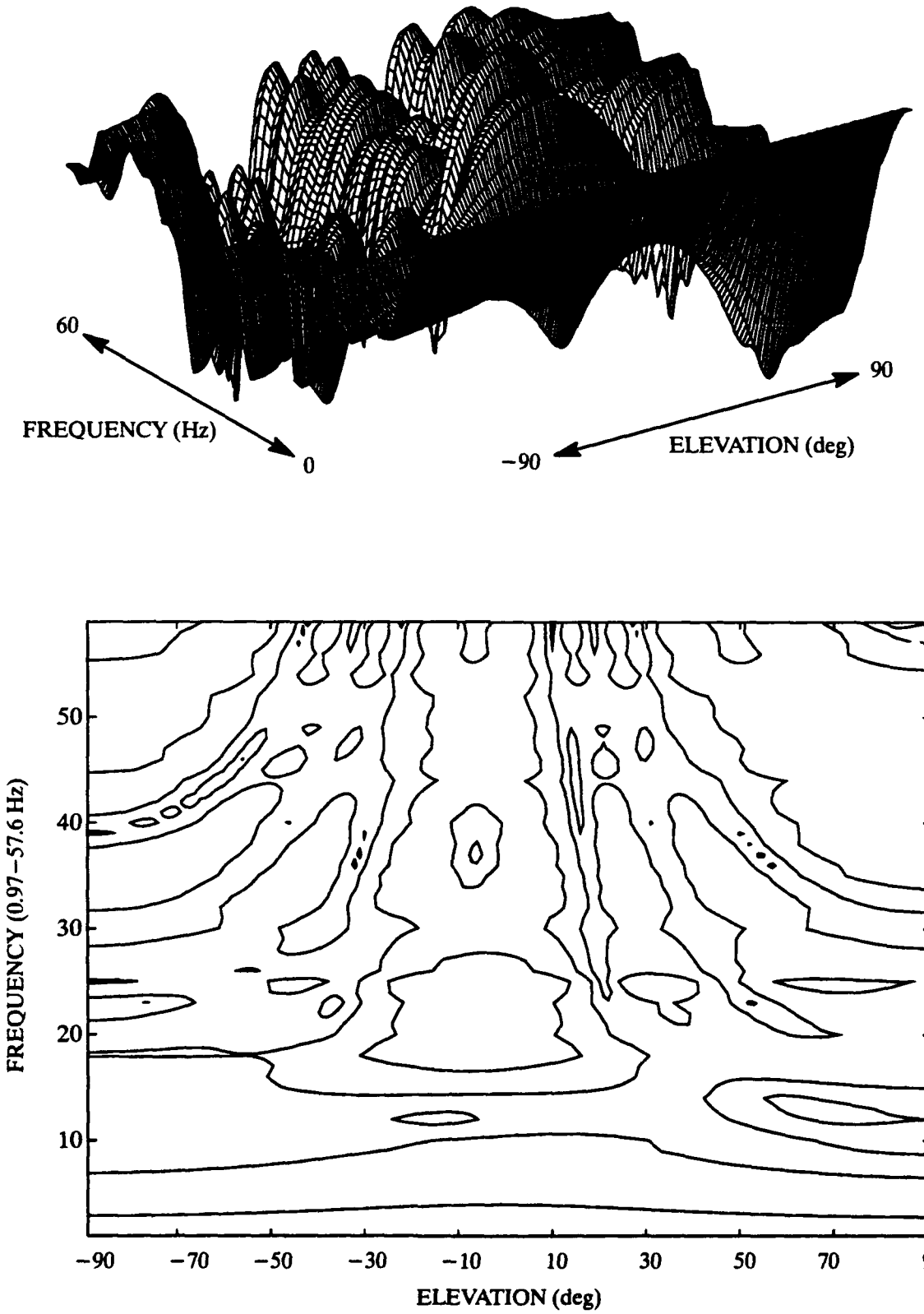


Figure 6.16. Bartlett beamforming for a single multipath; FFT size of 1024 points.

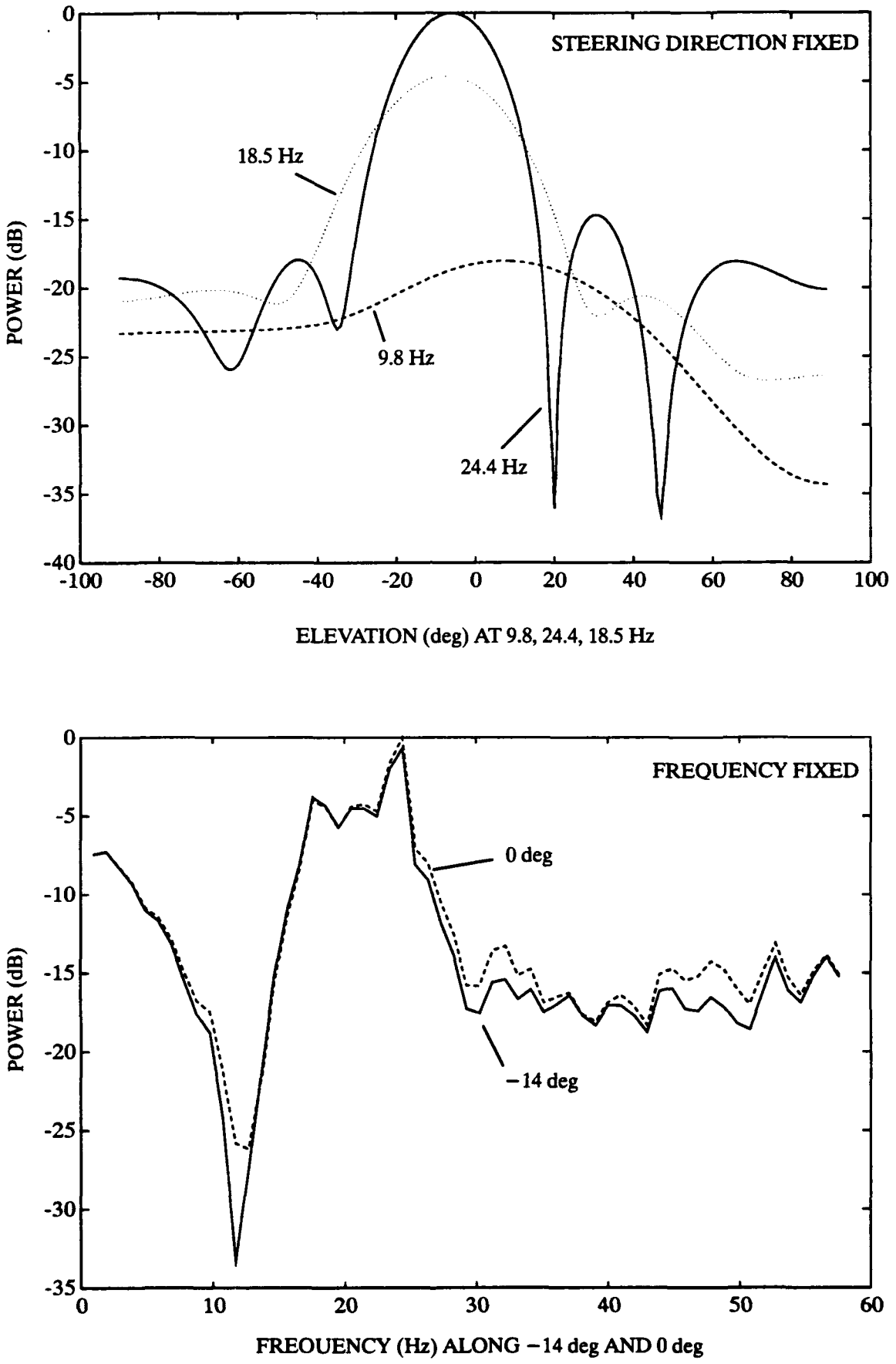
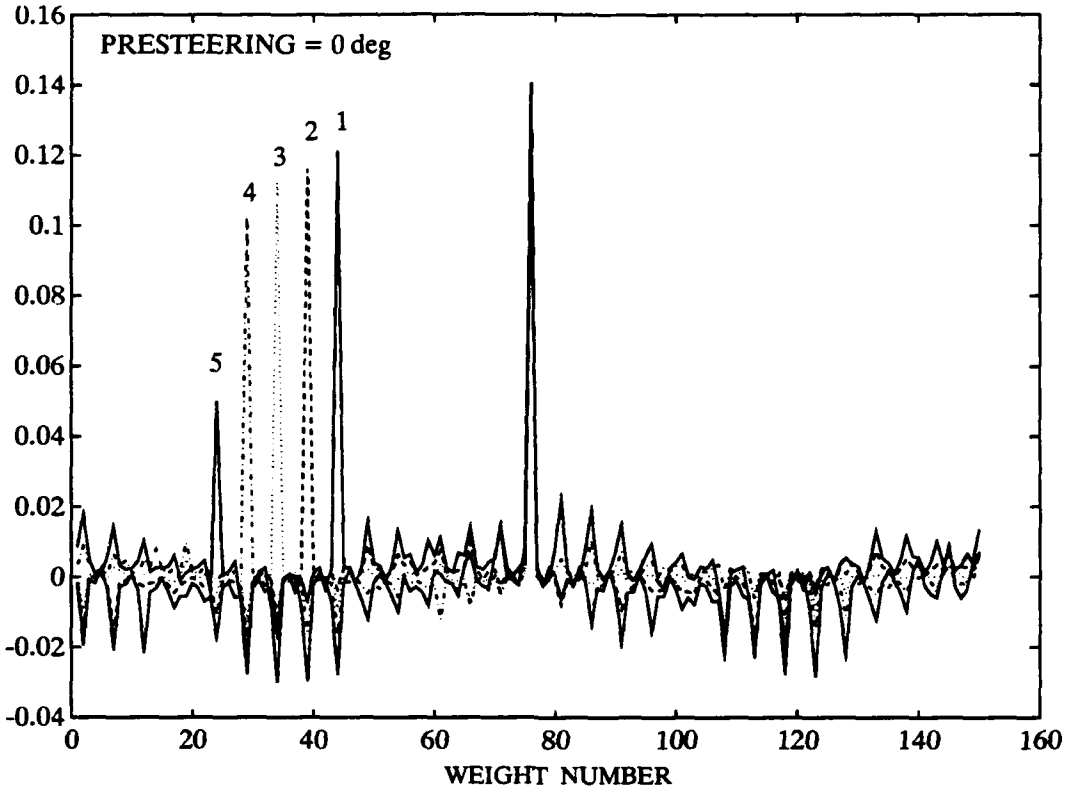


Figure 6.17. Bartlett beamformer output for a single multipath along fixed steering directions (top) and along fixed processing frequencies (bottom).



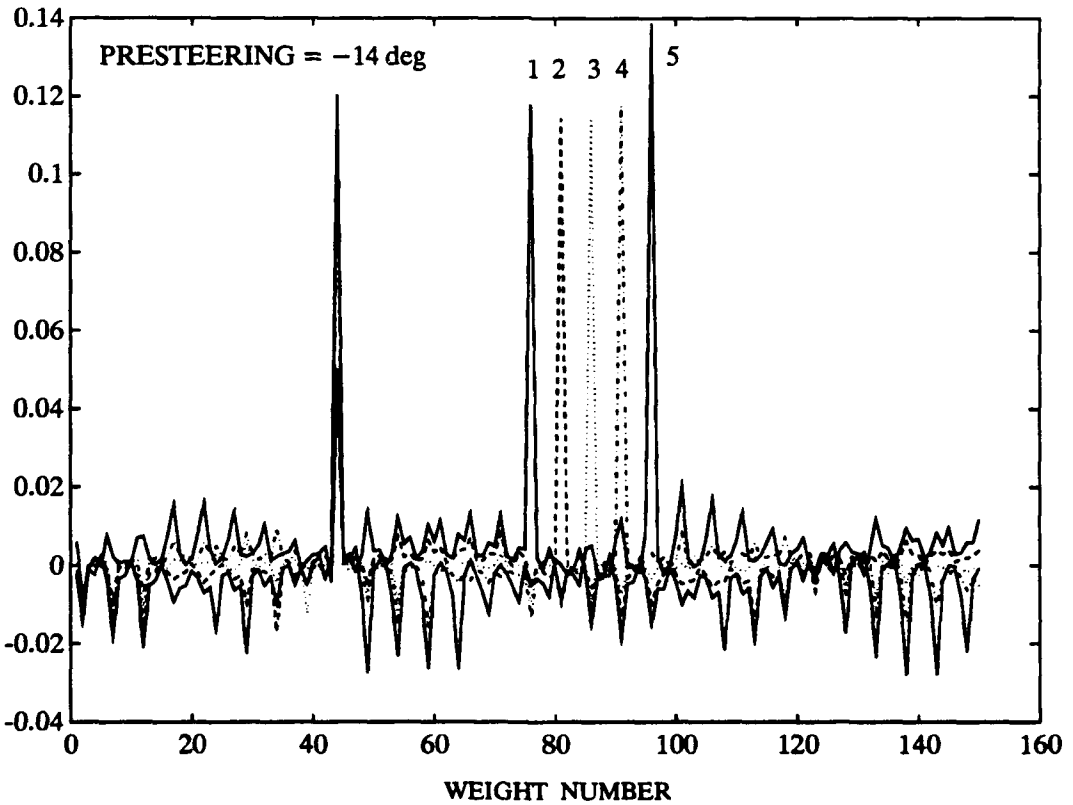


Figure 6.18. Multichannel adaptive filter weights for a single multipath: presteering at 0 (top) and −14 (bottom) degrees [$N_1 = N_2 = 75$].

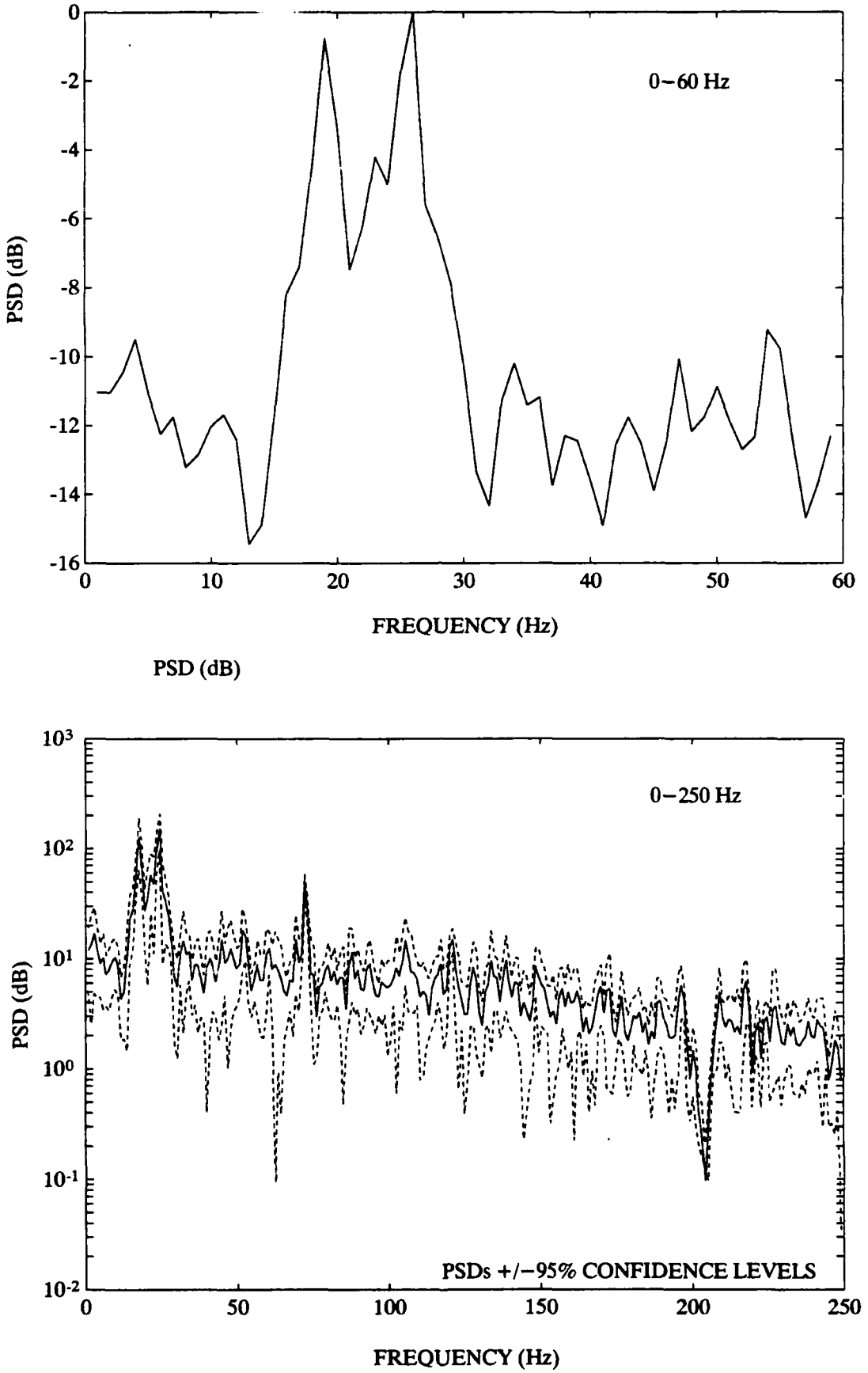
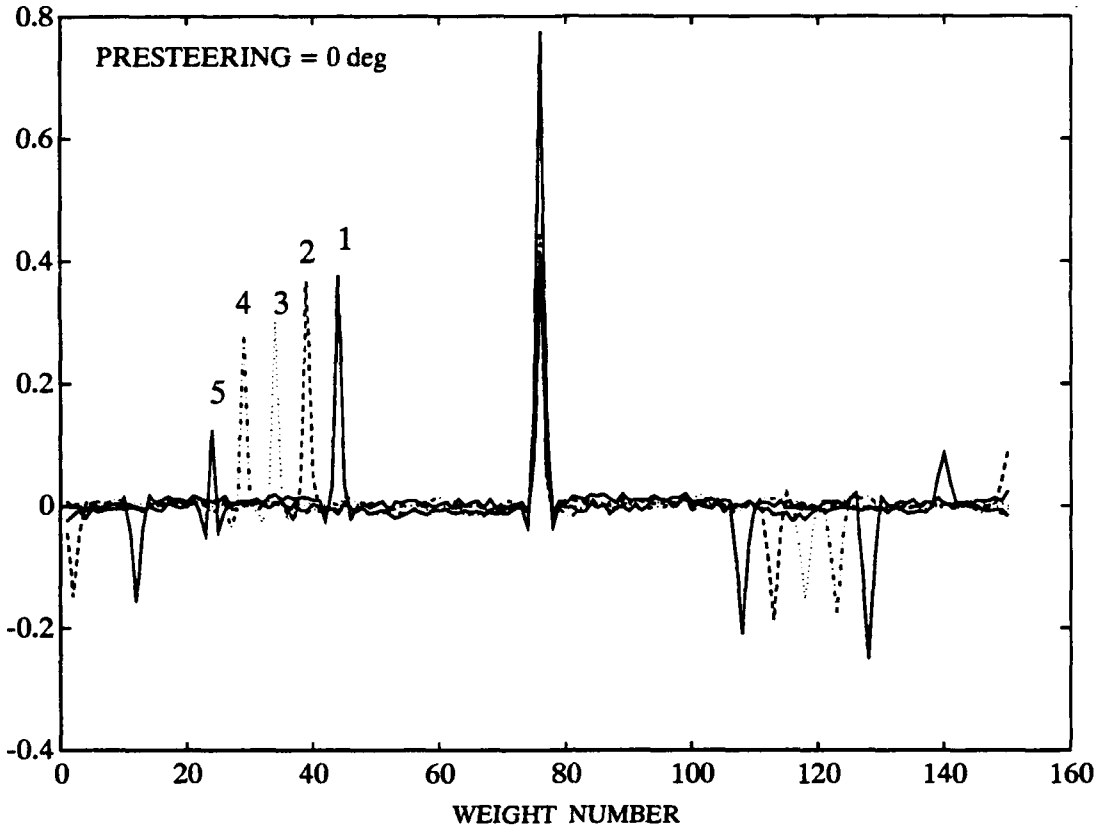


Figure 6.19. PSD of multichannel adaptive filter output for a single multipath: presteering of 0 degrees [$N_1 = N_2 = 75$].



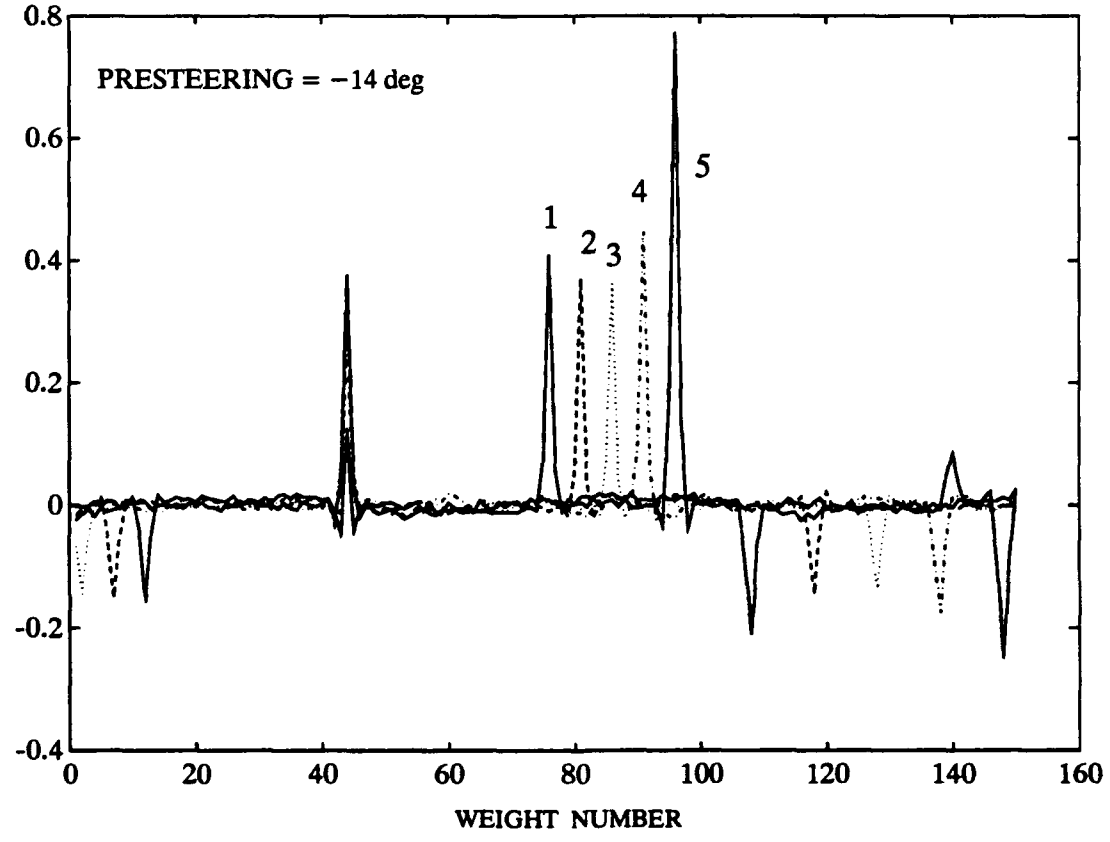


Figure 6.20. Weights of the multiple single-channel adaptive filter for a single multipath: presteering of 0 (top) and −14 (bottom) degrees [$N_1 = N_2 = 75$].

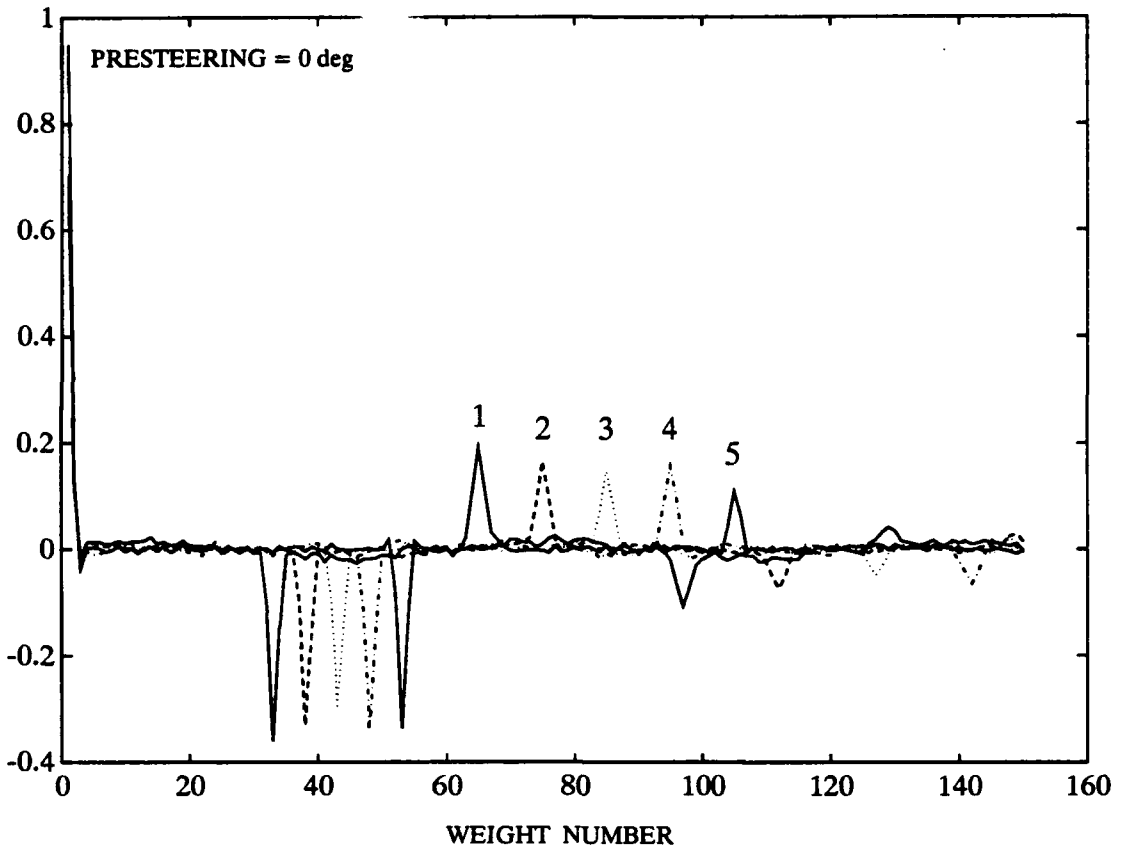


Figure 6.21. Weights of the multiple single-channel adaptive filter for a single multipath: presteering of 0 degrees [$N_1=150$, $N_2=0$].

7. CONCLUSION

This report has introduced and discussed the spatial–temporal processing capabilities of several adaptive and nonadaptive beamformers. Table 7.1 summarizes the results. Only the multichannel adaptive filter was found to be able to adaptively cancel uncorrelated interferers as well as to equalize the direct-path source with its correlated multipaths, true spatial–temporal processing. It was shown, however, that some a priori information about the source signal is needed for equalization and that the multichannel adaptive filter did not provide directionality information without a further processing of the weight values. The frequency-domain MVDR and the time-domain GSC beamformers were both shown to perform only spatial processing (adaptive interference cancellation) and no temporal processing. Furthermore, the performance of both algorithms degrades with the correlation of the interferers, greatly reducing the usefulness of these algorithms in a multipath environment. In short, further research is required to generate a true spatial–temporal beamformer. One possibility might be to incorporate blind equalization (equalization without a desired signal; see NO TAG) into the multichannel adaptive filter and develop a method of extracting directional information from the adapted weight matrix.

Table 7.1. Capabilities of beamforming algorithms.

Algorithm	Capabilities		
	Adaptive Interference Cancellation	Source-Element Equalization	Performance Affected by Source Correlation
Frequency Domain			
Bartlett	No	No	No
MVDR	Yes	No	Yes
Time Domain			
TD-CBF	No	No	No
Multi-Ch AF	Yes	Yes	No
GSC	Yes	No	Yes
Multiple Single-Ch AF	No	Yes	No

8. REFERENCES

General DSP

[1] D.J. DeFatta, J.G. Lucas and W.S. Hodgkiss, *Digital Signal Processing: A System Design Approach*, John Wiley & Sons, NY, 1988.

[2] F. J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proc. IEEE,* vol. 66, pp. 51–83, Jan. 1978.

[3] S.M. Kay and S.L. Marple, "Spectrum Analysis–A Modern Prospective," *Proc. IEEE,* vol. 69, no. 11, pp. 1380–1419, Nov. 1981.

[4] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1988.

General Adaptive Beamforming

[5] J.C. Allen, "Array Gain in Anisotropic Noise," Working Papers, Sept. 1992.

[6] J.C. Allen, "Direction Finding Database and Techniques (DFDT)," NOSC TN 1657, May 1991.

[7] W.S. Burdic, *Underwater Acoustic System Analysis,* Prentice–Hall, 1984.

[8] R.T. Compton, Jr., *Adaptive Antennas: Concepts and Performance,* Prentice–Hall, 1988.

[9] D.H. Johnson and D.E. Dudgeon, *Fundamentals of Array Signal Processing,* to be published, Prentice Hall, 1993.

[10] R. Schreiber, "Implementation of Adaptive Array Algorithms," *IEEE Trans. on Acoustic, Speech and Signal Processing,* vol. ASSP–34, no. 5, pp. 1038–1045, Oct. 1986.

[11] R.J. Urick, *Principles of Underwater Sound,* McGraw–Hill, New York, 1983.

Adaptive Equalization and Deconvolution

[12] R.A. Axford NRaD Code 844, *Notes and Conversations*, 1992.

[13] J.V. Candy, R.W. Ziolkowski, and D.K. Lewis, "Transient Wave Estimation: A Multichannel Deconvolution application," *J. Acoust. Soc. Am.,* vol. 88, no.5, pp. 2235–2247, 1990.

[14] S. Haykin, *Adaptive Filter Theory,* 2nd Edition, Prentice Hall, 1991.

[15] David Middleton, "Channel Modeling and Threshold Signal Processing in Underwater Acoustics: An Analytical Overview," *IEEE Journal of Oceanic Engineering*, vol. 12, no.1, pp. 4–28, 1987.

[16] J. Proakis, *Digital Communications*, McGraw–Hill, New York, Ch. 6 & 7, 1989.

[17] M. Stojanovic, J. Catipovic, and J.G. Proakis, "An Algorithm for Multichannel Coherent Digital Communications Over Long Range Underwater Acoustic Telemetry Channels," *preprint from J. Proakis, Northeastern Univ.*, July, 1991.

Curved Wavefronts Due to Nonuniform Sound Speed Profiles

[18] E. Webster NRAD Code 782, *Notes and Conversations,* 1992.

Time–Domain Beamforming

[19] M.G. Bellanger and P.A. Regalia, "The FLS–QR Algorithm For Adaptive Filtering: The Case of Multichannel Signals," *Signal Processing*, vol. 22, pp. 115–126, 1991.

[20] K.M. Buckley, "Spatial/Spectral Filtering with Linearly Constrained Minimum Variance Beamformers," *IEEE Trans. on SP*, vol. 40, no. 8, pp. 2112–2116, Aug. 1991.

[21] O.L. Frost, "An Algorithm for Linearly Constrained Adaptive Array Processing," *Proc. of the IEEE*, vol. 60, pp. 926–935, Aug. 1972.

[22] S. Godard, "Channel Equalization Using Kalman Filter for Fast Data Transmission," *IBM J. Res. Develop.*, pp. 267–273, May 1974.

[23] L.J. Griffiths, "Signal Extraction Using Real–Time Adaptation of Linear Multichannel Filter," *PhD Thesis (Tech Report No. 6788–1) Standford University*, Feb. 1968. see also L.J. Griffiths, "A Simple Adaptive Algorithm for Real–Time Processing in Antenna Arrays," *Proc. of the IEEE*, vol. 57, pp. 1696–1704, 1969.

[24] L.J. Griffiths, and C.W. Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming," *IEEE Trans. on Antennas and Propagation*, vol. 30, no. 1, pp. 27–34, Jan. 1982.

[25] F. Lung and J.G. Proakis, "A Generalized Multichannel Least Squares Lattice Algorithm Based on Sequential Processing Stages," *IEEE Trans. on Acoustic, Speech and Signal Processing*, vol. 32, no. 2, pp. 381–389, April 1984.

[26] R.G. Pridham and R.A. Mucci, "Digital Interpolation Beamforming for Lowpass and Bandpass Signals," *Proc. of the IEEE*, vol. 67, no. 6, pp. 904–919, June 1979.

[27] J. Ward and R.T. Compton, "Sidelobel Level Performance of Adaptive Sidelobe Canceller Arrays with Element Reuse," *IEEE Trans. on Antennas and Propagation*, vol. 38, no. 10, pp. 1684–1693, Oct. 1990.

[28] B. Widrow, P.E. Mantey, L.J. Griffiths, and B.B. Goode, "Adaptive Antenna Systems," *Proc. of the IEEE*, vol. 55, pp. 2143–2159, Dec. 1967.

[29] H. Ye and R.D. DeGroat, "A Generalized Sidelobe Canceller with Soft Constraints," *IEEE Trans. on ASSP*, vol. 35, no. 3, pp. 249–266, March 1987.

Frequency–Domain Beamforming

[30] BBN Systems and Technologies, "Issues in Adaptive Beamforming," NOSC Tech. Doc. 2105, May 1991.

[31] S.I. Chou, "A First Look at Narrowband Array Processing and Implementation for an Extra Large Multi–Dimensional Array," NOSC Tech. Note 1654, Mar. 1991.

[32] H. Cox, "Resolving Power and Sensitivity to Mismatch of Optimum Array Processors," *Journal of Acoustical Soc. of America*, vol. 54, no. 3, pp. 771–785, 1973.

[33] H. Cox, R.M. Zesking, and M.M. Owen, "Robust Adaptive Beamforming," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP−35, no. 10, pp. 1365−1376, 1987.

[34] M.D. Green (BBN), "STRAP: Baseline Definition and Performance Analysis," NOSC Tech. Doc. 888, April 1986.

[35] S.K. Lapic, J.C. Lockwood, and D.F. Gingras, "The Effects of the Conventional Beampattern on Adaptive Beamformer Performance," NOSC Tech. Doc. 1307, Dec. 1988.

[36] G.L. Mohnkern, "Effects of Errors and Limitations on Interference Suppression: Preliminary Report for the High Gain Initiative," NOSC Tech. Doc. 1478, Feb. 1989.

[37] M. Reuter, "Characterizing and Improving the Performance of the MVDR Processor," NOSC Tech. Doc. 1983, Nov. 1990.

[38] D.W.J. Stein, "Beamforming in the Presence of Moving Sources," NOSC Tech. Note 1655, March 1991.

[39] C.V. Tran, "Effects of Steering Vector Errors on MVDR and Signal Subspace Beamforming," NOSC Tech. Note 1649, Mar. 1991.

[40] J. Tran, "Approaches to the Processing of Data from Large Aperture Vertical Line Arrays," *PhD thesis,* University of California, San Diego, (also SIO reference 90−21), 1990.

Effects of Correlated Sources

[41] J. Krolik, and D. Swingler, "Multiple Broad−Band Source Location Using Steered Covariance Matrices," *IEEE Trans. on Acoustic, Speech and Signal Processing*, vol. 37, no. 10, pp. 1481−1494, Oct. 1989.

[42] B. Madan and S. Parker, "On Adaptive Beamforming in Correlated Noise," *Circuits Systems Signal Process*, vol. 7, no. 3, pp. 327−343, 1988.

[43] A. Paulraj and T. Kailath, "On Beamforming in Presence of Multipath," *Proc. of ICASSP−85*, pp. 564−567, 1985.

[44] V. Reddy, A. Paulraj, and T. Kailath, "Performance Analysis of the Optimum Beamformer in the Presence of Correlated Sources and Its Behavior Under Spatial Smoothing," *IEEE Trans. on Acoustic, Speech and Signal Processing*, vol. 35, no. 7, pp. 927−936, July 1987.

[45] T−J. Shan, M. Wax, and T. Kailath, "On Spatial Smoothing for Direction−of−Arrival Estimation of Coherent Signals," *IEEE Trans. on Acoustic, Speech and Signal Processing*, vol. 33, no. 4, pp. 806−811, Aug. 1985.

[46] K. Takao, N. Kikuma, and T. Yano, "Toeplitzization of Correlation Matrix in Multipath Environment," *Proc. of ICASSP−86*, pp. 1873−1876, 1986.

[47] B. Widrow, K. Duvall, R. Gooch, and W. Newman, "Signal Cancellation Phenomena in Adaptive Antennas: Causes and Cures," *IEEE Trans. on Antennas and Propagation*, vol. 30, no. 3, pp. 469−478, May 1982.

Appendix A
MATLAB ROUTINES

This Appendix describes the MATLAB routines used in this text to compare performance between various beamforming algorithms. MATLAB routines are patterned after those written by Jeff Allen [6]. The input–output vectors and notation have been matched to those used by Jeff Allen [6]. Figure A.1 highlights the interaction between the MATLAB programs. Element input data are typically stored in the time-domain array X, with T time-domain inputs from each element of an M-element array as

$$X = \overbrace{\begin{bmatrix} x_0(n) & x_0(n-1) & \cdots & x_0(n-T+1) \\ x_1(n) & x_1(n-1) & \cdots & x_1(n-T+1)) \\ x_2(n) & x_2(n-1) & \cdots & x_2(n-T+1) \\ \vdots & \vdots & \vdots & \vdots \\ x_{M-1}(n) & x_{M-1}(n-1) & \cdots & x_{M-1}(n-T+1) \end{bmatrix}}^{T \text{ time samples}} \Bigg\} M \text{ elements} \tag{A.1}$$

The M-element array (X,Y,Z) locations are stored in the array P:

$$P = \overbrace{\begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{M-1} \\ y_0 & y_1 & y_2 & \cdots & y_{M-1} \\ z_0 & z_1 & z_2 & \cdots & z_{M-1} \end{bmatrix}}^{\text{M elements}} \tag{A.2}$$

The steering matrix can be formed as time delays for multiple azimuthal angles, θ, and multiple elevation angles, ϕ:

$$A = \overbrace{\begin{bmatrix} \tau_1(\theta_0,\phi_0) & \tau_1(\theta_1,\phi_0) & \cdots & \tau_1(\theta_0,\phi_1) & \tau_1(\theta_1,\phi_1) & \cdots & \tau_1(\theta_0,\phi_2) & \tau_1(\theta_1,\phi_2) & \cdots \\ \tau_2(\theta_0,\phi_0) & \tau_2(\theta_1,\phi_0) & \cdots & \tau_2(\theta_0,\phi_1) & \tau_2(\theta_1,\phi_1) & \cdots & \tau_2(\theta_0,\phi_2) & \tau_2(\theta_1,\phi_2) & \cdots \\ \tau_3(\theta_0,\phi_0) & \tau_3(\theta_1,\phi_0) & \cdots & \tau_3(\theta_0,\phi_1) & \tau_3(\theta_1,\phi_1) & \cdots & \tau_3(\theta_0,\phi_2) & \tau_3(\theta_1,\phi_2) & \cdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & \end{bmatrix}}^{\text{n\_az*n\_el}} \Bigg\} M \text{ elements} \tag{A.3}$$

or phase delays

$$A = \exp\left(j\, 2\, \pi\, f \overbrace{\begin{bmatrix} \tau_1(\theta_0,\phi_0) & \tau_1(\theta_1,\phi_0) & \cdots & \tau_1(\theta_0,\phi_1) & \tau_1(\theta_1,\phi_1) & \cdots & \tau_1(\theta_0,\phi_2) & \tau_1(\theta_1,\phi_2) & \cdots \\ \tau_2(\theta_0,\phi_0) & \tau_2(\theta_1,\phi_0) & \cdots & \tau_2(\theta_0,\phi_1) & \tau_2(\theta_1,\phi_1) & \cdots & \tau_2(\theta_0,\phi_2) & \tau_2(\theta_1,\phi_2) & \cdots \\ \tau_3(\theta_0,\phi_0) & \tau_3(\theta_1,\phi_0) & \cdots & \tau_3(\theta_0,\phi_1) & \tau_3(\theta_1,\phi_1) & \cdots & \tau_3(\theta_0,\phi_2) & \tau_3(\theta_1,\phi_2) & \cdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & \end{bmatrix}}^{\text{n\_az*n\_el}} \Bigg\} M \text{ elements} \right) \tag{A.4}$$

The beamformer output power is stored in the array P_y (Y is the notation used in [6]) for multiple azimuthal angles, θ, and multiple elevation angles, ϕ,

$$P_y = \begin{bmatrix} y(\theta_0,\phi_0) & y(\theta_0,\phi_1) & y(\theta_0,\phi_2) & y(\theta_0,\phi_3) & \cdots \\ y(\theta_1,\phi_0) & y(\theta_1,\phi_1) & y(\theta_1,\phi_2) & y(\theta_1,\phi_3) & \cdots \\ y(\theta_2,\phi_0) & y(\theta_2,\phi_1) & y(\theta_2,\phi_2) & y(\theta_2,\phi_3) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{A.5}$$

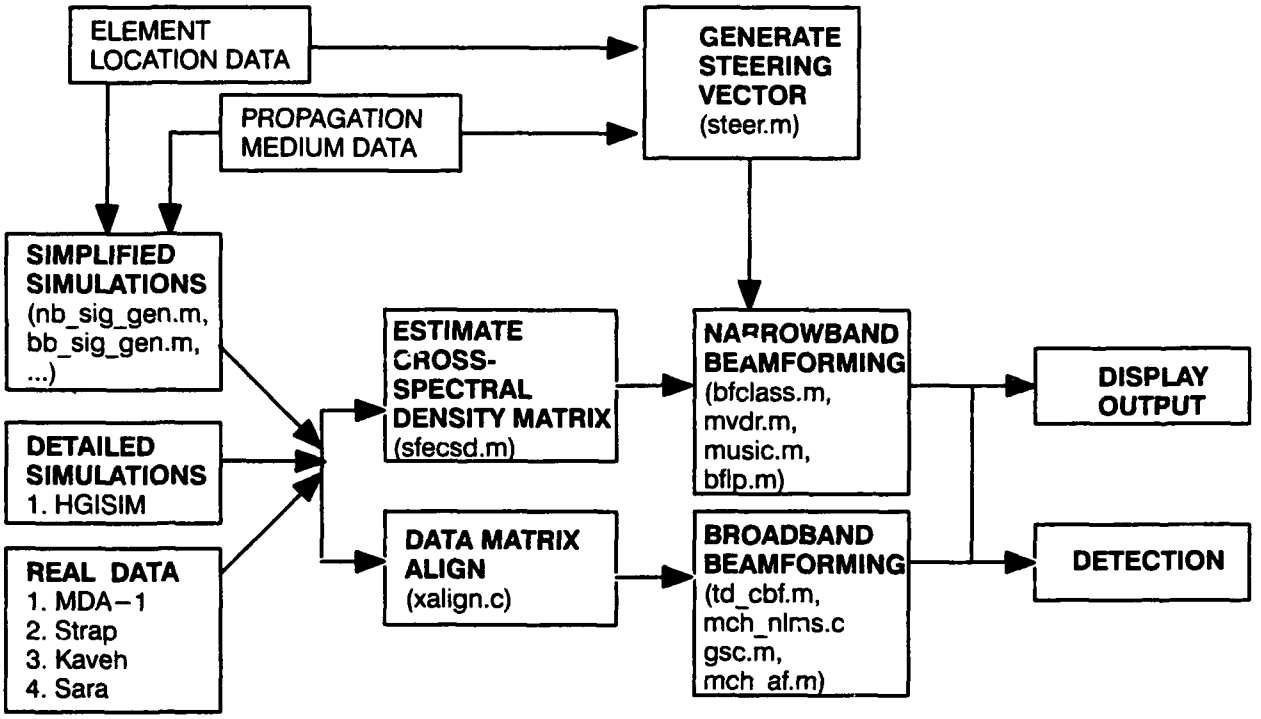


Figure A.1. Block diagram of beamforming signal processing computed with MATLAB using various sources of input data.

The following MATLAB programs were used extensively in this report and are listed in their entirety in Figure A.2.

1. **steer.m (modified slightly from [6])**
2. **bb_sig_gen.m**
3. **Xalign.c**
4. **td_cbf.m**
5. **mch_nlms.c**
6. **gsc.m**
7. **mch_af.m**
8. **arraypat.m**
9. **bfclass.m (modified slightly from [6])**
10. **mvdr.m (modified slightly from [6])**
11. **steer.m (modified slightly from [6])**

```
function A = steer( az , el , P , c , f0 , string)
%=========================================
% STEER  - creates an array of steering vectors for the given azimuth and
%          elevation angles as given in [1].
%
% INPUT:
%    az - vector of azimuth angles (deg)
%    el - vector of elevations angles (deg)
%     P - 3xM array of the M sensor positions
%         c - speed of signal propagation
%     f0 - frequency to beamform
% string - 'Phase' output phase delay (default if not defined)
%          'Time' output time delay (seconds)
%
% OUTPUT: let M denote the number of sensor; set
%    n_az = length(az)
%    n_el = length(el)
% Then the output array A is M x (n_az x n_el ) where each column has the
% form
%                                  T
%   string = 'Phase' (default): A(:,k) = [ exp( i 2 pi f0 P  u(k) / c ]
%
%   string = 'Time':          A(:,k) = [ P  u(k) / c ]
%
% where u(k) is the k-th look vector of length M determined as follows:
% factor k as
%
%    k = p * n_az + q       ( 0=< q < n_az )
%
% Then observing the MATLAB convention for indices starting at 1, we get
%
%                 | cos(az(q+1)) cos(el(p+1))  |
%       u(k+1) =  | sin(az(q+1)) cos(el(p+1))  |
%                 |              sin(el(p+1))  |
%
% REFERENCES:
% [1] Allen, Jeffery C.
%     "Direction Finding Database and Techniques (DFDT)"
%     Code 635, NOSC, San Diego CA 92152-5000, December 1990
%
% function A = steer( az , el , P , c , f0 , string)
%=========================================
```

Figure A.2. MATLAB code listings. (Contd)

```
n_az = length(az);
n_el = length(el);

u_az = [ cos(az*pi/180); sin(az*pi/180); ones(1,n_az)];
u_el = [ cos(el*pi/180); cos(el*pi/180); sin(el*pi/180)];

% for the k-th elevation angle, produce the look vector containing all
% the azimuth angles
A = zeros(3, n_az*n_el);
for k=1:n_el
  A(:, (n_az*(k-1)+1):(n_az*k)) = u_az .* (u_el(:,k) * ones(1,n_az));
end

% calculate delay
if (nargin < 6)
  A = exp( i*2*pi*f0*P'*A/c );
else
  if (strcmp( string, 'Phase' ))
    A = exp( i*2*pi*f0*P'*A/c );
  elseif strcmp( string, 'Time' )
    A = P'*A/c;
  else
    error(' ... string set incorrectly (Phase or Time) ...')
  end
end
```

bb_sig_gen.m

```
function s=bb_sig_gen(T,nvar1,nvar2,f0,f1,fs)
%**********************************************
% BB_SIG_GEN:
% This matlab function generates a broad band signal with odd
% harmonics.
%
% INPUTS:
%    T - number of data samples (T>2*N)
%  nvar1 - variance of the noise, var[n1(n)]
%  nvar2 - variance of the noise, var[n2(n)]
%    f0 - primary sinusoid frequency (+harmonics)
%    f1 - secondary sinusoid frequency
%    fs - sampling frequency
%
% OUTPUTs:
%    s - 1xT vector of broad band data
%
% ALGORITHM:
%
%                        Amp Modulation
%                             c+a(n)
%          |------|             |
%    n1(n)---| BPF |-f(n)- * --- + --s(n)
%          |______|             |
%                               |
%          |------|             |
%    n2(n)---| LPF |-l(n)------ +
%          |______|             |
%                               |
%         sin(f0/fs)------------ +
%                               |
%         sin(3*f0/fs)---------- +
%                               |
%         sin(5*f0/fs)---------- +
%                               |
%         sin(f1/fs)------------|
%
%
% s=bb_sig_gen(T,nvar1,nvar2,f0,f1,fs)
% written by: Richard North 6/92
%**********************************************

%----------------
% initialize variables
%----------------
t = 0:1:T-1; % t = 0,1,2,3,...,T-1 time
anoise1 = sqrt(nvar1);
anoise2 = sqrt(nvar2);
s = zeros(1,T);
rand('normal');

%----------------
% generate amplitude modulated bandpass noise
%----------------
n = anoise1*rand(t);          % random noise
flow = 15/(fs/2);
fhigh = 25/(fs/2);
[b,a]=butter(5,[flow fhigh]);  % 10th ordered band pass filter
fn = filter(b,a,n);           % band pass filtered noise
C = 0.5;                      % C=0.0 -> no mod
s = (ones(t) +C*cos(2*pi*2/fs*t)).*fn;  % amp mod (2 Hz) bpf noise

%----------------
% generate exponentially filtered low pass noise
%----------------
n = anoise2*rand(t);          % random noise
f = [ 0:10 ]/10;
m = [ exp(-(0:10)/5) ];
b=fir2(10,f,m);               % 10th ordered low pass filter
a=[1];
s = s + filter(b,a,n);   % signal + filtered noise

%----------------
% generate sinusoids
%----------------
for i = 1:2:5
  s = s + 0.8 ^ i*cos(2*pi*i*f0/fs*t);  % sinusoid
end;
phase = pi/4.6;  % any constant phase
s = s + 0.6*cos(2*pi*f1/fs*t + phase); % sinusoid
```

Xalign.c

```
/**********************************************************
* XALIGN:
* This matlab mex function delays and weights the time series
* data from multiple sensors and stores the aligned data for
* more processing.
*
* USAGE FROM MATLAB:
*      Xa=Xalign(X,tau)
*
* TO COMPILE:
*      /matlab/bin/cmex Xalign.c
*
* INPUTS:
*  X - MxT matrix where each row consists of the time series
*      data from a single element
*  M - number of elements
*  T - number of data sampels
* TAU - Mx1 matrix of delays
*
* OUTPUTS:
*  Xa - MxT matrix of weighted and delayed X elements
*
* ALGORITHM:
*
*        = X(m,t-tau(m,u))
*  Xa(m,t)
*        = 0  (if X(m,t-tau(m,u)) not in matrix)
*
* REFERENCE:
*   R. North, "Broadband Conventional Beamforming Incorporating
*     Adaptive Equalization," NRaD Tech. Note 1992
*
* written by: Richard North 5/92
**********************************************************/
#include <math.h>
#include <stdio.h>
#include "cmex.h"

/* input arguments */
#define X_IN prhs[0]
#define TAU_IN  prhs[1]

/* output arguments */
#define Xaligned_OUT plhs[0]

user_fcn(nlhs,plhs,nrhs,prhs)
int nlhs,nrhs;
Matrix *plhs[], *prhs[];
{
  /* Matlab variables */
  unsigned int M,Nlooks,T;
  double *X_re,*X_im,*TAU,*Xaligned_re,*Xaligned_im;

  /* misc local variables */
  int t,m,n,a,delay,delay_a;

  /* check number of arguments */
  if (nrhs != 2) mex_error("Xalign requires 2 inputs.");
  if (nlhs != 1) mex_error("Xalign requires 1 output.");

  /* set variables and allocate memory */
  M = TAU_IN->m;    /* # of rows equals # of elements */
  if (X_IN->m != M)
    mex_error("Xalign requires X, and tau to have same number of elements.");
  Nlooks = TAU_IN->n;  /* # of columns equals # of steer directions */
  if (Nlooks != 1)
    mex_error("Xalign requires only 1 look direction.");
  T = X_IN->n;      /* # of columns equals # of time samples */

  X_re = X_IN->pr;
  X_im = X_IN->pi;
  TAU = TAU_IN->pr;

  if (X_im == NULL) /* real */
    {
    Xaligned_OUT = create_matrix(M,T,REAL);
    Xaligned_re = Xaligned_OUT->pr;
    }
  else              /* complex */
    {
    Xaligned_OUT = create_matrix(M,T,COMPLEX);
    Xaligned_re = Xaligned_OUT->pr;
    Xaligned_im = Xaligned_OUT->pi;
    }

  if (X_im == NULL) /* real */
    {
    /* do delay and weighting for each time t */
    /* NOTE: Matlab stores by columns so that X(i,j)=X[i+TotalNrows*j]
          If we only have vectors, this is transparent */
```

Figure A.2. Continued. (Contd)

```
    for (t=0; t<T; t++)
                    {
      /* across elements */
      for (m=0; m<M; m++)
                    {
        a = t-(int)TAU[m];
                    if ((a >= 0) && (a < T))
                      Xaligned_re[m+M*t] = X_re[m+M*a];
        else
                      Xaligned_re[m+M*t] = 0;
                    } /* end m */

      } /* end t */
  }
  else          /* complex */
   {
   /* do delay and weighting for each time t */
   /* NOTE: Matlab stores by columns so that X(i,j)=X[i+TotalNrows*j]
        If we only have vectors, this is transparent */
    for (t=0; t<T; t++)
                    {
      /* across elements */
      for (m=0; m<M; m++)
                    {
        a = t-(int)TAU[m];
                    delay_a = m+M*t;  /* new matrix's location */
                    if ((a >= 0) && (a < T))
                      {
                      delay = m+M*a;  /* location within old matrix */
                      Xaligned_re[delay_a] = X_re[delay];
                      Xaligned_im[delay_a] = X_im[delay];
                      }
        else
          {
                      Xaligned_re[delay_a] = 0;
                      Xaligned_im[delay_a] = 0;
          }
                    } /* end m */

      } /* end t */
   }

}
```

td_cbf.m

```
function [Py,Y] = td_cbf( X, az, el, P, dt, c)
%=========================================
% TD_CBF - classical time domain "delay and sum" beamforming
%
% INPUT:
%    X  - MxT array of T time samples of the M sensor outputs
%    az - beamforming azimuth angles (degrees)
%    el - beamforming elevation angles (degrees)
%    P  - 3xM array of the M sensor positions (ft) (m)
%    dt - sample increment (sec)
%    c  - speed of sound in propagation media (ft/sec) (m/sec)
%
% OUTPUT:
%    Py - array of classical beamformer power output; Py(k,l) is the output
%         at azimuth angle az(l) and elevation angle el(k).
%    Y  - NlooksxT array of T time samples of the beamformer output
%
% ALGORITHM:
% As discussed in [1], the time domain classical beamformer has output
%                    H
%    y(t) = [sum {m=1}^{M} X(m:t-tau_m)] [sum {m=1}^{M} X(m:t-tau_m)]
%
% for steering delay vector
%          H
%    a = [p_{m} u/c)] (seconds)
%
% where p_{m} is the position vector of the mth sensor, u is the look
% direction.
%
% REMARKS:
% (1) to beamform (ie delay) in a arbitrary desired direction requires
%     interpolation due to the discrete nature of the input time samples.
%     ( For other techniques see R. Pridham and R. Mucci, "Digital
%       Interpolation Beamforming for Low-Pass and Bandpass Signals"
%       Proc. of IEEE, vol. 67, no. 6, pp. 904-919, June 1979. )
% (2) y' is the conjugate transpose of the vector y
%
% REFERENCES:
%    [1] D.H. Johnson and D.E. Dudgeon,
%        "Fundamentals of Array Signal Processing",
%        Prentice-Hall to be published in 1993.
%
% function [Py,y] = td_cbf( X, az, el, P, dt, c)
%=========================================

%=========================================
% initialize variables
%=========================================
[M,T] = size(X);
fs = 1/dt;   % sampling frequency
n_az = length(az);
n_el = length(el);
Nlooks = n_az*n_el;
%W = ones(M,1);                  % unity sensor weighting
Y = zeros(Nlooks,T);
Py = zeros(1,Nlooks);
x = zeros(T,1);

%=========================================
% Generate steering time delays for all directions of look
%=========================================
A = steer( az , el, P , c , 1 , 'Time');  % delays in seconds (freq not used)
TAU = round(A*fs);                  % delays in iterations

%=========================================
% align data matrix and equalize channels for each look direction
%=========================================
for n=1:Nlooks
  % align all element data for look direction n
  Xa = xalign(X,TAU(:,n));
  % weight all element data for look direction n
  %Xa = diag(W)*Xa;

  % equalize each element's channel i
  Y(n,:) = sum(Xa);
end;

Py = mean(Y'.*conj(Y'));

%=========================================
% tidy up and reshape the output;
%=========================================
% we know Py must be real ...
Py = real(Py);
% ... and return y with the rows containing the azimuth for a fixed
% elevation angle.
Py = reshape( Py , n_az , n_el )';

%=========================================
```

mch_nlms.c

```
/*****************************************************************
* MCH_NLMS.C:
* This matlab mex function uses the normalized LMS algorithm to
* filter the multiple channel reference input X to generate the
* filtered output y.  y is subtracted from the primary input, d
* to generate the error output, err.
*
* USAGE FROM MATLAB:
*      [y,err,wts]=mch_nlms(X,d,N,mu,alpha)
*
* TO COMPILE:
*      /matlab/bin/cmex mch_nlms.c
*
* INPUTS:
*     X - MxT vector of reference inputs
*     T - number of data samples
*     d - 1xT vector of primary input
*     N - number of filter weights/per channel
*    mu - LMS step size
*  alpha - leak parameter
*        (alpha=0 results in "standard LMS"
*         1/(1-2mu)>alpha>0 results in "leaky LMS")
*
* OUTPUTs:
*     y - 1xT vector of LMS filter outputs
*   err - 1xT vector of LMS error outputs
*   wts - MxN vector of the final converged filter weights
*
* ALGORITHM:
* The Leaky/Std LMS algorithm is defined as follows at time k,
*   1. y(k) = sum {m=1}^M sum {i=1}^N conj(wts(m,i))*x(m,k-i)
*   2. err(k) = d(k) - y(k)
*   3. Ex(k) = W*Ex(k-1) + (1-W)*sum {m=1}^M |x(k)|^2
*   4. mu' = (2*mu) / (eps + N*Ex(k))
*   5. wts(k+1|m,p) = (1-mu'*alpha)*wts(k|m,p) +
*                     mu'*conj(err(k))*x(m,k-p)
*
* For alpha=0, the Leaky algorithm reduces to the Std LMS
* algorithm.  The filter is configured as an Adaptive Noise
* Canceller.
*
* REFERENCE:
*   S. Haykin, "Adaptive Filter Theory,", Prentice Hall, 1991.
*
* written by: Richard North 8/92
*****************************************************************/
#include <math.h>
#include <stdio.h>
#include "cmex.h"

#define MAX_N 151  /* Maximum filter Length for Momentum LMS */
#define EPS  0.000001  /* Minimum number = 1x10-6 */

/* input arguments */
#define X_IN prhs[0]
#define D_IN  prhs[1]
#define N_IN  prhs[2]
#define mu_IN  prhs[3]
#define alpha_IN  prhs[4]

/* output arguments */
#define Ypred_OUT plhs[0]
```

Figure A.2. Continued. (Contd)

```
#define Err_OUT plhs[1]
#define Wts_OUT plhs[2]

user_fcn(nlhs,plhs,nrhs,prhs)
int nlhs,nrhs;
Matrix *plhs[], *prhs[];
{
  /* Matlab variables */
  unsigned int T;
  double N,mu,alpha;
  double *X_re,*D_re,*Ypred_re,*Err_re,*Wts_re;
  double *X_im,*D_im,*Ypred_im,*Err_im,*Wts_im;
  double Wts_temp[MAX_N],Wts_old[MAX_N];

  /* misc local variables */
  int t,m,n,d1,d2,M;
  double Ex,norm_mu,W=0.999;

  /* check number of arguments */
  if (nrhs != 5) mex_error("'lms' requires 5 inputs.");
  if ((nlhs > 2) && (nlhs != 3)) mex_error("'lms' requires 3 outputs.");
  if (nlhs < 2) mex_error("'lms' requires at least 2 outputs.");

  /* set variables and allocate memory */
  N = *N_IN->pr;        /* # of filter taps */
  if (N > MAX_N) mex_error("'lms' N > MAX_N need to recompile lms.c");
  mu = *mu_IN->pr;      /* value of step size */
  alpha = *alpha_IN->pr; /* value of leak parameter */
  T = X_IN->n;          /* # of columns equals T */
  M = X_IN->m;          /* # of rows equals M (no. of channels) */

  X_re = X_IN->pr;
  X_im = X_IN->pi;
  D_re = D_IN->pr;
  D_im = D_IN->pi;

  if ((X_im == NULL) && (D_im != NULL)) /* x is real & d is complex */
    mex_error("x is real & d is complex");
  if ((X_im != NULL) && (D_im == NULL)) /* x is complex & d is real */
    mex_error("x is complex & d is real");

  if (X_im == NULL)  /* real */
    {
    Ypred_OUT = create_matrix(1,T,REAL);
    Err_OUT = create_matrix(1,T,REAL);
    Wts_OUT = create_matrix(M,(int)N,REAL);

    Ypred_re = Ypred_OUT->pr;
    Err_re = Err_OUT->pr;
    Wts_re = Wts_OUT->pr;
    }
  else          /* complex */
    {
    Ypred_OUT = create_matrix(1,T,COMPLEX);
    Err_OUT = create_matrix(1,T,COMPLEX);
    Wts_OUT = create_matrix(M,(int)N,COMPLEX);

    Ypred_re = Ypred_OUT->pr;
    Ypred_im = Ypred_OUT->pi;
    Err_re = Err_OUT->pr;
    Err_im = Err_OUT->pi;
    Wts_re = Wts_OUT->pr;
    Wts_im = Wts_OUT->pi;
    }

  if (X_im == NULL)  /* real */
    {
    /* NOTE: Matlab stores by columns so that X(i,j)=X[i+Nrows*j] */
    /* initialize real variables */
    for (t=0; t<T; t++)
      Ypred_re[t] = 0.0;
    for (m=0; m<M; m++)
      for (n=0; n<N; n++)
        Wts_re[m+M*n] = 0.0;
    Ex = 0;
    for (t=0; t<(int)N-1; t++)
                    for (m=0; m<M; m++)
      Ex = W*Ex + (1-W)*(X_re[m+M*t]*X_re[m+M*t]);

    /* filter real data with real LMS */
    for (t=(int)N-1; t<T; t++)
      {
                    for (m=0; m<M; m++)
                      {
      for (n=0; n<(int)N; n++)
                        {
        Ypred_re[t] += Wts_re[m+M*n]*X_re[m+M*(t-n)];
                        }
      Ex = W*Ex + (1-W)*(X_re[m+M*t]*X_re[m+M*t]);
                      }
      Err_re[t] = D_re[t]-Ypred_re[t];
      if (Ex > EPS)
        norm_mu = (2*mu) / (N*Ex);
      else
        norm_mu = (2*mu) / (N*EPS);
                    for (m=0; m<M; m++)
                      {
      for (n=0; n<(int)N; n++)
        {
        /* Leaky/Std LMS */
        Wts_re[m+M*n] = (1.0 - norm_mu*alpha)*Wts_re[m+M*n] +
                        norm_mu*Err_re[t]*X_re[m+M*(t-n)];
        }
      }
    }
  else          /* complex */
    {
    /* NOTE: Matlab stores by columns so that X(i,j)=X[i+Nrows*j] */
    /* initialize complex variables */
    for (t=0; t<T; t++)
      {
      Ypred_re[t] = 0.0;
      Ypred_im[t] = 0.0;
      }
    for (m=0; m<M; m++)
      for (n=0; n<N; n++)
                      {
        Wts_re[m+M*n] = 0.0;
        Wts_im[m+M*n] = 0.0;
                      }
    Ex = 0;
    for (t=0; t<(int)N-1; t++)
                    for (m=0; m<M; m++)
                      {
      d1 = m+M*t;
      Ex = W*Ex + (1-W)*(X_re[d1]*X_re[d1] + X_im[d1]*X_im[d1]);
                      }

    /* filter complex data with complex LMS */
    for (t=(int)N-1; t<T; t++)
      {
                    for (m=0; m<M; m++)
                      {
      for (n=0; n<(int)N; n++)
        {
                      d1 = m+M*n;
                      d2 = m+M*(t-n);
        Ypred_re[t] += Wts_re[d1]*X_re[d2] + Wts_im[d1]*X_im[d2];
        Ypred_im[t] += Wts_re[d1]*X_im[d2] - Wts_im[d1]*X_re[d2];
        }
      d1 = m+M*t;
      Ex = W*Ex + (1-W)*(X_re[d1]*X_re[d1] + X_im[d1]*X_im[d1]);
      }
    Err_re[t] = D_re[t]-Ypred_re[t];
    Err_im[t] = D_im[t]-Ypred_im[t];
    if (Ex > EPS)
      norm_mu = 2*mu / (N*Ex);
    else
      norm_mu = 2*mu / (N*EPS);
                    for (m=0; m<M; m++)
                      {
    for (n=0; n<(int)N; n++)
      {
                      d1 = m+M*n;
                      d2 = m+M*(t-n);
      /* Leaky/Std LMS */
      Wts_re[d1] = (1.0 - norm_mu*alpha)*Wts_re[d1] +
          norm_mu*(Err_re[t]*X_re[d2] + Err_im[t]*X_im[d2]);
      Wts_im[d1] = (1.0 - norm_mu*alpha)*Wts_im[d1] +
          norm_mu*(Err_re[t]*X_im[d2] - Err_im[t]*X_re[d2]);
      }
    }
  }
}
```

gsc.m

```
function [Py,Y] = gsc( X, az, el, P, dt, c, N, mu, B)
%=========================================
% GSC - Generalized Sidelobe Canceller (Linear Constrained Adaptive
%       Beamformer)
%
% INPUT:
%   X  - MxT array of T time samples of the M sensor outputs
%   az - beamforming azimuth angles (degrees)
%   el - beamforming elevation angles (degrees)
%   P  - 3xM array of the M sensor positions (ft) (m)
%   dt - sample increment (sec)
%   c  - speed of sound in propagation media (ft/sec) (m/sec)
%   N  - channel filter length (same for all elements)
%   mu - LMS step size
%   B  - JxM blocking matrix
%        (where J < M is the number of sidelobe cancellers)
%
% OUTPUT:
%   Py - array of beamformer power output; Py(k,l) is the output
%        at azimuth angle az(l) and elevation angle el(k).
%   Y  - NlookxxT array of T time samples of the beamformer output
%
% ALGORITHM:
% As discussed in [1] & [2], the Linear Constrained Adaptive Beamformer
% is implemented iteratively as a Generalized Sidelobe Canceller (GSC).
% It consists of a nonadaptive time domain classical beamformer followed
% by a FIR filter.  To cancel interferers which bleed in through the CBF
% sidelobes, a sidelobe canceller (consisiting of a blocking matrix and
% a multichannel adaptive filter) adaptively filters the interferes in
% space and time to subtract from the CBF output.
%
% NOTE:
% Two possibilities for B are [1]
%   1. col = [ 1 zeros(1,M-2)];    2. col = [ 1 zeros(1,M-3)];
```

Figure A.2. Continued. (Contd)

```
%       row = [ 1 -1 zeros(1,M-2)];     row = [ 1 -2 1 zeros(1,M-3)];
%               B = toeplitz(col,row);
%
% REFERENCES:
% [1] L. Griffiths, C.W. Jim,"An Alternative Approach to Linear Constrained
%       Adaptive Beamforming"
%       IEEE Trans. Antennas and Prop., vol. AP-30, pp. 27-34, 1982.
% [2] R. North, "Broadband Conventional Beamforming Incorporating
%       Adaptive Equalization," NRaD Tech. Note 1992
%
% function [Py,Y] = gsc( X, az, el, P, dt, c, N, mu, B)
%=========================================

%=========================================
% initialize variables
%=========================================
[M,T] = size(X);
[J,M1] = size(B);
if ((M1 ~= M) | (J >= M))
  error('Blocking Matrix, B, wrong size!');
end;
fs = 1/dt;   % sampling frequency
n_az = length(az);
n_el = length(el);
Nlooks = n_az*n_el;
%W = ones(M,1);       % unity sensor weighting for CBF (see also chebwin)
Y = zeros(Nlooks,T);
Yc = zeros(1,T);
Ya = zeros(1,T);
Py = zeros(1,Nlooks);
Xa = zeros(M,T);
Xp = zeros(J,T);

%=========================================
% Generate steering time delays for all directions of look
%=========================================
A = steer( az , el, P , c , 1 , 'Time');  % delays in seconds (freq not used)
TAU = round(A*fs);                  % delays in iterations

%=========================================
% align data matrix and beamform in each look direction
%=========================================
for n=1:Nlooks
  % align all element data
  Xa = xalign(X,TAU(:,n));

  % form sidelobe canceller input matrix
  Xp = B * Xa;

  % weight all element data for TD-CBF
  %Xa = diag(W)*Xa;

  % form TD-CBF output
  Yc = sum(Xa)/M;

  % filter TD-CBF output
  %Yc = filter(b,a,Yc);

  % multichannel adaptive filter Xa matrix
  [Ya,Y(n,:),W]=mch_nlms(Xp,Yc,N,mu,0);
end;

Py = mean(Y'.*conj(Y'));

%=========================================
% tidy up and reshape the output;
%=========================================
% we know Py must be real ...
Py = real(Py);
% ... and return y with the rows containing the azimuth for a fixed
% elevation angle.
Py = reshape( Py , n_az , n_el )';

%=========================================
```

mch_af.m

```
function [Py,Y] = mch_af( X, az, el, P, dt, c, N, mu, d)
%=========================================
% MCH_AF - Multichannel Adaptive Filter with presteering (data alignment)
%
% INPUT:
%    X  - MxT array of T time samples of the M sensor outputs
%    az - beamforming azimuth angles (degrees)
%    el - beamforming elevation angles (degrees)
%    P  - 3xM array of the M sensor positions (ft) (m)
%    dt - sample increment (sec)
%    c  - speed of sound in propagation media (ft/sec) (m/sec)
%    N  - channel filter length (same for all elements)
%    mu - LMS step size
%    d  - 1xT vector of desired values
%
% OUTPUT:
%    Py - array of beamformer power output; Py(k,l) is the output
%         at azimuth angle az(l) and elevation angle el(k).
%    Y  - NlooksxT array of T time samples of the beamformer output
%
% REFERENCES:
% [1] R. North, "Broadband Conventional Beamforming Incorporating
%       Adaptive Equalization," NRaD Tech. Note 1992
%
% function [Py,Y] = mch_af( X, az, el, P, dt, c, N, mu, d)
%=========================================

%=========================================
% initialize variables
%=========================================
[M,T] = size(X);
fs = 1/dt;   % sampling frequency
n_az = length(az);
n_el = length(el);
Nlooks = n_az*n_el;
%W = ones(M,1);       % unity sensor weighting for CBF (see also chebwin)
Py = zeros(1,Nlooks);
Xa = zeros(M,T);

%=========================================
% Generate steering time delays for all directions of look
%=========================================
A = steer( az , el, P , c , 1 , 'Time');  % delays in seconds (freq not used)
TAU = round(A*fs);                  % delays in iterations

%=========================================
% Shift Delays so that the Filtering is centered giving equal +/- looks
%=========================================
TAU = TAU - N/2*ones(TAU);          % shift all delays by N/2
MinTAU = min(min(TAU));
MaxTAU = max(max(TAU));
if MinTAU < 0
  Tmax = T+MinTAU;
else
  Tmax = T;
end
if MaxTAU > 0
  Tmin = MaxTAU+1;
else
  Tmin = 1;
end
Y = zeros(Nlooks,length(Tmin:Tmax));

%=========================================
% align data matrix and beamform in each look direction
%=========================================
for n=1:Nlooks
  % align all element data
  Xa = xalign(X,TAU(:,n));

  % weight all element data
  %Xa = diag(W)*Xa;

  % multichannel adaptive filter nonzero Xa data
  [Y(n,:),E,W]=mch_nlms(Xa(:,Tmin:Tmax),d(Tmin:Tmax),N,mu,0);
end;

Py = mean(Y'.*conj(Y'));

%=========================================
% tidy up and reshape the output;
%=========================================
% we know Py must be real ...
Py = real(Py);
% ... and return y with the rows containing the azimuth for a fixed
% elevation angle.
Py = reshape( Py , n_az , n_el )';

%=========================================
```

arraypat.m

```
function W = arraypat( az , el , P , c , f0 , u , w )
%=========================================
% ARRAYPAT - computes the array pattern assuming unity shading
%       and plane-wave inputs.
%
% INPUT:
%  az - vector of azimuth angles to sweep (deg)
%  el - vector of elevation angles to sweep (deg)
%   P - 3xM array of the M sensor positions
%   c - speed of signal propagation
%  f0 - frequency to beamform
%   u - array steered direction vector [az el] (deg)
%   w - 1xM vector of element shading coefficients
%       (if w is not entered, then unity shading is assumed)
%
% OUTPUT: let M denote the number of sensors; set
%  n_az = length(az)
%  n_el = length(el)
% Then the output array pattern W is n_azxn_el where each column
% has the for
%                              T
% W(:,k) = sum {m=1} ^ {m=M} w_m*[exp( i 2 pi f0 P (A(:,k)-u_s)/c)]
%
% where u_s (3x1) is the xyz coordinates of the array steered
% direction,
%
%   u_s = | cos(u(1)) cos(u(2)) |
%         | sin(u(1)) cos(u(2)) |
%         |         sin(u(2))   |
%
% and A(:,k) (3xn_az*n_el)is the xyz matrix of assumed incident
```

Figure A.2. Continued. (Contd)

```
% narrowband complex exponential sources
%
%   A(l,k) = | cos(az(l)) cos(el(k)) |
%            | sin(az(l)) cos(el(k)) |
%            |          sin(el(k))   |
%
% written: Rich North 5-19-92
% W=arraypat(az,el,P,c,f0,u)  or
% W=arraypat(az,el,P,c,f0,u,w)
%=============================================

n_az = length(az);
n_el = length(el);
[nrow,M] = size(P);
if nargin < 7
  w = ones(1,M);
end

% use steer.m to generate time delays
A = steer( az , el , P , c , f0 , 'Time' );
u_s = steer( u(1) , u(2) , P , c , f0 , 'Time' );

T = exp(i * 2 * pi * f0 * (A-u_s*ones(1,n_az*n_el)));

% sum all shaded elements contribution to the spatial component of total
%   array output (time-domain)
W = ones(1,n_az*n_el);
[nrow,ncol] = size(w);
if nrow == 1    % w = row vector
  W = w*T/M;
elseif ncol == 1 % w = column vector
  W = w'*T/M;
else
  error('...Error: w should be a row vector 1xM');
end

W = reshape( W , n_az , n_el )';
```

bfclass.m (modified slightly from [6])

```
function y = bfclass( X, R, f0, az, el, P, dt, c)
%=============================================
% BFCLASS - beamforming using the classical estimation procedure
%
% INPUT:
%    X  - MxN array of N time samples of the M sensor outputs
%    R  - Estimate the MxM cross-spectral density matrix at f0:
%    f0 - frequency to operate beamformer (Hz)
%  n_sg - number of points per block average segment -> must divide N
%    az - beamforming azimuth angles (degrees)
%    el - beamforming elevation angles (degrees)
%    P  - 3xM array of the M sensor positions (ft)
%    dt - sample increment (sec)
%    c  - speed of sound in propagation media (ft/sec)
%
% OUTPUT:
%    y  - array of classical beamformer output; y(k,l) is the output
%         at azimuth angle az(l) and elevation angle el(k).
%
% ALGORITHM: As discussed in [1], the classical beamformer has output
%        H
%    y = a  R a
%
% for steering vector
%                  H
%    a = [exp(i 2 pi f0 p_{m} u/c)]
%
% where p_{m} is the position vector of the mth sensor, u is the look
% direction.
%
% REMARKS:
% (1) to beamform at a desired frequency f0 (Hz) requires that the sample
%     increment dt (sec) first satisfy
%          f0 < 1/2dt
% (2) y' is the conjugate transpose of the vector y
%
% REFERENCES:
% [1] Don H. Johnson,
%     "The Application of Spectral Estimation Methods to Bearing
%      Estimation Problems",
%      Proceedings of the IEEE, Volume 70, Number 9, pages 1018-1028, 1982
%
% function y = bfclass( X, R, f0, az, el, P, dt, c)
%=============================================

% Get the array A of steering vectors
n_az = length(az);
n_el = length(el);
A = steer( az , el, P , c , f0 );

% Beamform using the correlation matrix R at the specified azimuth angles
% and the assumed input frequency f0.
y = zeros(1,n_az*n_el);
for k=1:(n_az*n_el)
  y(k) = A(:,k)' *R * A(:,k);
end

% tidy up and reshape the output;
% we know y must be real ..
y = real(y);

% ... and return y with the rows containing the azimuth for a fixed
% elevation angle.
y = reshape( y , n_az , n_el )';
```

mvdr.m (modified slightly from [6])

```
function [y,w] = mvdr( X, R, f0, az, el, P, dt, c)
%=============================================
% MVDR - Minimum-Variance Distortionless Response beamforming
%
% INPUT:
%    X  - MxN array of N time samples of the M sensor outputs
%    R  - Estimate of the MxM cross-spectral density matrix at f0:
%    f0 - frequency to operate beamformer (Hz)
%  n_sg - number of points in each block averaging segment -> must divide N
%    az - beamforming azimuth angles (degrees)
%    el - beamforming elevation angles (degrees)
%    P  - 3xM array of the M sensor positions (ft)
%    dt - sample increment (sec)
%    c  - speed of sound in propagation media (ft/sec)
%
% OUTPUT:
%    y  - MVDR beamformer output; y(k,l) is the output at azimuth angle
%         az(l) and elevation angle el(k).
%    w  - (n_az*n_el)xM array of row vectors containing the
%         element weights for each azimuth and elevation angle
%         (only outputed if nargout = 2)
%
% ALGORITHM: As analyzed in [2], the MVDR beamformer has output
%          H      H +
%    y = w  R w  = 1/(a  R  a)   (1)
%          +    H +
%    w = (R  a)/(a  R  a)        (2)
%
% for steering vector
%                  H
%    a = [exp(i 2 pi f p_{m} u/c )]    (m=1,2, ..., M)
%
%    +
%   R
% is the Moore-Penrose inverse R  PROVIDED the steering vector is
% orthogonal to the null space of R.  Otherwise, if the steering
% vector has a non-zero component in the null space of R, then
% the MVDR output is y = 0.
%
% WARNING:
% This function MVDR assumes that the estimated CSD matrix R is full
% rank and therefore will calculate y as in equation (1).
%
% REMARKS:
% (1) to beamform at a desired frequency f0 (Hz) requires that the sample
%     increment dt (sec) first satisfy
%          f0 < 1/2dt
% (2) y' is the conjugate transpose of the vector y
% (3) to obtain a full rank CSD estimate, it is necessary to use at least
%     M block averages in function "sfecsd".
%
% REFERENCES:
% [1] Don H. Johnson,
%     "The Application of Spectral Estimation Methods to Bearing
%      Estimation Problems",
%      Proceedings of the IEEE, Volume 70, Number 9, pages 1018-1028, 1982
% [2] Jeffrey Speiser
%     "MVDR Fundamentals and Computation"
%     preprint, Code 635, NOSC, San Diego, CA, 92152-5000
%
% y = mvdr( X, R, f0, az, el, P, dt, c) or
% [y,w] = mvdr( X, R, f0, az, el, P, dt, c)
%=============================================

% Calculate the inverse of the CSD matrix:  it is assumed that R is full
% rank and therefore the Moore-Penrose inverse and inv(R) are the same.
R_pinv = pinv(R);

% Get the array of steering vectors.
% On output, the matrix A will be M x ( n_az n_el ).
n_az = length(az);
n_el = length(el);
A = steer( az , el , P , c , f0 );

% Beamform using the correlation matrix R at the specified azimuth angles
% and the assumed input frequency f0
y = zeros(1, (n_az*n_el) );
for k=1:(n_az*n_el)
  y(k) = A(:,k)' *R_pinv * A(:,k);
end
% we know y must be real ...
y = 1 ./real(y);

if nargout == 2
  % generate the weights for each azimuth and elevation steering
  [nrow,M] = size(P);
  w = zeros((n_az*n_el),M);
  w = (R_pinv * A).*.' (y'*ones(1,M));
end

% tidy up and reshape y:
% ... the k-th row of y contains the beamformer output at elevation
% el(k) and all the azimuth angles.
y = reshape( y , n_az , n_el )';
```

Figure A.2. Continued.

Appendix B
TIME-DOMAIN CBF OF A LINEAR ARRAY WITH UNIFORMLY SPACED ELEMENTS

As an example of time-domain CBF (and a more detailed CBF review), consider a linear array with uniform element spacing, d, and a single source such that

$$\vec{x}_m = (m\ d)\ \hat{z} \qquad m = 0,1,2,\ldots,M-1$$
$$\vec{s}_1 = (\cos\phi_1\ \cos\theta_1)\ \hat{x} + (\cos\phi_1\ \sin\theta_1)\ \hat{y} + (\sin\phi_1)\ \hat{z}$$
$$\vec{u} = (\cos\phi\ \cos\theta)\ \hat{x} + (\cos\phi\ \sin\theta)\ \hat{y} + (\sin\phi)\ \hat{z}$$

It is easy to observe that the delay required for coherent summing across the array requires $\vec{u} = \vec{s}_1$, or a delay equal to

$$\tau_m(\vec{u}) = \frac{\vec{x}_m \cdot \vec{u}}{c} = \frac{m\ d\ \sin\phi_1}{c} \quad \text{(sec)} \tag{B.1}$$

$$= round\left[\frac{m\ d\ \sin\phi_1}{c}\ f_{sample}\right] \quad \text{(iterations)} \tag{B.2}$$

Figure B.1 shows the uniformly spaced, linear, vertical array (ULVA) and a single source. When the array is not steered directly at the source, Eq. NO TAG becomes

$$y(n,\vec{u}) = \sum_{m=0}^{M-1} w_m\ n_m\left(n - round\left[\frac{md\sin\phi}{c}\ f_{sample}\right]\right) + \sum_{m=0}^{M-1} w_m\ s_1\left(n + round\left[\frac{md(\sin\phi_1 - \sin\phi)}{c}\ f_{sample}\right]\right) \tag{B.3}$$

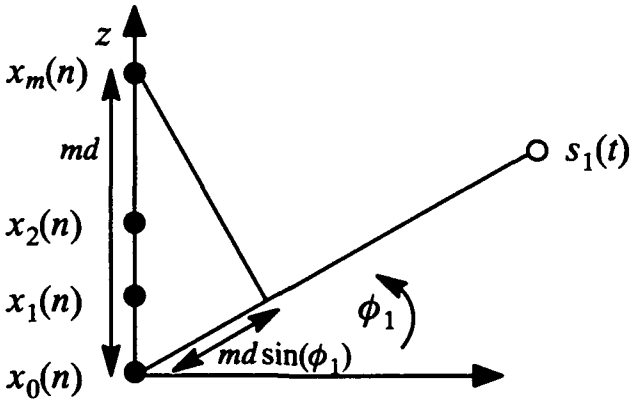


Figure B.1. Single source ULVA.

To proceed analytically, we must make some assumptions about the source model. Assume a sampled, narrowband complex exponential source, $s_1 n) = A_1\ e^{j\,2\pi\,n f_1'}$ where $f_1' = f_1\ /\ f_{sample}$ is the normalized source frequency. Then Eq. (B.3) can be rewritten as

$$y(n,\vec{u}) = \sum_{m=0}^{M-1} w_m \, n_m\left(n - round\left[\frac{m\,d}{c}\,\sin\phi\, f_{sample}\right]\right) + \left[A_1\, e^{j2\pi n f_1'}\right]\left[\sum_{m=0}^{M-1} w_m\, e^{j2\pi f_1' \, round\left[\frac{m\,d}{c}(\sin\phi_1 - \sin\phi) f_{sample}\right]}\right] \tag{B.4}$$

And the power spectrum of Eq. (B.4) for stationary white noise, $n_m(n)$, is [6]

$$P_y(n,\vec{u}) = \mathcal{F}\{R_y(\tau,\vec{u})\} = \mathcal{F}\left\{E\left[y(n+\tau,\vec{u})\; y(n,\vec{u})^H\right]\right\} \tag{B.5}$$

$$P_y(f,\vec{u}) = \sigma_n^2(f) \sum_{m=0}^{M-1} w_m + P_s(f)\left|\sum_{m=0}^{M-1} w_m\, e^{j2\pi f_1' \, round\left[\frac{m\,d}{c}(\sin\phi_1 - \sin\phi) f_{sample}\right]}\right|^2$$

Note that the second term of Eq. (B.4) is composed of a time-varying component (left term) and a spatially-varying, time-invariant component (right term). The spatially-varying term is a function of wavelength, $\lambda = \frac{f_1}{c}$, element spacing, and the direction of the source relative to the array. The spatially-varying component is usually written in terms of the wavenumber, $K = \frac{2\pi}{\lambda}$. Thus, we find the continuous-time spatial component of the array is equal to

$$W(\phi) = \sum_{m=0}^{M-1} w_m\, e^{-jK_1 m d(\sin\phi - \sin\phi_1)} \tag{B.6}$$

and the discrete-time spatial component of the array equal to

$$W(\phi) = \sum_{m=0}^{M-1} w_m\, e^{-j2\pi f_1' \, round\left[\frac{m\,d}{c}(\sin\phi - \sin\phi_1) f_{sample}\right]} \tag{B.7}$$

Eq. (B.6) can be viewed as the *spatial transfer function* of the array, since we can let $z = e^{jK_1 d(\sin\phi - \sin\phi_1)}$ and rewrite Eq. (B.6) as a (right-sided i.e., appropriate for causal sequences) Z transform, $W(\phi) = \sum_{m=0}^{M-1} w_m\, z^{-m}$. Equation (B.7) can be thought of as the sampled *spatial transfer function.* Thus, the number of zeros defining the spatial component of the array is equal to the number of elements, M. It is apparent that the shading coefficients behave similarly to standard FFT weighting [2] for the uniformly spaced linear array. However, because of the nonlinear relationship between $W(\phi)$ and ϕ_1, the 3-dB beamwidth of the main lobe is minimum at $\phi = 0$ degrees (broadside) and maximum at $\phi = \pm 90$ degrees (end-fire).

The spatial filtering characteristics (called the *array pattern* for the general array configuration case) of an 8−element ULVA with half-wavelength spacing and no shading are demonstrated in Figures B.2 and B.3 for the array steered in the broadside direction and the near end-fire direction. It is easily seen from the figures that the main lobe's spatial width increases as the steering angle approaches +/− 90 degrees. It was found in [1] that the main lobe beamwidth is approximately equal to

$$BW \approx 49.6\,\frac{\lambda}{Nd}\; sec(\phi) \quad \text{(degrees)} \tag{B.8}$$

for steering angle ϕ. The effects of sampling the continuous data are seen in figures B.4 and B.5 for f_{sample} = 250 Hz and f_{sample} = 1500 Hz, respectively. The delineation of the array pattern is seen to improve as the sampling frequency increases. Finally, figures B.6 and B.7 show the ULVA pattern for detecting frequencies for which the array is not exactly tuned, i.e., $\frac{d}{\lambda} \neq 0.5$. Referring back to the Z-transform description of the array pattern, it is seen that the Z-plane unit circle is traversed exactly $2\frac{d}{\lambda}$ times. Thus, for $\frac{d}{\lambda} < 0.5$, the source frequency is less than the designed frequency and the array pattern only traverses part of the Z-plane unit circle. For $\frac{d}{\lambda} > 0.5$, the source frequency is larger than the designed frequency and the array pattern traverses the Z-plane unit circle more than once, resulting in a type of aliasing. This generates spurious lobes called "grating lobes" which create an ambiguity as to the true direction of arrival. Only for $\frac{d}{\lambda} = 0.5$ is the Z-plane unit circle traversed exactly once.

The time-domain CBF can be demonstrated for two narrowband inputs separated by only 5 degrees. A 40-element ULVA is used with low SNRs for both inputs. Figure B.8 plots the beampattern averaged over 100 iterations for $\phi_1 = 0$ degrees, $f_1 = 25$ Hz, $SNR_1 = 0$ dB and $\phi_2 = -5$ degrees, $f_2 = 45$ Hz, $SNR_2 = -5$ dB. From figure B.8 one can see the excellent results which can be obtained with time-domain CBF. Appendix A lists the MATLAB code used to generate the ULVA data.

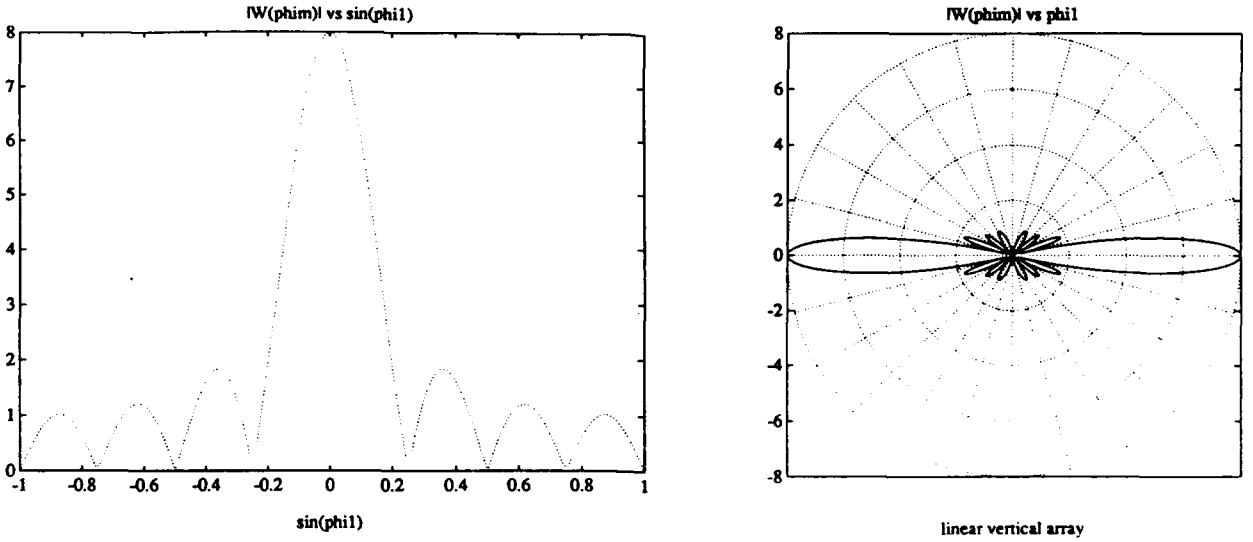


Figure B.2. ULVA pattern for $\frac{d}{\lambda} = 0.5$, $\phi_m = 0$ degrees, $N = 8$, $f_{sample} = \infty$.

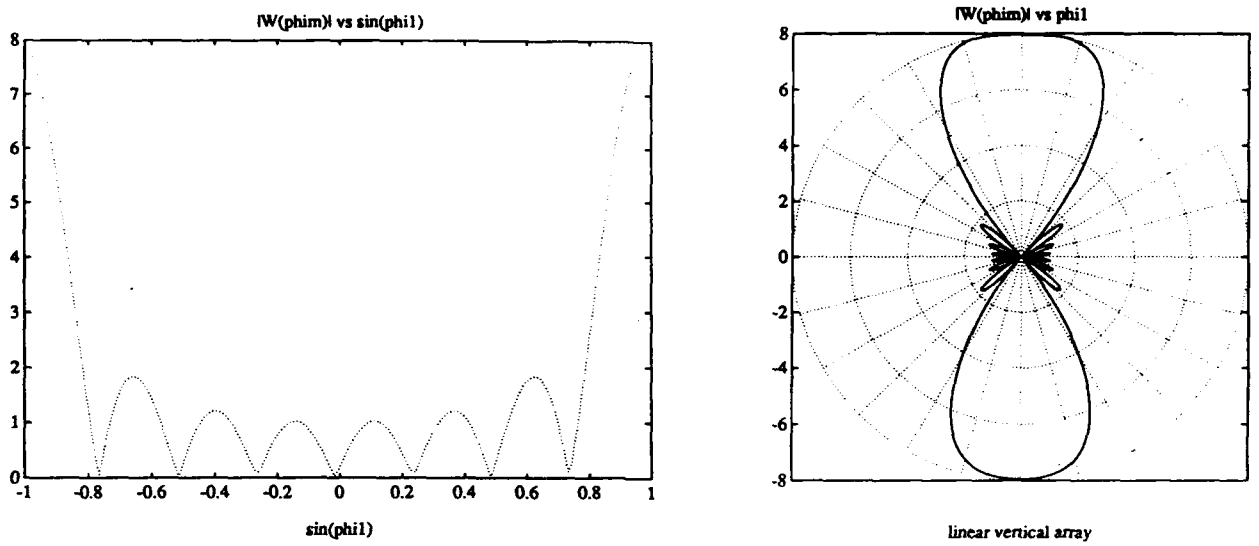


Figure B.3. ULVA pattern for $\frac{d}{\lambda} = 0.5$, $\phi = 80$ degrees, $M = 8$, $f_{sample} = \infty$.

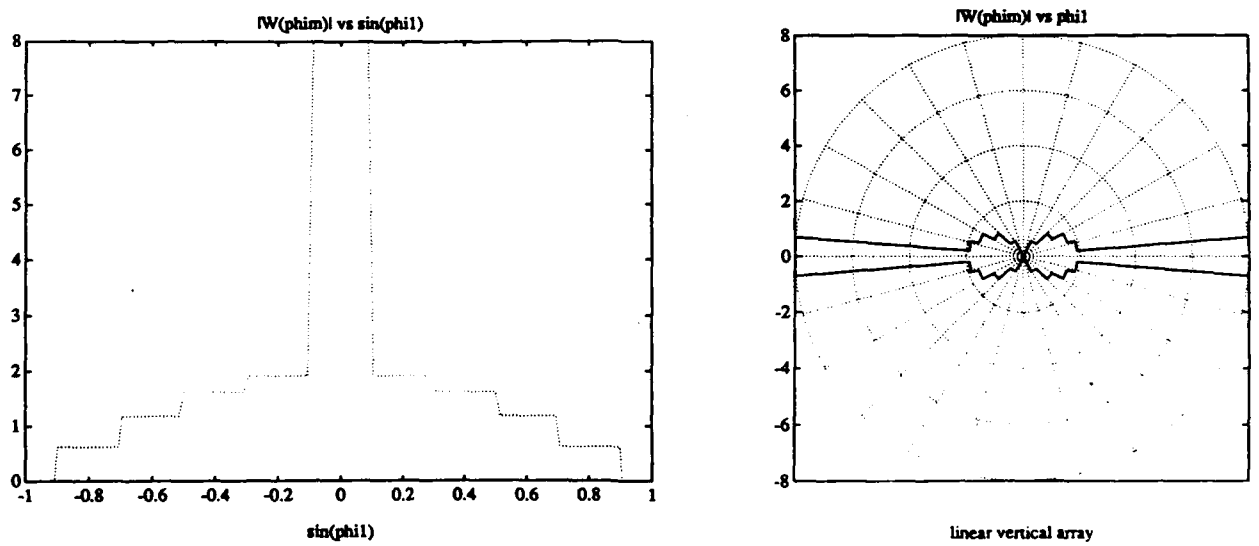


Figure B.4. ULVA pattern for $\frac{d}{\lambda} = 0.5$, $\phi = 0$ degrees, $M = 8$, $f_{sample} = 250$ Hz.

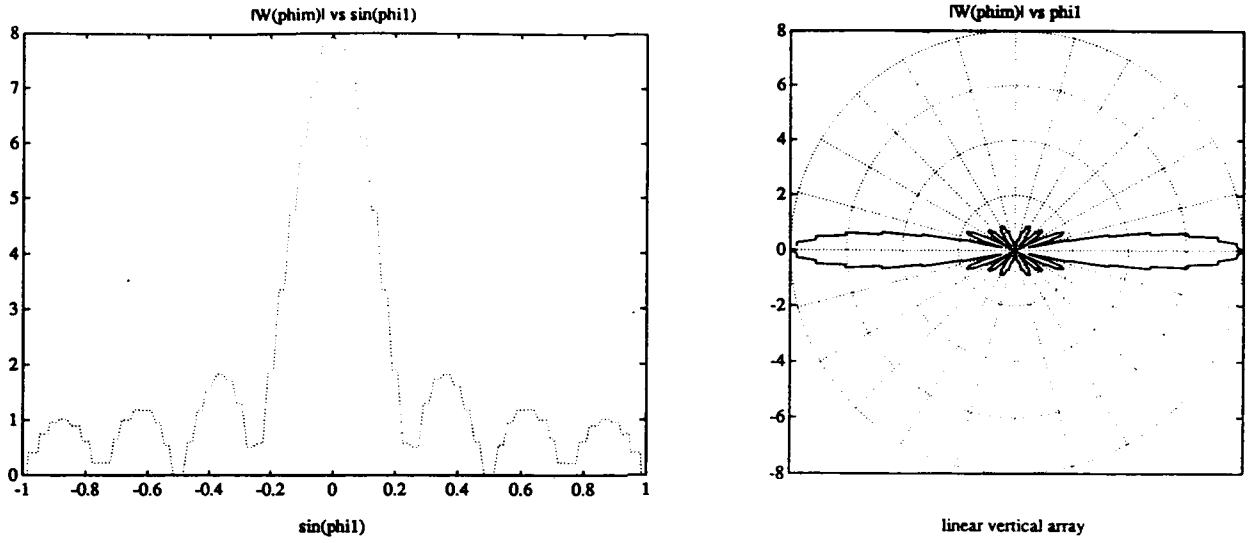


Figure B.5. ULVA pattern for $\frac{d}{\lambda} = 0.5$, $\phi = 0$ degrees, $M = 8$, $f_{sample} = 1500$ Hz.

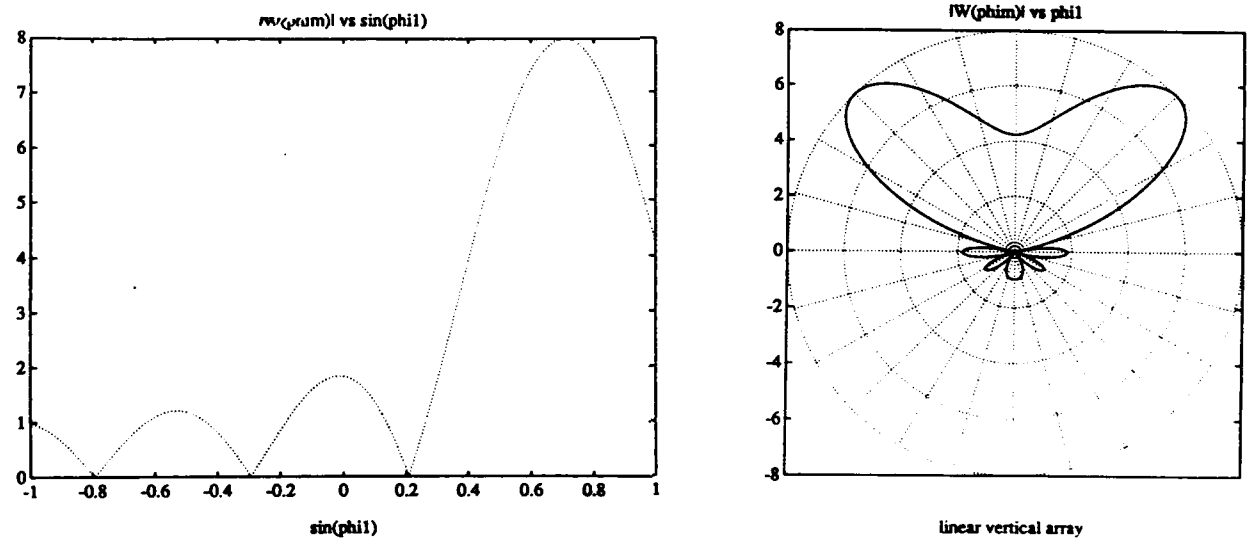


Figure B.6. ULVA pattern for $\frac{d}{\lambda} = 0.25$, $\phi = 45$ degrees, $M = 8$, $f_{sample} = \infty$.

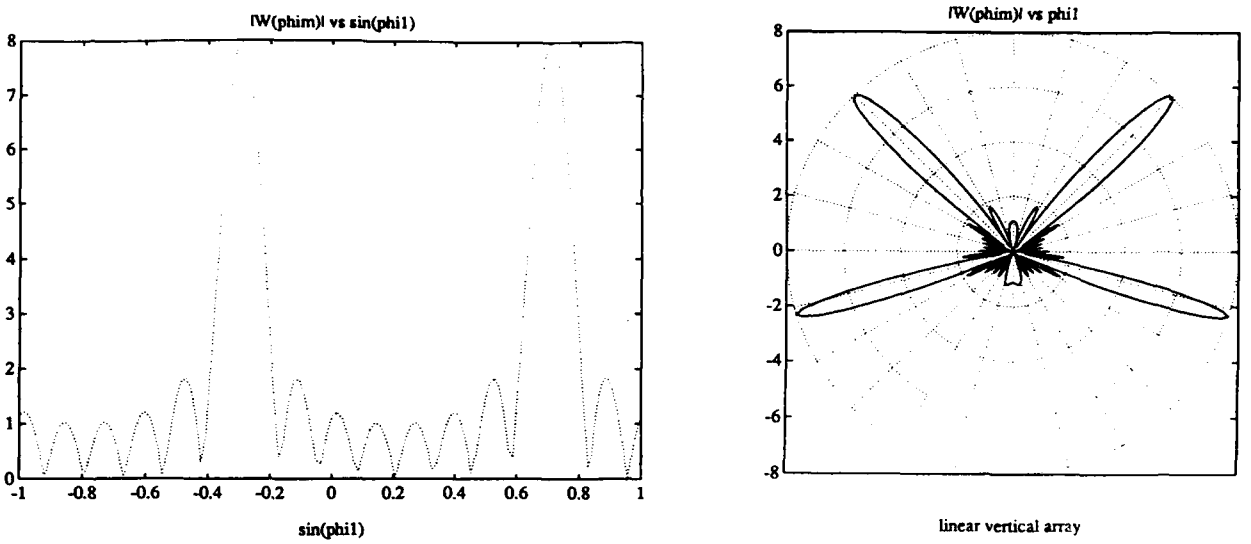


Figure B.7. ULVA pattern for $\frac{d}{\lambda} = 1.0$, $\phi = 45$ degrees, $M = 8$, $f_{sample} = \infty$.

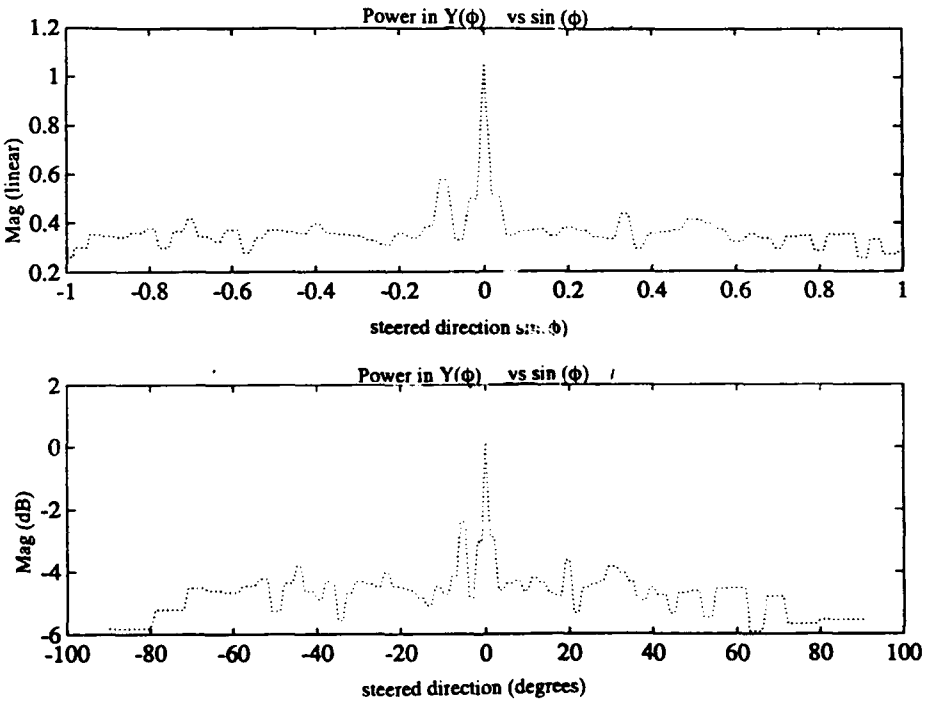


Figure B.8. ULVA TD-CBF averaged beampattern $M = 40$, $d = 30.0$ m, $\phi_1 = 0$ degrees, $f_1 = 25$ Hz, $SNR_1 = 0$ dB, $\phi_2 = -5$ degrees, $f_2 = 45$ Hz, $SNR_2 = -5$ dB, $f_{sample} = 750$ Hz, $I = 2$.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1992	Final: Jan 92—Sep 92

4. TITLE AND SUBTITLE
BROADBAND CONVENTIONAL BEAMFORMING INCORPORATING ADAPTIVE EQUALIZATION

5. FUNDING NUMBERS
0602314N
DN308291

6. AUTHOR(S)
Richard C. North

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

8. PERFORMING ORGANIZATION REPORT NUMBER
TR 1575

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Technology
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report discusses methods which attempt to coherently recombine the spatially and temporally different multipaths with the direct path to enhance array gain. These methods are based on utilizing a broadband conventional beamformer incorporating adaptive equalization to recombine multipath information. Preliminary results are compared with two well-known narrowband, purely spatial processing techniques: the Bartlett beamformer and the minimum variance distortionless response beamformer. While preliminary results are promising, several important research issues are outlined which remain to be addressed.

14. SUBJECT TERMS
adaptive beamforming
broadband beamforming
multichannel equalization
adaptive equalization
temporal/spatial processing

15. NUMBER OF PAGES
44

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE *(Include Area Code)*	21c. OFFICE SYMBOL
Richard C. North	(619) 553–4323	Code 1431

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 572	T. A. Albert	(1)
Code 782	R. Dukelow	(1)
Code 80	K. D. Regan	(1)
Code 804	J. R. Zeidler	(1)
Code 82	R. J. Kochanski	(1)
Code 824	J. B. Rhode	(1)
Code 824	R. C. North	(15)
Code 824	C. Fuzak	(1)
Code 844	R. A. Axford	(1)
Code 961	Archive/Stock	(6)
Code 964B	Library	(2)

Defense Technical Information Center
Alexandria, VA 22304–6145 (4)

Center for Naval Analyses
Alexandria, VA 22302–0268

NCCOSC Washington Liaison Office
Washington, DC 20363–5100

Navy Acquisition, Research and Development
Information Center (NARDIC)
Washington, DC 20360–5000

GIDEP Operations Center
Corona, CA 91718–8000

NCCOSC Division Detachment
Warminster, PA 18974–5000