University of Central Florida
STARS

Retrospective Theses and Dissertations

1986

Comparison of Apodization Geometries for Bidirectional Surface Acoustic Wave Transducers Based on Efficient Track Modelling

Armando J. Vigil
University of Central Florida

Part of the Engineering Commons
Find similar works at: https://stars.library.ucf.edu/rtd
University of Central Florida Libraries http://library.ucf.edu

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Vigil, Armando J., "Comparison of Apodization Geometries for Bidirectional Surface Acoustic Wave Transducers Based on Efficient Track Modelling" (1986). *Retrospective Theses and Dissertations*. 4969.
https://stars.library.ucf.edu/rtd/4969

COMPARISON OF APODIZATION GEOMETRIES
FOR
BIDIRECTIONAL SURFACE ACOUSTIC WAVE TRANSDUCERS
BASED ON
EFFICIENT TRACK MODELLING

BY

ARMANDO JUAN VIGIL
B.S.E., University of Central Florida, 1984

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering in the
Graduate Studies Program of the College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1986

ABSTRACT

Apodization, or spatial overlap weighting, is a common tap weighting technique for Surface Acoustic Wave filter transducers. The two most common apodization strategies are beam centered and slant centered overlap weighting. Two lesser known apodization geometries are beam spread overlap weighting and generalized distance weighting.

A wide variety of SAW transducer weighting techniques are described. Analysis is focussed on the four apodization geometries listed above. An efficient track based modelling system is developed in order to accurately solve for the acoustic radiation profiles, apodization loss and frequency dependent acoustic conductance. This modelling system becomes the foundation for a set of apodized transducer simulation routines named "STRUCTURECAD." These routines may be used in conjunction with SAWCAD to generate complete filter response simulations based on the impulse response model and the electrical network responses.

Computer simulations are carried out on filters whose transducers are weighted by the four apodization techniques described above. Transducer geometries are compared on the basis of these simulations for both an Eigen window filter and a low shape factor filter with time sidelobes. It is shown that the two less popular apodization geometries exhibit more favorable apodization loss characteristics. These transducers function more like amplitude weighted transducers than the beam and slant centered overlap transducers. Some apodized transducer filter responses are predicted to have less passband ripple than their amplitude weighted counterparts.

ACKNOWLEDGEMENTS

This material is based upon work supported under a National Science Foundation Graduate Fellowship.

I would like to thank my advisor for his guidance and his patience, but most of all for his friendship. Each member of the U.C.F. Solid State Devices and Systems Group should be acknowledged for their support in terms of resources, technical assistance, experimental device data and encouragement. Finally, I would like to thank my family and friends in Orlando and Miami for their moral support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Bidirectional SAW Filters

A Surface Acoustic Wave (SAW) filter is an implementation of a transversal filter on a polished piezoelectric substrate [1]. SAW filters are bandpass in nature, operating in the 10 MHz. to 1 GHz. frequency range. SAW filters are fabricated using standard microelectronic techniques for metallization and photolithography. Applications for filtering are found in Radio Frequency communication and RADAR systems.

SAW filters employ at least two electroacoustic transducers, one at the filter input and one at the filter output. Each transducer consists of a series of overlapping electrodes. Due to the piezoelectric effect, voltages are sampled acoustically at the filter input transducer. In a reciprocal manner, the surface wave generated is sampled electrically by the output transducer. A typical SAW device is illustrated in Figure 1.

Each transducer's electroacoustic frequency response is obtained, to a first order, by the Fourier transform of the transducer's time sampled impulse response. The complete transfer function of a two transducer SAW filter

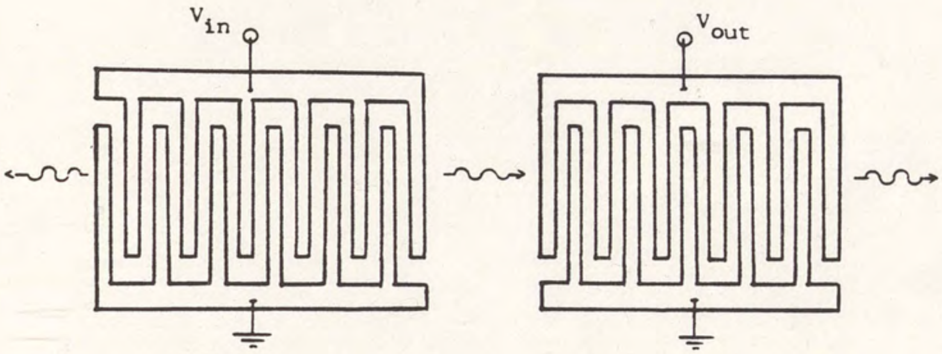


Figure 1. A Typical SAW Device.

is given by the product of the frequency responses of each transducer [2]. Although the technology exists to build unidirectional SAW transducers [3,4], only bidirectional transducers are to be analyzed.

Weighting Techniques

There are many strategies used to implement electroacoustic sampling in interdigital transducers. These strategies, or weighting techniques, may be divided into three general groups, which are uniform weighting, amplitude weighting and spatial weighting. Various weighting techniques from all three groups are described in Chapter II.

Four weighting techniques will be analyzed in detail. These techniques are Beam Centered Overlap Apodization, Slant Centered Apodization, Beam Spread Apodization, and Generalized Distance Weighting Apodization. All four are periodic spatial weighting techniques.

Analysis Methods

The analysis methods to be applied to the four transducer geometries of interest are described in Chapter III. Analysis is based on the impulse response model [1,5,6]. An efficient track analysis strategy is used to calculate conductance and apodization loss. A device simulation is used to compare the four transducers of interest. Chapter IV describes the results of this simulation applied to two filters. The four transducer geometries are compared on the basis of these results.

CHAPTER II

WEIGHTING TECHNIQUES

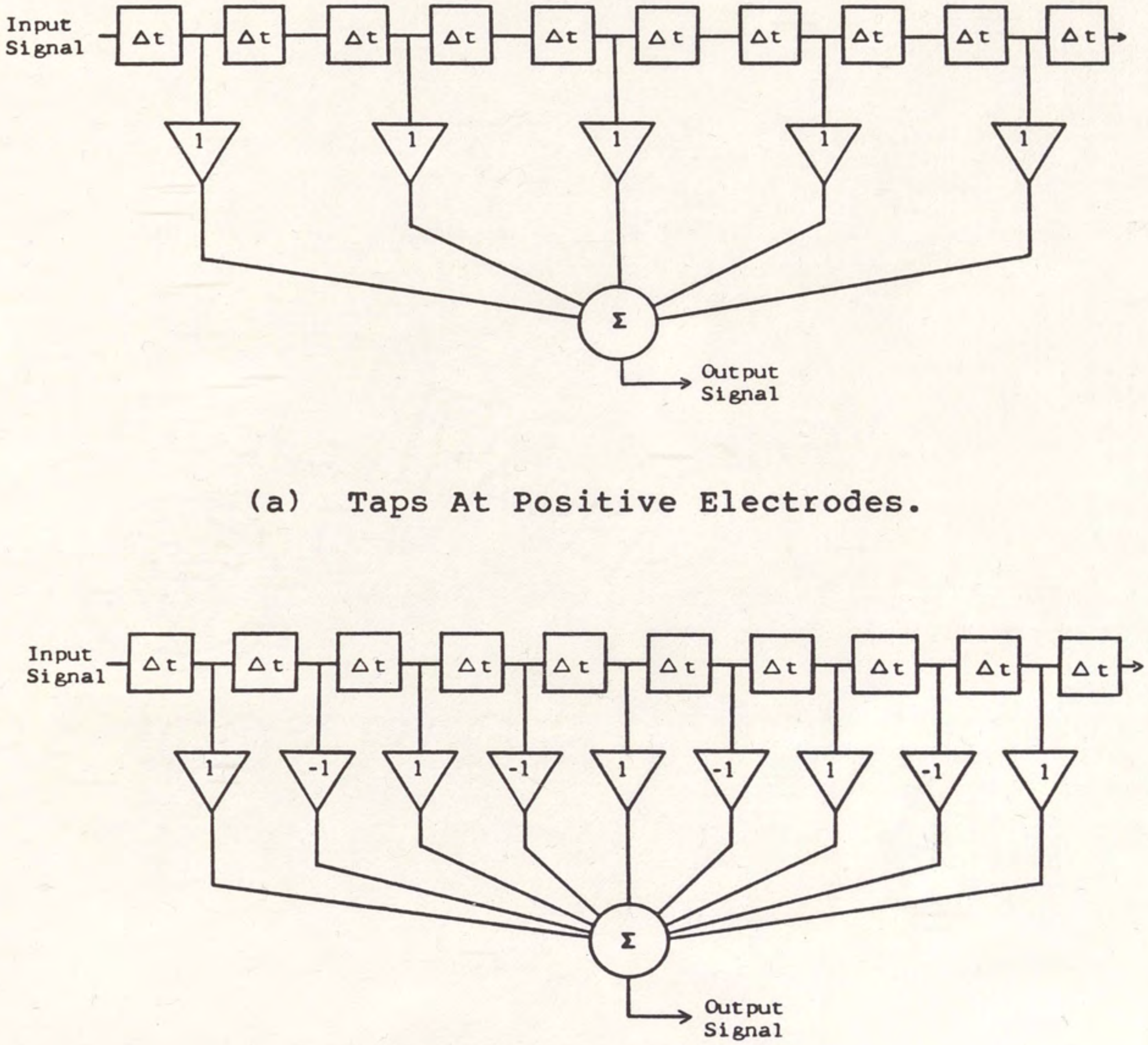
"Weighting Techniques" refers to the strategy used to implement interdigital sampling in a SAW transducer. A variety of weighting techniques are available to the SAW designer. Weighting techniques may be divided into three general classes: uniform weighting, amplitude weighting and spatial weighting.

Uniform Weighting

Uniform weighting for periodic transducers results in every tap, or electrode overlap, having equal length. The typical SAW device illustrated in Figure 1 consists of two uniformly weighted transducers. Figure 2 illustrates the tapped delay line models associated with a uniform overlap transducer. The acoustic time delay between fingers is given as Δt. Since weighting is uniform, all taps have equal magnitude. In Figure 2(a), taps are modelled as occurring at positive fingers. Figure 2(b) represents finger overlaps as taps.

Amplitude Weighting

Amplitude weighting refers to tap implementation via wave amplitude modulation. The amplitude of the acoustic



(a) Taps At Positive Electrodes.

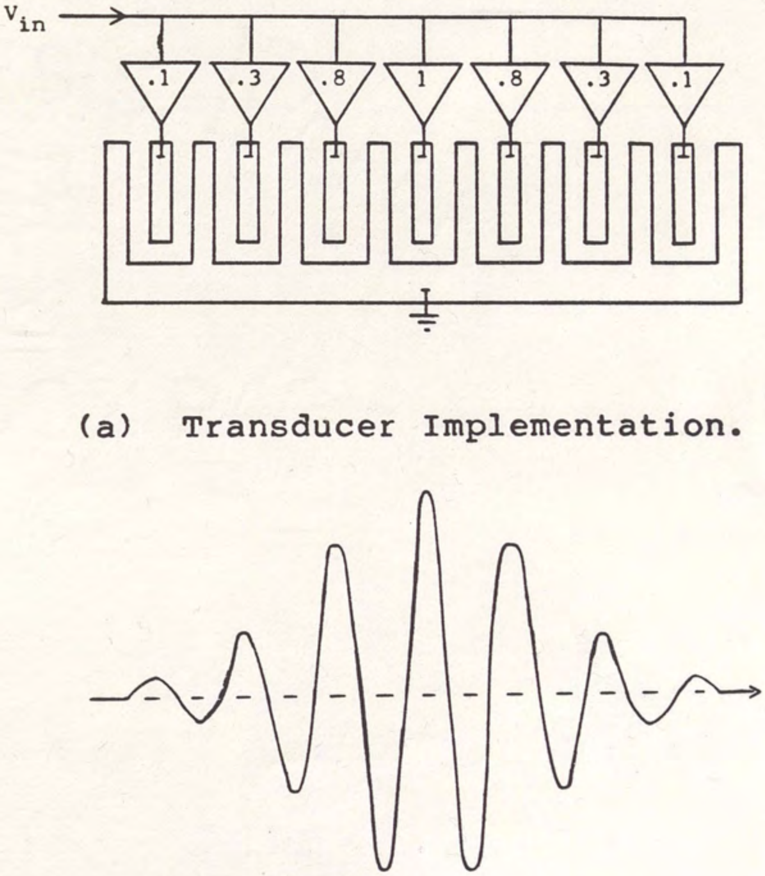
(b) Taps At Electrode Overlaps.

Figure 2. Tapped Delay Line Models for Uniformly Weighted Transducer.

signal leaving a finger is proportional to that finger's corresponding tap weight. Figure 3 depicts a schematic representation of a possible amplitude weighted transducer.

Amplitude weighted devices are advantageous over spatially weighted devices by having uniform beam profiles for all frequencies, therefore no apodization loss. Most amplitude weighted devices, however, are more difficult to fabricate.

Amplitude weighting has been implemented both actively and passively. Active tap weight control has resulted in prototype programmable and adaptive filters [7,8].



(a) Transducer Implementation.

(b) Resulting Output Waveform.

Figure 3. Schematic Representation of an Amplitude Weighted SAW Transducer.

Passive amplitude weighting is accomplished in capacitive tap weighted transducers through thin film capacitor attenuating networks [9]. Withdrawal weighting is still another form of amplitude weighting [10]. Withdrawal weighting approximates a desired time impulse response with groups of strategically placed uniform weighted finger pairs. Figure 4 depicts a withdrawal weighted transducer.

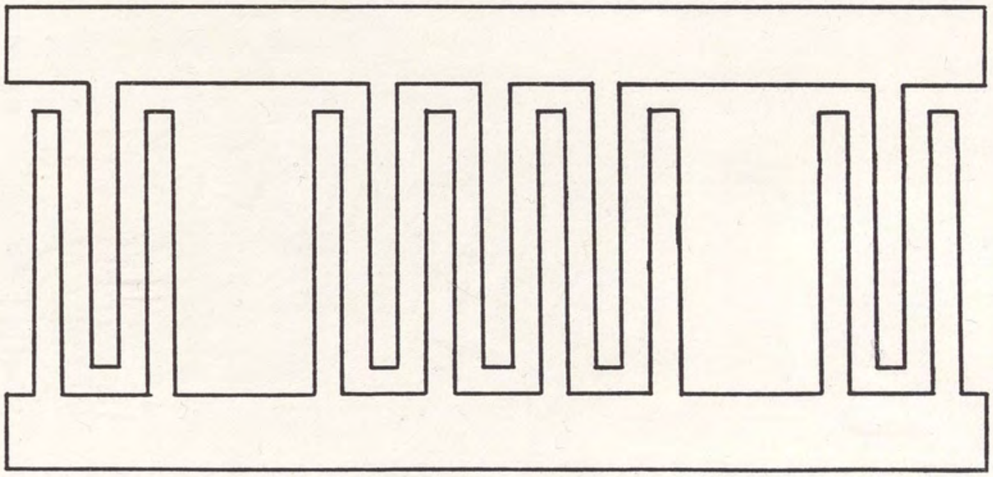


Figure 4. Withdrawal Weighted Transducer.

Spatial Weighting

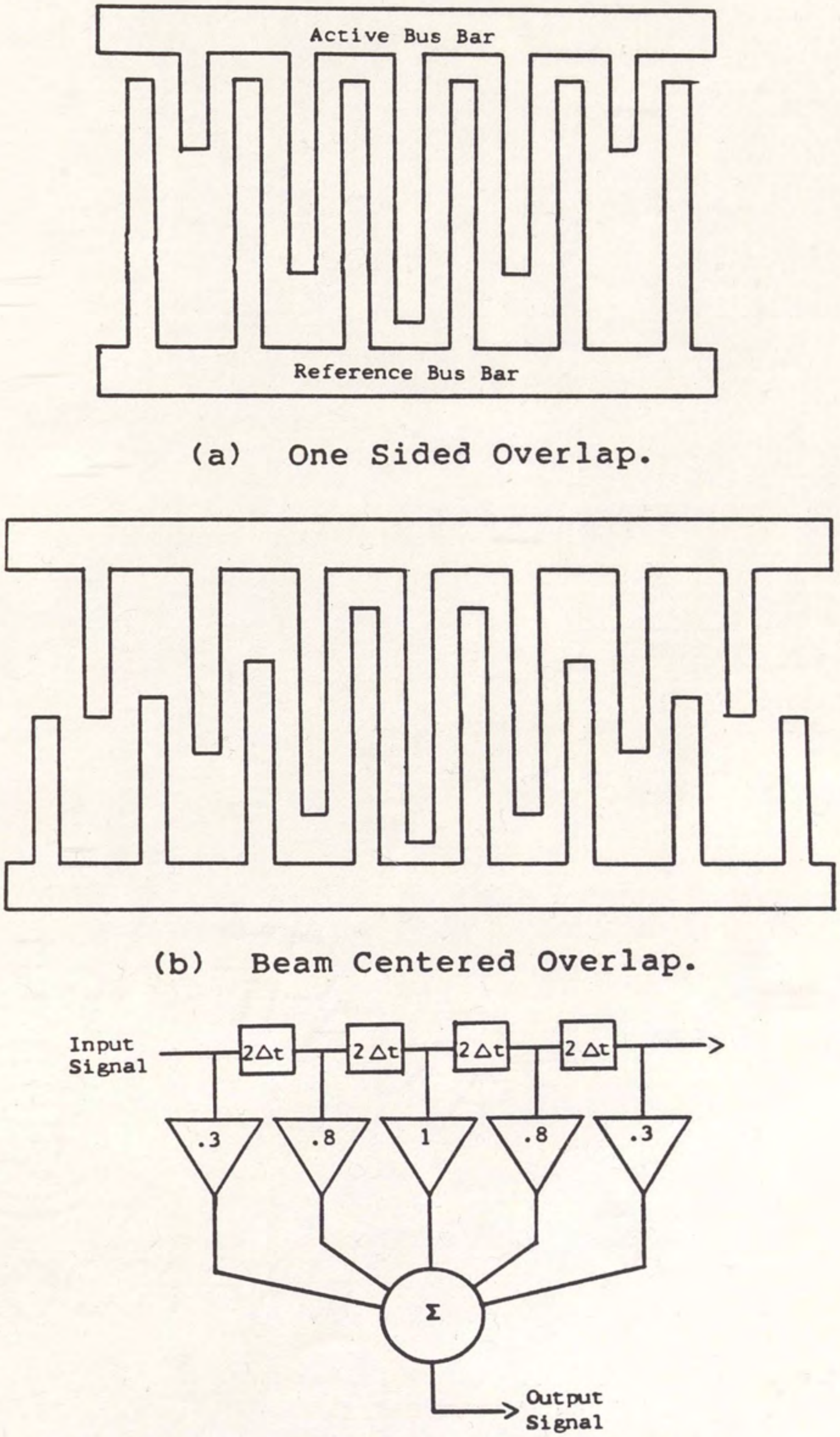
Spatial weighting differs fundamentally from amplitude weighting by modulating the position rather than the amplitude of the acoustic signal. The most common approach to spatial weighting is apodization.

Apodized Transducers

Apodized transducers are characterized by fingers whose lengths are a function of their corresponding tap weights. Figure 5 illustrates two apodized transducers. Apodized transducers are generally periodic, and are much easier to fabricate than amplitude weighted transducers.

Apodized transducers distribute acoustic energy unevenly across the acoustic beamwidth. Because of this phenomenon, there is typically an apodization loss associated with every apodized transducer. It will be shown in Chapter III that apodized transducers may be divided into tracks and thought of as a parallel combination of uniform weighted transducers, each having a different frequency and/or phase response [11].

Simple Apodization. A "simple" or "one-sided" apodized transducer is illustrated in Figure 5(a). Active electrodes, connected by the active bus bar, are proportional in length to their corresponding positive tap weights. Reference electrodes are uniform in length and shorted together by the reference bus bar. It can be seen that closer to the active bus bar side, there are more active finger to reference finger overlap regions. This characteristic lends itself to the name "one sided overlap" apodization.



(a) One Sided Overlap.

(b) Beam Centered Overlap.

(c) Tapped Delay Line Model.

Figure 5. Apodized SAW Transducers.

Beam Centered Overlap. An example of a beam centered overlap transducer is depicted in Figure 5(b). Regions of overlap are symmetrical about the center of the acoustic beamwidth and proportional to their corresponding tap weights. This approach is the most common SAW transducer implementation. The greatest number of overlaps occur at the center of the acoustic aperture. The beam centered overlap SAW transducer structure will be analyzed carefully in Chapter IV.

Slant Centered Overlap. Whereas overlap regions in the beam centered transducer are symmetrical about the center of the acoustic beamwidth, slant centered transducer overlaps are symmetrical about a line which crosses the transducer diagonally. A slant centered overlap transducer is illustrated in Figure 6.

As with the beam centered overlap transducer, overlap regions for the slant centered transducers are proportional to their corresponding sampled impulse response tap weights. However, it can be seen that the concentration of overlap regions towards the center of the beamwidth or towards either bus bar is much less extreme than for either the one sided or the beam centered case. The slant centered overlap geometry will be analyzed carefully in Chapter IV.

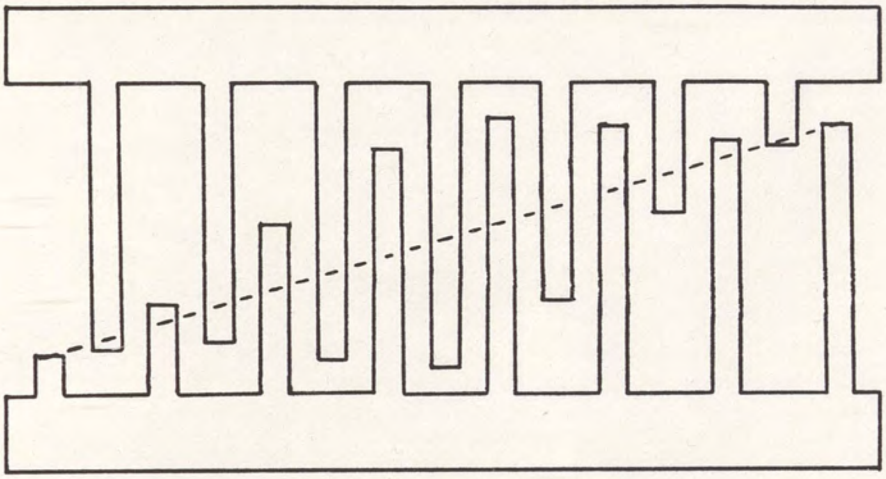


Figure 6. Slant Centered Overlap Apodized Transducer.

Beam Spread Overlap Apodization. An example of a beam spread overlap apodized transducer is illustrated in Figure 7. The beam spread geometry has two active bus bars. Like simple apodization, active electrode lengths are determined by the positive tap weights of the desired weighting function. Consecutive active electrodes originate from opposite bus bars except at time response null points, where two positive electrodes may not be separated by a reference electrode. As in the simple apodization structure, reference electrodes are uniform in length, each spanning the acoustic beamwidth. However, in lieu of a reference bus bar, reference electrodes are connected in series, forming a meandering reference line which weaves itself through the transducer.

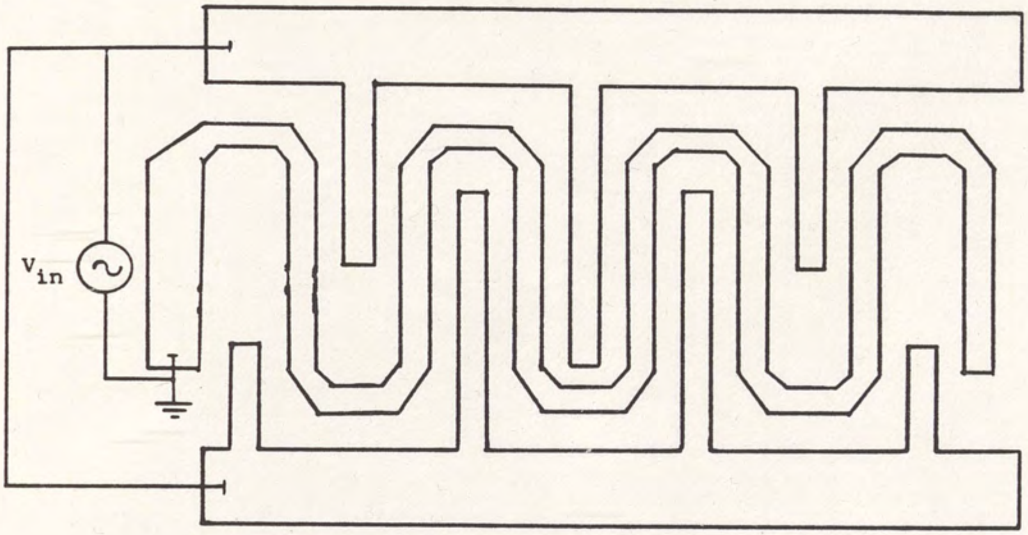
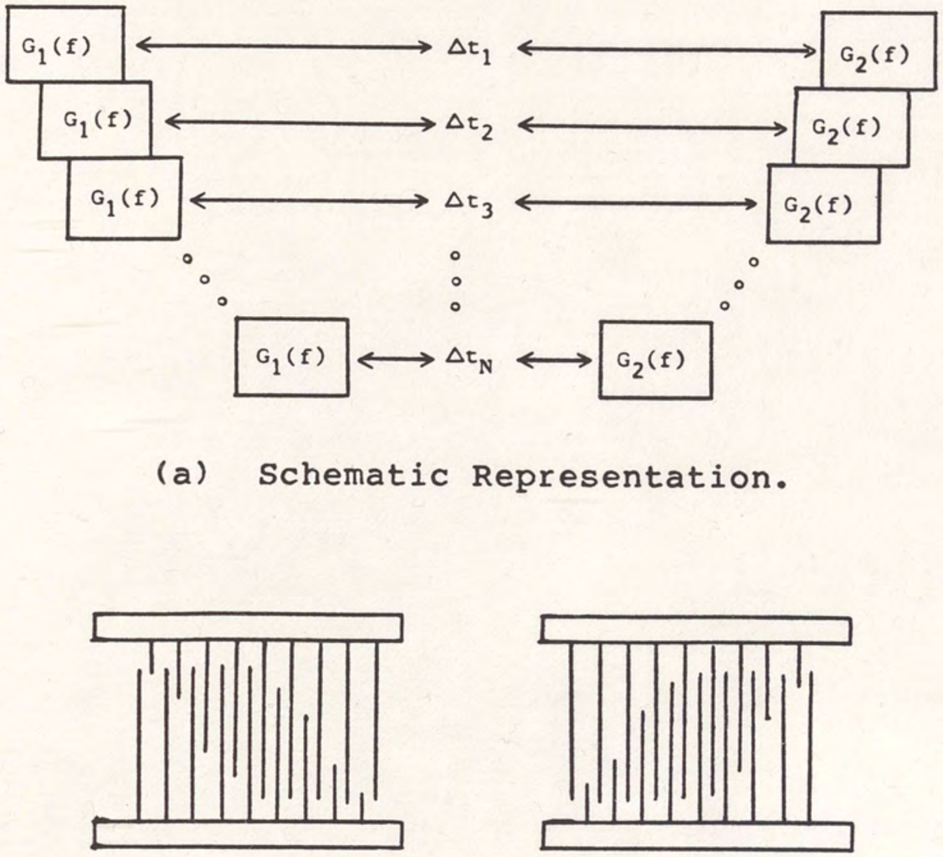


Figure 7. Beam Spread Overlap Apodized Transducer.

As the name "beam spread overlap" suggests, areas of overlap region concentration are spread across the width of the transducer. This structure will be analyzed in greater detail in Chapter IV.

Distance Weighting. Distance weighting was proposed by Yamanouchi in 1980 as a low loss SAW filter weighting scheme [12]. An example of a distance weighted filter is illustrated in Figure 8.

Cross sections of distance weighted transducers parallel to the propagation direction are uniform. Distance weighted filter responses are synthesized by varying the filter group delay across the acoustic beamwidth. Frequency responses of tracks, or cross



(a) Schematic Representation.

(b) Filter Configuration.

Figure 8. Distance Weighted SAW Filter [12].

sections parallel to propagation direction, are identical in magnitude. Track responses do differ in phase due to their different group delays.

Allowable transducer time responses are constrained by the fact that track time responses must be uniform. A result of this constraint is the fact that distance weighted filter transducers may not have time sidelobes.

Generalized Distance Weighting. An example of a generalized distance weighted transducer is shown in Figure 9. Generalized distance weighted transducers may assume most arbitrary transducer weighting functions.

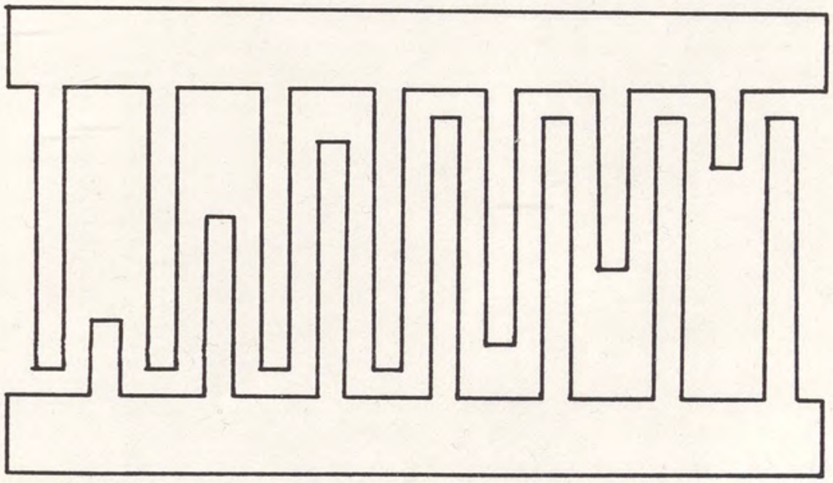


Figure 9. Generalized Distance Weighted SAW Transducer.

For half of the transducer time response, the generalized distance weighted transducer resembles the simple apodized transucer described earlier. Active electrodes, shorted to the active bus bar, are proportional in length to their corresponding positive tap weights while reference electrodes span the acoustic beamwidth. For the second half of the transducer, however, reference electrodes are proportional in length to their corresponding negative sampled impulse response values while positive electrodes are of equal length across the acoustic beamwidth. Regions of finger overlap are concentrated first on one side of the beam and then on the other, resulting in a general sharing of finger overlap areas across the beamwidth in the transducer.

Although this geometry has been generalized for arbitrary time impulse responses, it resembles the transducer geometries in Yamanouchi's distance weighted filters. The difference is that acoustic beam tracks in generalized distance weighed transducer vary in magnitude as well as spatial phase. Generalized distance weighted transducers will be studied in greater detail in Chapter IV.

Aperiodic Transducers

Spatially weighted transducers may also be aperiodic. In dispersive transducers, the electrode periodicity is maintained parallel to the acoustic beamwidth but varies along the direction of acoustic propagation [13]. A dispersive transducer is illustrated in Figure 10. Slanted finger transducers have fingers with constant periodicity parallel to the axis of propagation but varying periodicity across the transducer acoustic beamwidth [14]. A slanted finger transducer is illustrated in Figure 11.

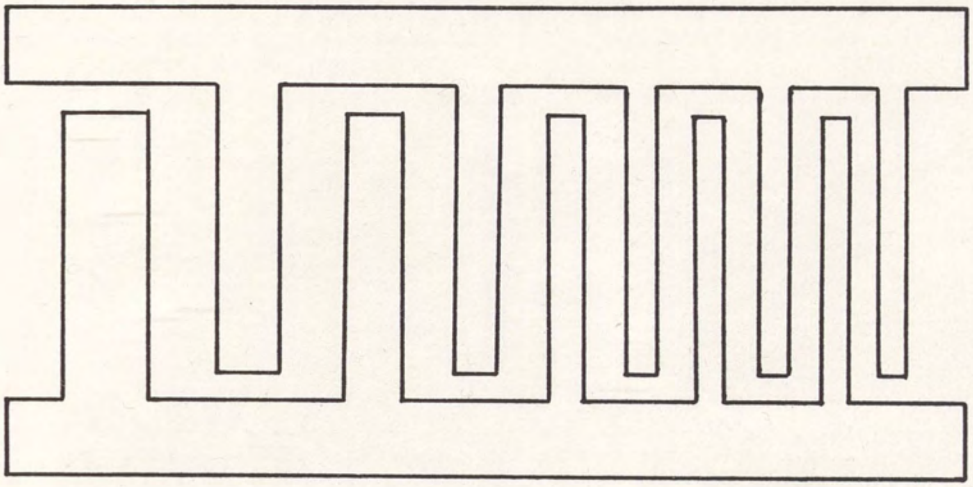


Figure 10. Dispersive SAW Transducer.[13]

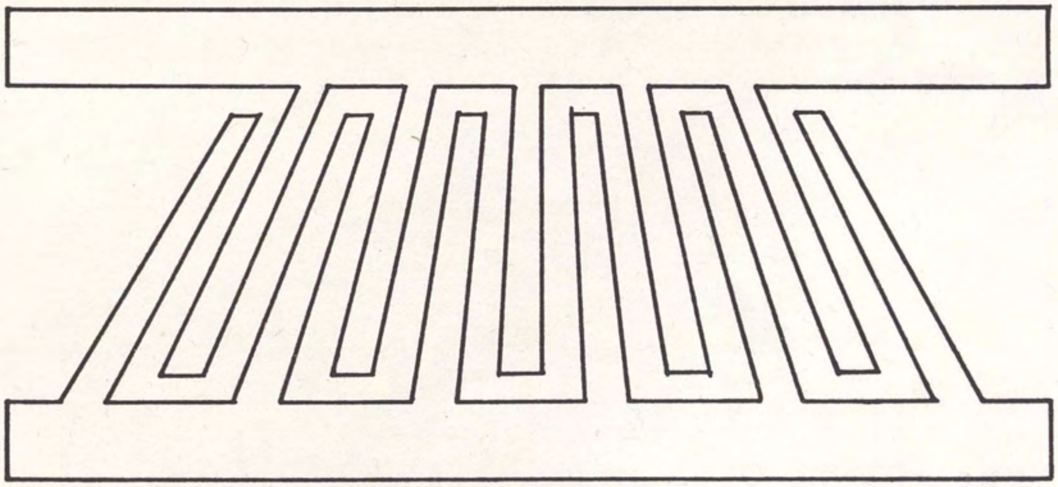


Figure 11. Slanted Finger SAW Transducer [14].

CHAPTER III

ANALYSIS STRATEGY

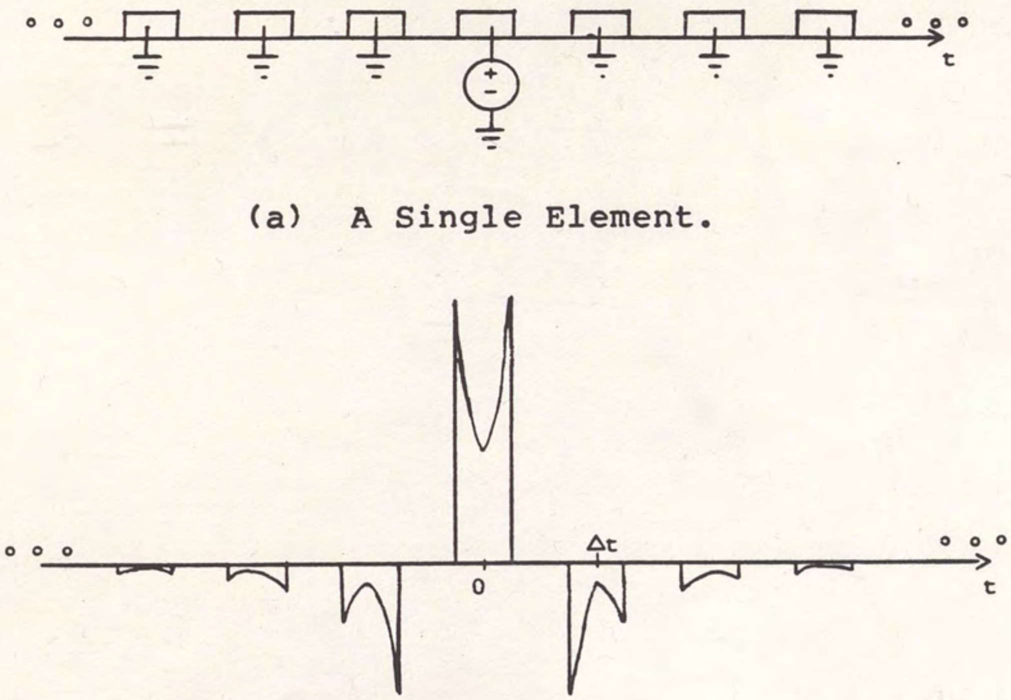
The frequency response of an apodized SAW transducer may be approximated, to a first order, by the Fourier Transform of the charge distribution on the electrodes resulting from the application of a voltage potential to one bus bar with respect to the other and assuming no piezoelectricity [5]. Superposition theory may be used to evaluate this charge distribution and its Fourier transform [6]. Superposition may also be used to simulate the transducer response on a track by track basis and then to include the effects of apodization loss, the transducer equivalent circuit and bidirectionality. Each of these are discussed in the following sections. The analysis strategy developed will then be used in Chapter IV to simulate frequency responses of apodized transducers of beam centered, slant centered, beam spread and generalized distance weighting geometries.

Impulse Response Model

The static charge distribution on the electrodes of a SAW transducer may be thought of as the convolution of a suitably chosen element factor with its corresponding array factor [6]. The Fourier transform of the element factor is referred to as the element factor response. The Fourier transform of the array factor is called the array factor response.

Element Factor

Analysis of a transducer begins with the frequency domain analysis of a single active electrode surrounded by an infinite number of ground electrodes, as depicted in Figure 12. The element factor response is obtained from the Fourier transform of the resulting charge distribution on this element. Details are presented in Reference [15].



(a) A Single Element.

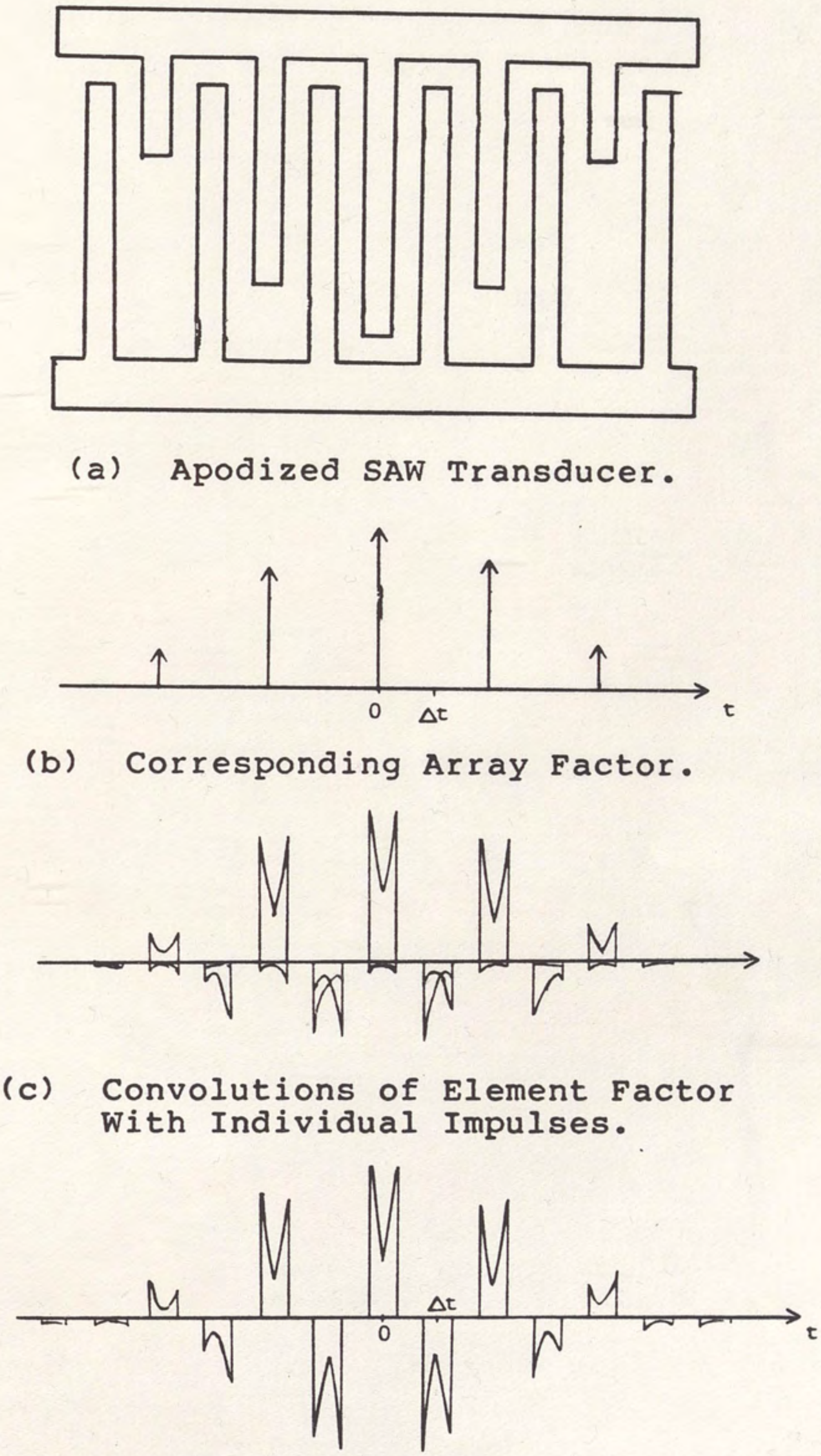
(b) Resulting Charge Distribution.

Figure 12. Evaluation of the Element Factor.

Array Factor

A SAW transducer may be modelled as the superposition of elements such as the one illustrated in Figure 12(a). One element would exist at every active electrode. The charge distribution under an entire device could then be modelled as the convolution of the charge distribution for a single element with an array of positive impulses, as shown in Figure 13. Each positive impulse in the array factor repesents an active finger. The Fourier transform of the transducer charge distribution would therefore be the product of the element factor response and the Fourier transform of the positive impulse array. The Fourier transform of this positive impulse array is referred to as the array factor response.

For this model to be successful, it is important for the array factor to be properly formed. Each time impulse corresponds to a positive electrode. The spacing of impulses in time should be proportional to the spacing of positive fingers in the transducer. Each impulse must be weighted in proportion to its corresponding electrode length.



(a) Apodized SAW Transducer.

(b) Corresponding Array Factor.

(c) Convolutions of Element Factor With Individual Impulses.

(d) Corresponding Charge Distribution.

Figure 13. Charge Distribution Associated With a SAW Transducer.

Inconsistency of Finger Superposition

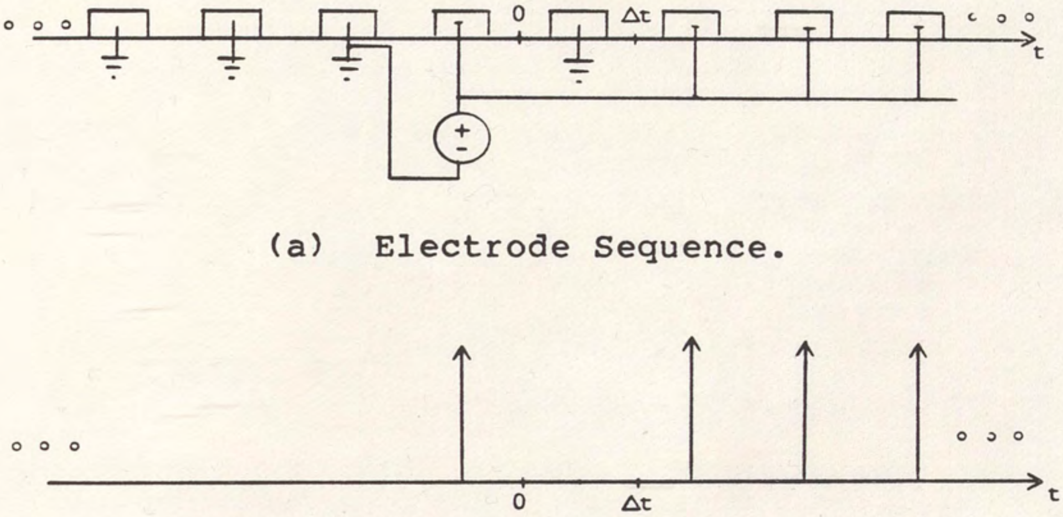
It will be shown later in this chapter that by using track analysis, an apodized transducer may be modelled as the parallel combination of several uniform transducers. Consider the uniform transducer illustrated in Figure 14. A discrepancy arises in the analysis of this transducer due to the fact that it begins with a series of ground electrodes and ends with a series of active fingers. An infinite series of active fingers following a ground electrode should only produce acoustic energy near that ground electrode. This theoretical infinite series of fingers is approximated in a transducer by three or more fingers, which are enough to cause the electric fields and charge distributions to decay to a negligible amount.

To correspond theoretically to this physical model, the array factor response for an infinite number of active taps must converge to the approximation of 3 consecutive active taps.

The array factor, AF(t), for an infinite sequence of active fingers beginning at a zero time reference is

$$AF(t) = \delta(t) + \delta(t - \Delta t) + \delta(t - 2\Delta t) + \circ\circ\circ \quad (1)$$

or



(a) Electrode Sequence.

(b) Corresponding Array Factor.

Figure 14. Odd Symmetry Uniform SAW Transducer.

$$AF(t) = \sum_{n=0}^{\infty} \delta(t - n\Delta t) \qquad (2)$$

Since

$$\mathcal{F}\left\{\delta(t - n\Delta t)\right\} = \exp(j2\pi nf\Delta t) \qquad (3)$$

and since we are operating in a linear system, the array factor response is evaluated as

$$AF(f) = \sum_{n=0}^{\infty} \exp(j2\pi nf\Delta t) \qquad (4)$$

which will not converge. The finger superposition method therefore fails to model an infinite sequence of positive taps which follow or precede a negative tap.

The finger element, as illustrated in Figure 12, shows that the finger superposition impulse response model cannot represent an infinite sequence of active fingers. Each active finger represents an element such as that shown in Figure 12(b). No matter how long the finger array is, the last finger will be followed by an infinite number of reference fingers. This final transition from positive to ground theoretically predicts acoustic generation at a point where physically there is no acoustic generation.

Development of Overlap Superposition

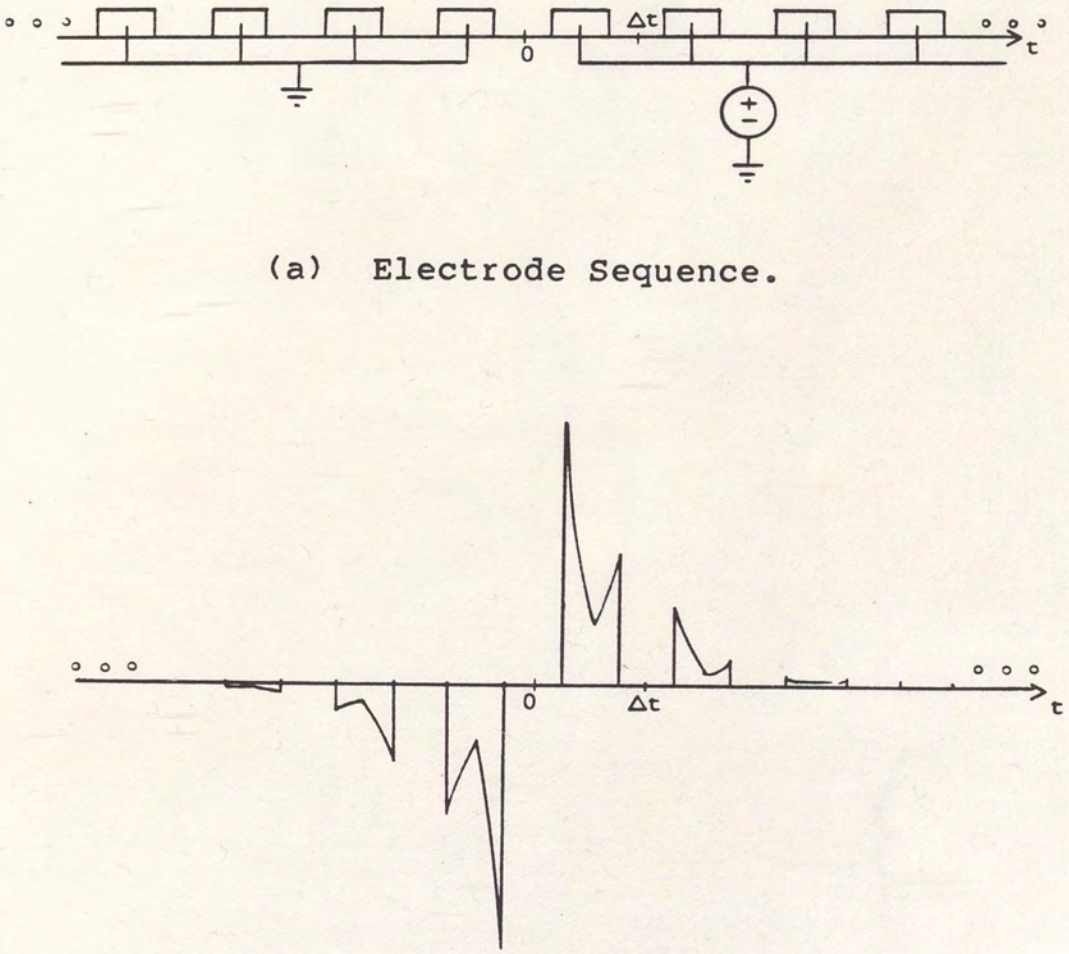
An element which lends itself to the modelling of abrupt sequences of active taps is the overlap element, as depicted in Figure 15 [5]. The overlap element consists of an infinite sequence of reference electrodes followed by an infinite sequence of active fingers. The overlap element factor, denoted OEF(t), is a function of the charge distribution associated with the overlap element and is seen in Figure 15 to decay on either side of the overlap. The overlap element factor response, denoted OEF(f), is the Fourier transform of OEF(t).

Associated with the overlap element factor is the overlap array factor, denoted OAF(t). The overlap array factor consists of a train of impulses, each coincident with a finger overlap, as shown in Figure 16. The overlap array factor is formed by assigning positive impulses to overlaps with a reference electrode on the left and an active electrode on the right, while assigning negative impulses to overlaps with an active finger to the left of a reference finger. For an apodized transducer, the impulses are weighted in magnitude in proportion to the overlap lengths. Impulses associated with array factors of uniform transducers will have uniform magnitudes.

The overlap array factor response, designated OAF(f), is the Fourier transform of OAF(t). The charge distribution associated with a track or transducer is obtained by the convolution OAF(t) (*) OEF(t), as shown in Figure 16(c) and 16(d). This charge distribution should be similar to that found through conventional finger superposition for the same transducer as shown in Figure 13. The corresponding frequency response is the product of their responses OAF(f) • OEF(f).

Derivation of the Overlap Element Factor Response

The finger element factor response EF(f) has been evaluated analytically [15]. The overlap element factor response may be solved as a function of the finger element factor response, which is already known.

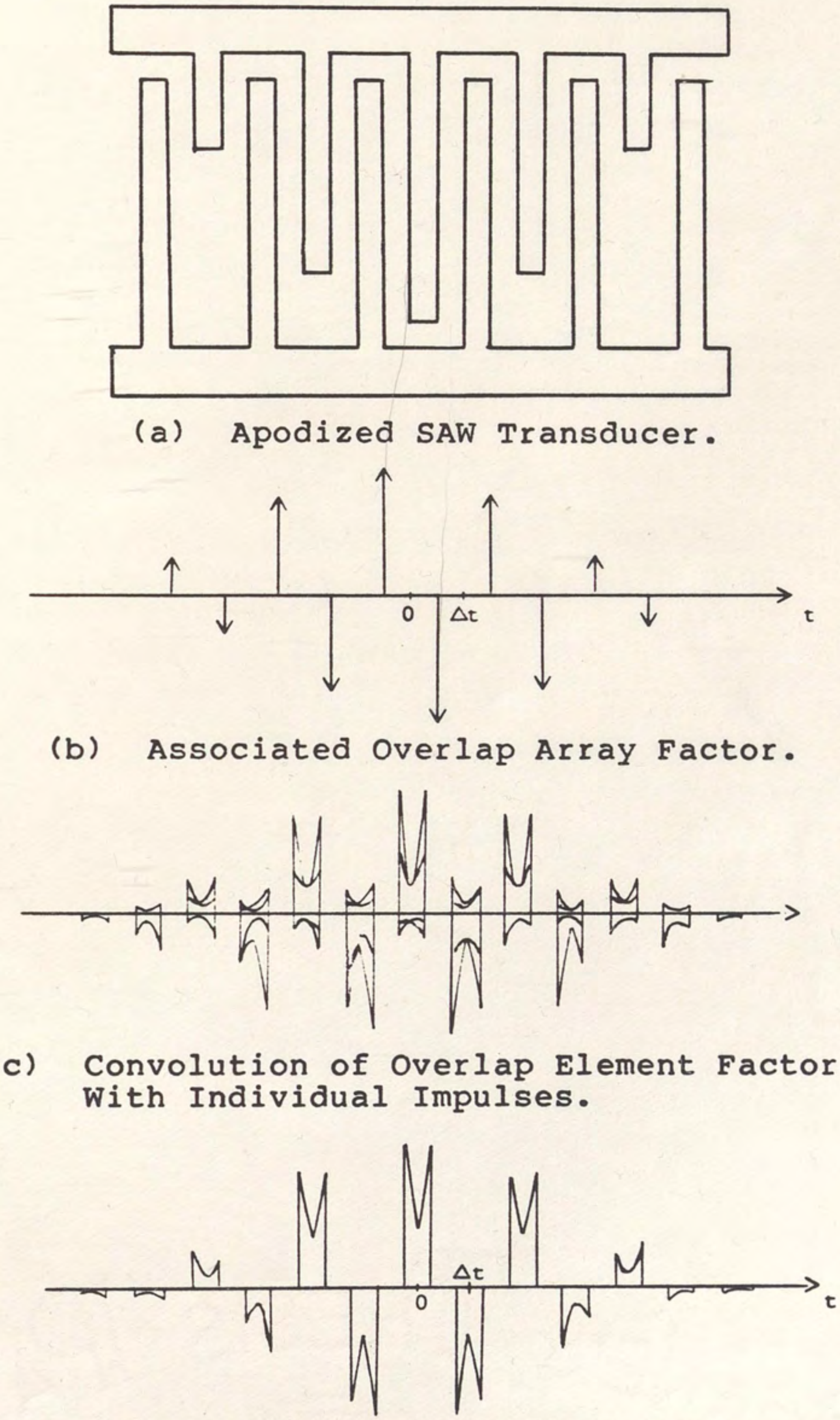


(a) Electrode Sequence.

(b) Resulting Charge Distribution.

Figure 15. Overlap Element Factor.

Consider the overlap superposition analysis of a single finger element, as shown in Figure 17. Both charge superposition methods should yield the same distribution for the same transducer, therefore



(a) Apodized SAW Transducer.

(b) Associated Overlap Array Factor.

(c) Convolution of Overlap Element Factor With Individual Impulses.

(d) Corresponding Charge Distribution.

Figure 16. Charge Distribution Associated With an Apodized SAW Transducer by Overlap Analysis.

$$OEF(t)\ (*)\ OAF(t) = EF(t)\ (*)\ AF(t) \qquad (5)$$

The equivalent frequency domain equation is

$$OEF(f) \circ OAF(f) = EF(f) \circ AF(f) \qquad (6)$$

For a single finger element at t = 0,

$$AF(t) = \delta(t) \qquad (7)$$

Therefore

$$AF(f) = 1 \qquad (8)$$

The overlap array factor which corresponds to the finger element is

$$OAF(t) = \delta(t + \Delta t/2) - \delta(t - \Delta t/2) \qquad (9)$$

applying a Fourier Transform yields

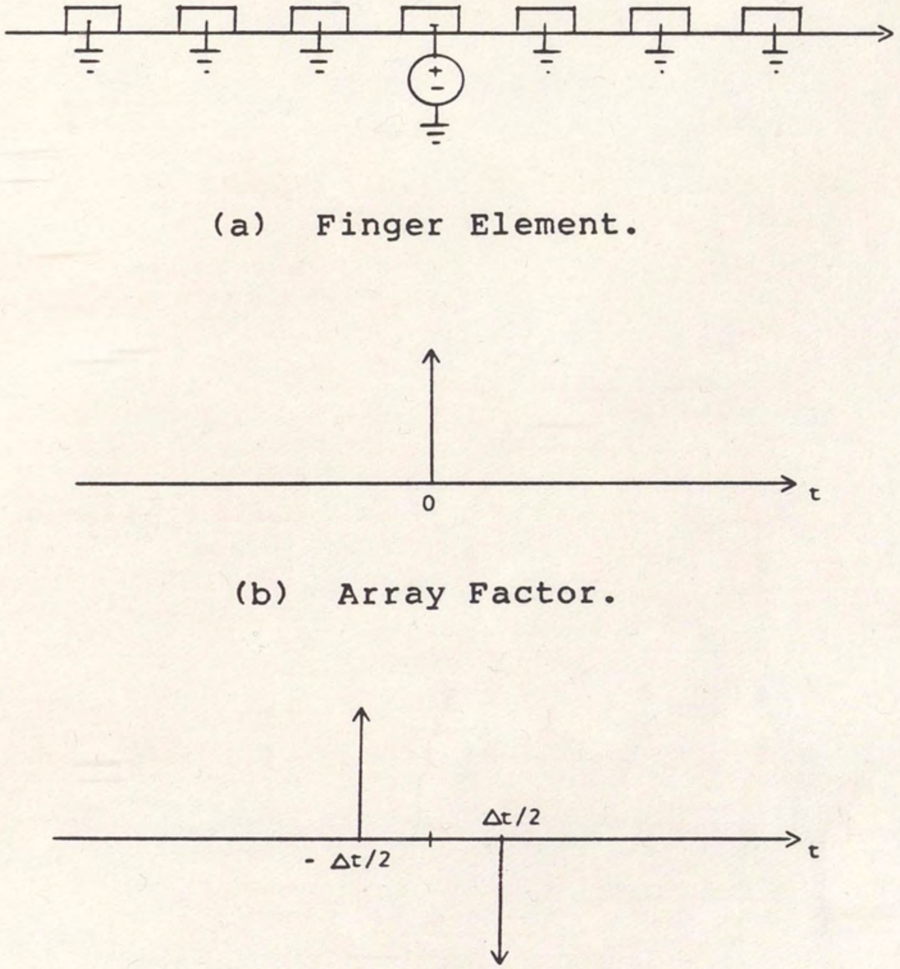
$$OAF(f) = \exp(j\pi f \Delta t) - \exp(-j\pi f \Delta t) \qquad (10)$$

or

$$OAF(f) = 2\ j\ \sin(\pi f \Delta t) \qquad (11)$$

Substituting equations (8) and (11) into equation (6) yields

$$2j\ \sin(\pi f \Delta t)\ OEF(f) = EF(f) \qquad (12)$$



(a) Finger Element.

(b) Array Factor.

(c) Overlap Array Factor.

Figure 17. Analysis of a Single Element.

or

$$OEF(f) = -(j/2)\,[\sin(\pi f \Delta t)]^{-1}\,EF(f) \qquad (13)$$

Since

$$EF(f) \quad = \quad 2 \quad k \; \frac{P_{n}(\,\cos(\eta\pi)\,)}{P_{s_{o}-1}(\,-\cos(\eta\pi)\,)} \; \sin(\pi s_{o}) \qquad (14)$$

[15]

where

$s \;=\; f\,\Delta t \;=\; s_{o} + n, \qquad 0 \leqq s_{o} \leqq 1$ and n is an integer;

η is the metallization ratio;

P_{v} is the Legendre function of order v; and

k is a normalizing constant;

then

$$OEF(f) = -(j/2)\,[\sin(\pi s)]^{-1}\;2k\;\frac{P_{n}(\,\cos(\eta\pi)\,)}{P_{s_{o}-1}(\,-\cos(\eta\pi)\,)}\;\sin(\pi\, so) \qquad (15)$$

or

$$OEF(f) \;=\; j\;k\;(-1)^{(n+1)}\;\frac{P_{n}(\,\cos(\eta\pi)\,)}{P_{s_{o}-1}(\,-\cos(\eta\pi)\,)} \qquad (16)$$

Track Analysis

Apodized SAW transducers launch acoustic energy unevenly across the transducer acoustic beamwidth. Neglecting second order effects, the transducer frequency response is modelled as the sum of responses at all points across the beam as seen by a uniform receiving transducer. Track analysis is a technique in which an apodized transducer is described as the superposition of many uniform transducers in parallel. Track analysis assumes a self collimating beam with negligible diffraction effects.

Infinite Track Model

The infinite track model treats the transducer mathematically as an infinite number of uniform transducers of differential width. Superposition of tracks is accomplished by continuous integration.

An apodized SAW transducer and coordinate system is shown in Figure 18. The acoustic beamwidth is normalized to unity. Each point in the x coordinate axis has a corresponding time impulse response h(x,t). The corresponding x coordinate frequency response, H(x,f), is related to h(x,t) by the Fourier transform as

$$H(x,f) = \int_{-\infty}^{\infty} h(x,t)\ \exp(-j\ 2\pi\ ft)\ dt \qquad (17)$$

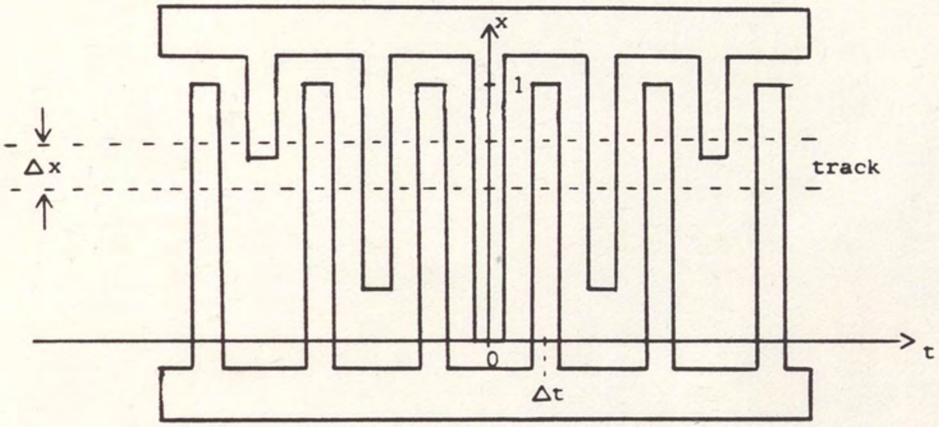


Figure 18. Apodized SAW Transducer Under Track Analysis.

The transducer impulse response may be found by integrating h(x,t) across the beam:

$$h(t) = \int_0^1 h(x,t)\ dx \tag{18}$$

Similarly, the transducer frequency response may be found by integrating H(x,f) across the beam:

$$H(f) = \int_0^1 H(x,f)\ dx \tag{19}$$

Substituting (17) into (19) and integrating with respect to x yields

$$H(f) \;=\; \int_{-\infty}^{\infty} h(t)\, \exp(-j\; 2\pi\, ft)\; dt \qquad (20)$$

Finite Track Model

The frequency response of a transducer may be approximated by a summation of equal width tracks which are assumed to be uniform.

$$H(f) \;\overset{\sim}{=}\; \sum_{i\,=\,1}^{N} H(\;(i-1/2)\circ\Delta x,\; f\;)\circ\Delta x \qquad (21)$$

where N is the number of tracks, $\Delta x = 1/N$, and $H(\;(i-1/2)\circ\Delta x,\; f\;)$ represents $H(x,f)$ evaluated at the center of the i'th track. Letting $H(\;(i-1/2)\circ\Delta x,\; f\;)$ equal $H(i,f)$ yields:

$$H(f) \;\overset{\sim}{=}\; \sum_{i\,=\,1}^{N} H(i,f)\circ\Delta x \qquad (22)$$

Minimum Track Model

Inaccuracies in the finite track model arise from the fact that the samples may not be uniform in a track. Figure 18 shows a track with nonuniform taps which results in model errors. In order to increase numerical accuracy, more tracks may be evaluated at the expense of calculation time. The finite track model presents an inherent tradeoff between accuracy and calculation time.

A minimum track model may be used to minimize calculation time while maintaining exact accuracy [11]. In the minimum track model, as shown in Figure 19, an arbitrary transducer is divided into unequal tracks whose limits each coincide with the tip of an electrode. Because of this constraint on the track limits, each track may be unique and must be uniform. The number of track simulations performed is minimized under the requirement of full accuracy.

The transducer frequency response now becomes the summation of unequal width track responses as

$$H(f) = \sum_{i=1}^{N} H(i,f) \circ \Delta x_i \qquad (23)$$

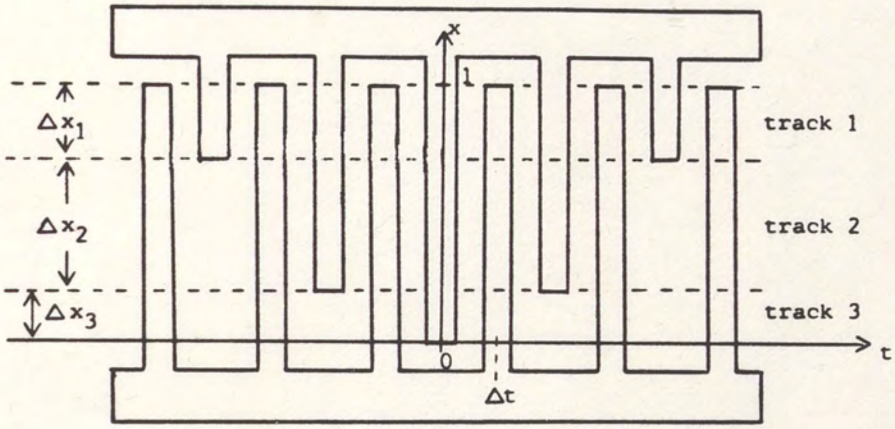


Figure 19. Apodized SAW Transducer Under Minimum Track Model Analysis.

where N is now the minimum number of tracks required to maintain full accuracy.

Radiated Power

Applying a normalized voltage waveform at any frequency f to an apodized transducer results in the radiation of acoustic signals across the transducer of amplitude in proportion to H(x,f). The power density of the radiated signal is proportional to the square of the amplitude of the x dependent transfer function:

$$P(x,f) = | H(x,f) |^2 \tag{24}$$

[16]

Where P(x,f) is the normalized x dependent power. Total radiated power may be integrated across the beamwidth:

$$P_{out}(f) = \int_{0}^{1} | H(x,f) |^{2} dx \qquad (25)$$

[16]

where $P_{out}(f)$ is the normalized total radiated power. Total radiated power may also be summed with the same accuracy using the minimum track model:

$$P_{out}(f) = \sum_{i=1}^{N} | H(i,f) |^{2} \circ \Delta x_{i} \qquad (26)$$

Apodization Loss

Apodized transducers typically spread the acoustic signal unevenly across the acoustic beamwidth. Uniform receiving transducers operate efficiently on acoustic wavefronts whose amplitudes do not vary with respect to x. Apodization loss represents the inefficiency associated with uneven acoustic wave fronts.

Apodization loss for a transducer may be found by applying a normalized continuous wave signal at center frequency f_o [16]. Since $H(x,f_o)$ is then the x dependent wave amplitude, the signal detected by a uniform receiving transducer may be determined through integration as

$$H_{rec.}(f_o) = \int_0^1 H(x,f_o)\ dx \qquad (27)$$

The normalized received power is the square of the received signal:

$$P_{rec.}(f_o) = \left| \int_0^1 H(x,f_o)\ dx \right|^2 \qquad (28)$$

Apodization loss, denoted AL, is defined as the ratio of received power to radiated power at center frequency for the case of a uniform receiving transducer. Apodization loss is given by

$$AL = \frac{\text{in}\ \left| \int_0^1 H(x,f_o)\ dx \right|^2 = \underset{\nearrow}{E(x^2)}\ [illegible]}{\text{out}\ \underbrace{\int_0^1 | H(x,f_o) |^2\ dx}_{\text{total}} \qquad \underset{\nearrow}{E^2[x]}} \qquad (29)$$

[16]

Apodization loss may be defined through the minimum track model as

$$AL = \frac{\left| \sum_{i=1}^{N} H(i,f_o) \circ \Delta x_i \right|^2}{\sum_{i=1}^{N} | H(i,f_o) |^2 \circ \Delta x_i} \qquad (30)$$

Apodization loss is calculated at center frequency.

Acoustic Conductance

It was shown previously that the normalized power radiated by an apodized transducer may be evaluated numerically as a function of frequency. Applying conservation of power to the transducer, the acoustic power radiated must be equal to the absorbed electrical power:

$$G_a(f)\, V^2 = P_{rad}(f) = G_o\, P_{out}(f) \qquad (31)$$

where "V" is the voltage applied to the transducer, $G_a(f)$ is the acoustic conductance, $P_{rad}(f)$ is the total radiated power, $P_{out}(f)$ is the normalized radiated power previously defined, and G_o is a function of substrate parameters:

$$G_o = 2\, K^2\, W\, C_o\, f_o \qquad (32)$$

where K^2 is the acoustic coupling coefficient, C_o is the substrate capacitance in picofarads per centimeters, W is the transducer width and f_o is the center frequency [5,11,17,18]. Substituting equation (32) into (31) and noting that the normalized radiated power was evaluated with "V" set equal to one yields:

$$G_a(f) = 2 K^2 C_o W f_o P_{out}(f) \qquad (33)$$

Equivalent Circuit Model

The complete frequency response of a transducer includes the behavior of the transducer in its electrical circuit. A transducer may be represented electrically in terms of a lumped parameter model as shown in Figure 20 [17].

$G_a(f)$ is the acoustic conductance described in the previous section. $B_a(f)$ is the frequency dependent

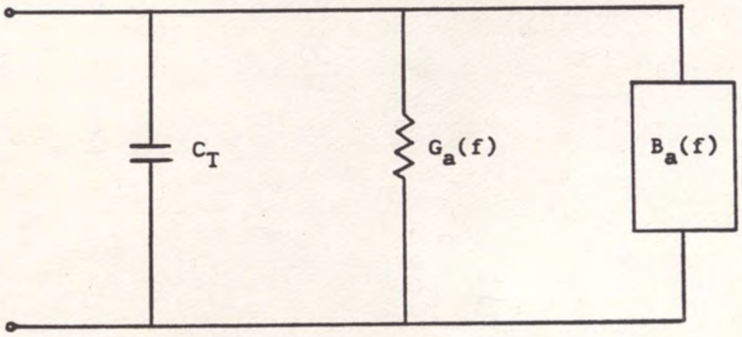


Figure 20. SAW Transducer Lumped Parameter Model [17].

susceptance related to $G_a(f)$ by the Hilbert Transform [17]:

$$Ba(f) = (1/\pi) \int_{-\infty}^{\infty} \frac{G_a(u)}{(u-f)} du \qquad (34)$$

[17]

C_T is the transducer capacitance, which may be determined by summing electrode overlap capacitances throughout the transducer.

Transducer Frequency Response

In summary, the transducer frequency response may be approximated, to a first order, as the product of the proper array factor response, element factor response, apodization loss, bidirectional loss and electrical response. As described earlier, the array and element factors used must be properly chosen and mutually consistent. These responses may be evaluated using track superposition models. Apodization loss is evaluated at center frequency and is independent of frequency. Bidirectional loss is 3 dB. which occurs because half the power is radiated from each side of the transducer. The electrical response may be calculated as the response of the previously described equivalent circuit network in the driving circuit.

CHAPTER IV

RESULTS

Two types of filters are considered. The first is an Eigen window filter, representing a time impulse response with no time sidelobes. The second is a low shape factor filter with several time sidelobes.

Each filter type is analyzed with five different weighting techniques. Analysis is first performed for the case of the filter with transducers implemented by amplitude weighting. The case of amplitude weighting is used as a standard of comparison for the four remaining weighting techniques. These four weighting techniques are all apodization schemes which were discussed in Chapter II. The four apodization geometries to be analyzed are beam centered overlap, slant centered overlap, beam spread overlap and generalized distance weighting.

Analysis was carried out in two steps. Radiation profiles, apodization loss and acoustic conductance were obtained through "STRUCTURECAD." Total device analysis was performed using SAWCAD [19].

STRUCTURECAD is a set of Fortran programs designed to analyze periodic spatially weighted transducers using the minimum track model described in Chapter III. STRUCTURECAD is capable of analyzing all transducer geometries under study. The routines which presently comprise STRUCTURECAD are listed in Appendix A.

STRUCTURECAD uses overlap element superposition in calculating acoustic conductance data. Resulting acoustic conductance data is downloaded into a disk file for access later by SAWCAD. STRUCTURECAD normalizes conductance data to the normalized radiated power as described in Chapter III. Radiation profile data is normalized to the root mean square center frequency acoustic signal level within the acoustic aperture. Apodization loss is dimensionless.

SAWCAD is a user friendly computer aided design system developed by the Solid State Devices and Systems Laboratory to design and analyze SAW filters on the basis of finger element superposition as described in Chapter III [19]. Complete device analysis obtained using SAWCAD includes the electrical responses of the transducers as described in Chapter III. SAWCAD was modified in order to access the acoustic conductance data generated by STRUCTURECAD. This modification took the form of "CALCOND," a routine which was added to SAWCAD in order to access acoustic conductance files. CALCOND is listed in Appendix B.

Eigen Window Filter

The first type of filter evaluated was a SAW filter implementation of an Eigen window function [20,21]. The first transducer is an unweighted transducer. The second transducer represents the impulse response of the desired window function modulated to the center frequency of 79 MHz. The Eigen window filter was chosen for analysis to represent a filter whose time impulse response has no sidelobes. The time response file used to generate the various Eigen window transducer models is illustrated in Figure 21. Figure 22 shows the resulting total Eigen pulse transducer frequency response. A complete set of analysis data for the amplitude weighted Eigen window device may be found in Appendix C.

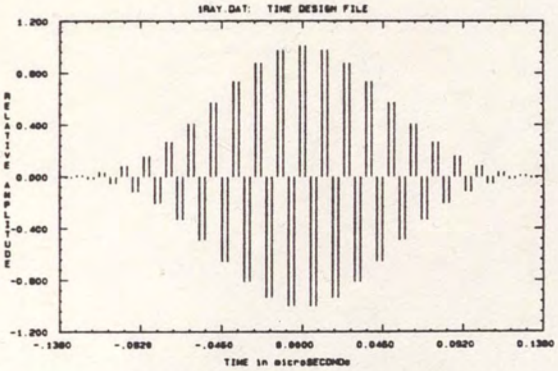


Figure 21. Weighting Function for Eigen Window Transducers.

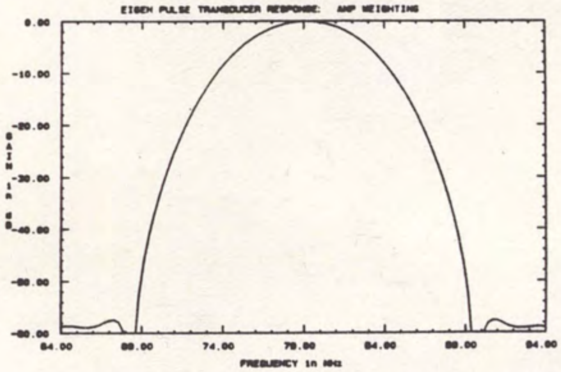


Figure 22. Amplitude Weighted Eigen Window Transducer Response.

Radiation Profiles

Figure 23 illustrates the radiation profile data for the Eigen window transducers. The beam centered overlap radiation profile shows a gross nonuniformity of energy across the acoustic aperture. The slant centered overlap transducer shows a much more uniform radiation profile. The acoustic energy radiated across the apertures of the beam spread overlap and generalized distance weighted transducers are almost constant.

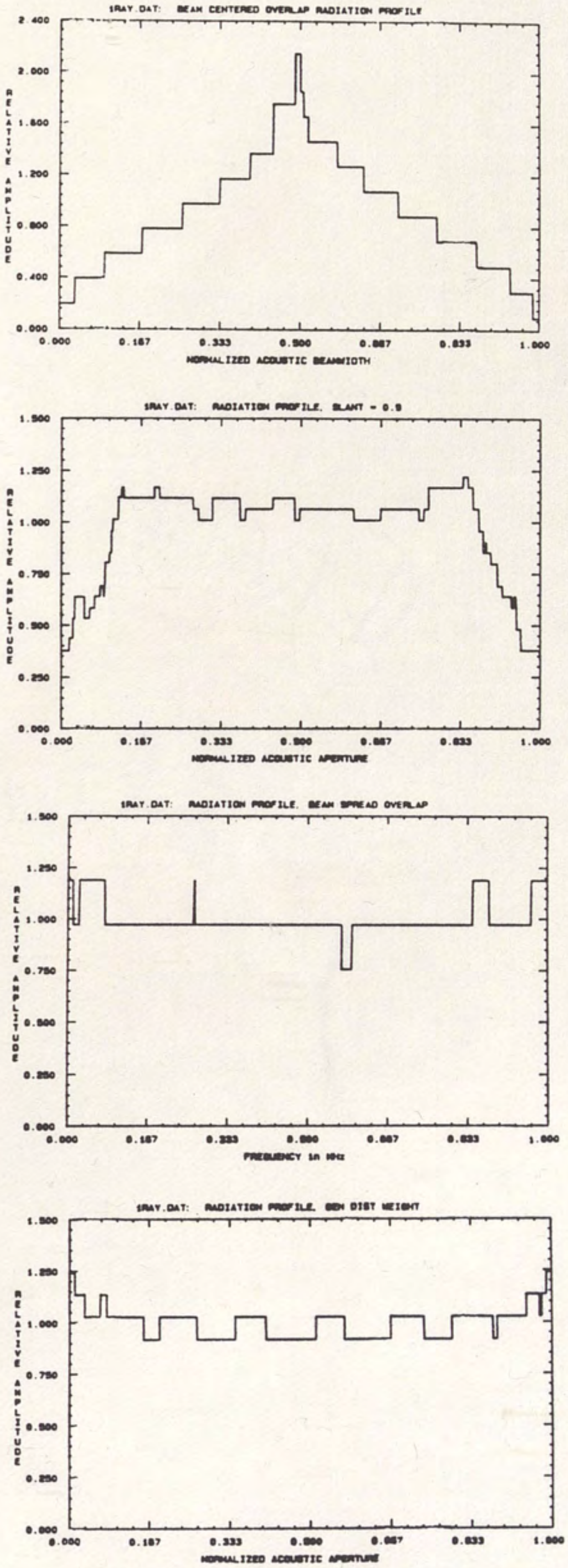
Apodization Loss

Apodization loss data for the Eigen window transducers is presented in Table I. Transducer apodization loss is seen to increase with radiation profile nonuniformity.

TABLE I

APODIZATION LOSS DATA FOR APODIZED
EIGEN WINDOW TRANSDUCERS

TRANSDUCER TYPE	APODIZATION LOSS
Beam Centered Overlap	1.8139 dB.
Slant = 0.9	0.4787 dB.
Beam Spread Overlap	0.0580 dB.
Gen. Dist. Weighting	0.0478 dB.



(a) Beam Centered Overlap Transducer.

(b) Slant Centered Overlap Transducer.

(c) Beam Spread Overlap Transducer.

(d) Generalized Distance Weighted Transducer.

Figure 23. Center Frequency Radiation Profiles for Apodized Eigen Window Transducers.

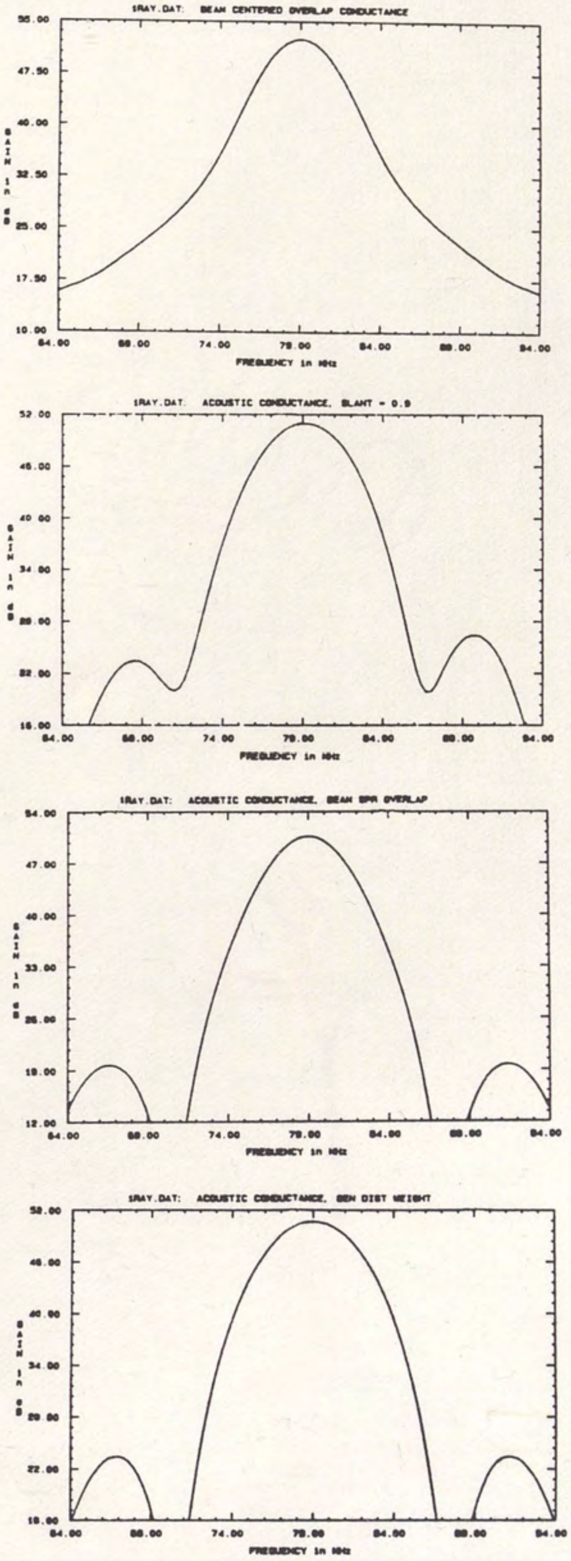
Conductance

Conductance plots for the various geometrical configurations of the Eigen window transducer are shown in Figure 24. These plots may be compared to the amplitude weighted Eigen window transducer response by referring to Figure 23. It is noticed that as the center frequency radiation profiles approach uniformity, the shape of the frequency dependent acoustic conductance begins to approach the shape of the expected amplitude weighted conductance.

Wideband conductance data for these transducers is presented in Appendix C. It is noticed that the sidelobes of the beam spread overlap transducer conductance roll off much more slowly than those associated with the two other geometries under study.

Transducer Responses

Figure 25 shows the resulting total narrowband transducer frequency responses associated with each version of the Eigen window transducer. The only response significantly different from that of the amplitude weighted transducer is that of the slant centered overlap transducer. The slant centered overlap transducer has first sidelobes which are 2 and 4 dB. higher than those associated with the other geometries. The shape of the main lobe of the slant centered apodized transducer



(a) Beam Centered Overlap Transducer.

(b) Slant Centered Overlap Transducer.

(c) Beam Spread Overlap Transducer.

(d) Generalized Distance Weighted Transducer.

Figure 24. Acoustic Conductance for Apodized Eigen Window Transducers.

response also differs from the shape of the other transducers by as much as 2 dB. at some points.

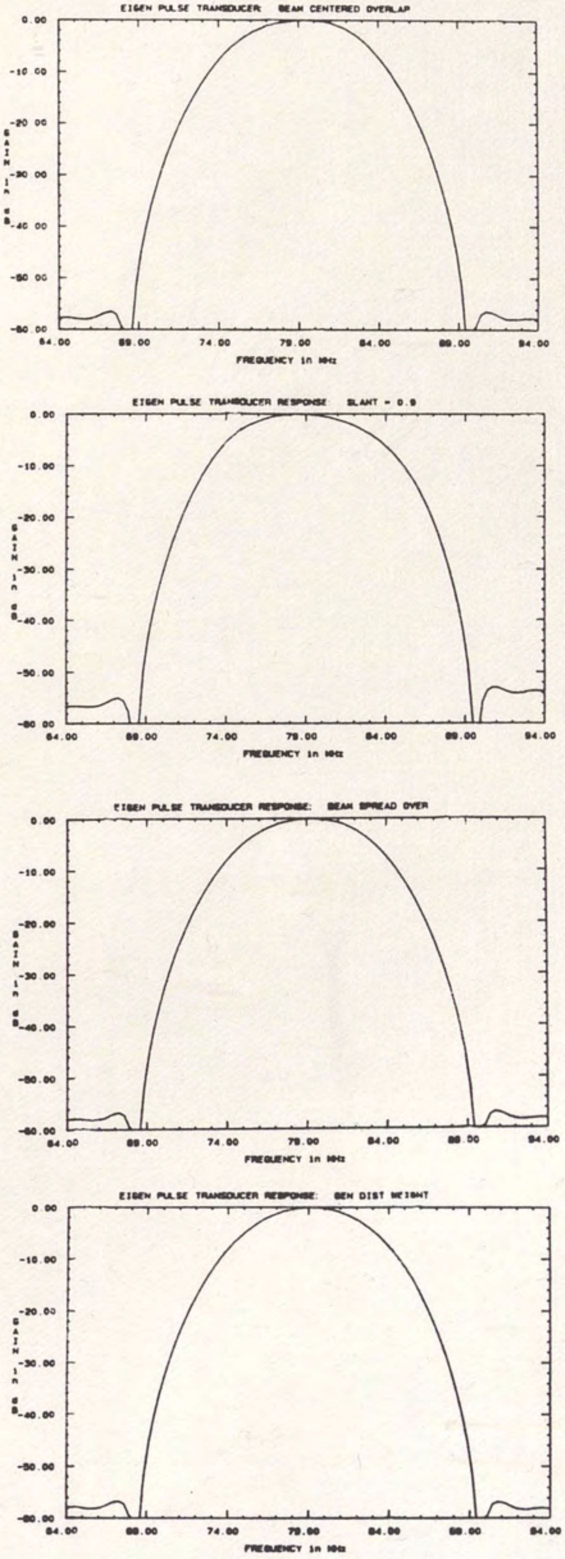
Summary

Apodization loss and acoustic conductance for each transducer are related to the transducer's corresponding acoustic radiation profile. Apodization loss decreases with radiation profile uniformity for this device. As the transducer radiation profiles begin to approach uniformity, the corresponding narrowband frequency dependent acoustic conductances begin to approach the case for amplitude weighting. In most cases, these changes have little effect on the transducer frequency responses.

A complete set of data obtained from the computer analysis of the five Eigen window filter devices is included in Appendix C.

Low Shape Factor Filter

The second type of filter evaluated was a low shape factor 60 MHz. bandpass filter. This filter was designed using Eigen synthesis techniques developed by Bishop and Malocha [21,22]. Each transducer is weighted in time by functions which each have ten sidelobes. The filter is designed for a 3 dB. fractional bandwidth of 10 percent and a shape factor of 1.17. This design was chosen to represent a general class of filters having time sidelobes.



(a) Beam Centered Overlap Transducer.

(b) Slant Centered Overlap Transducer.

(c) Beam Spread Overlap Transducer.

(d) Generalized Distance Weighted Transducer.

Figure 25. Eigen Window Apodized Transducer Responses.

The data included in this section represents a summary of the analysis of the first transducer of this filter as implemented by each of the four apodization techniques under study. The time weighting function used to generate the various input transducer models is illustrated in Figure 26. The amplitude weighted transducer response is given in Figure 27. Complete sets of analysis data for this filter type are included in Appendix D.

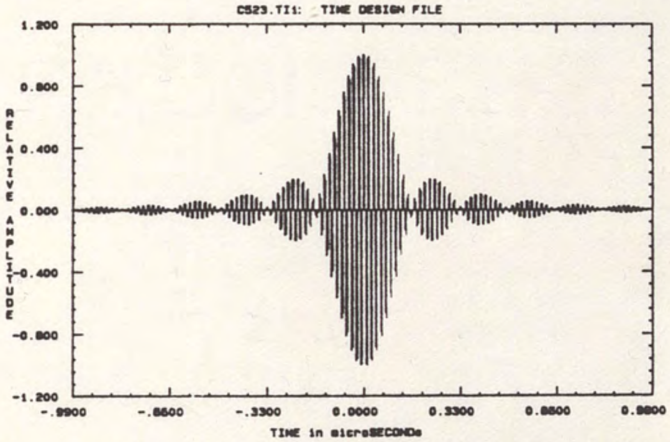


Figure 26. Weighting Function for First Low Shape Factor Filter Transducer.

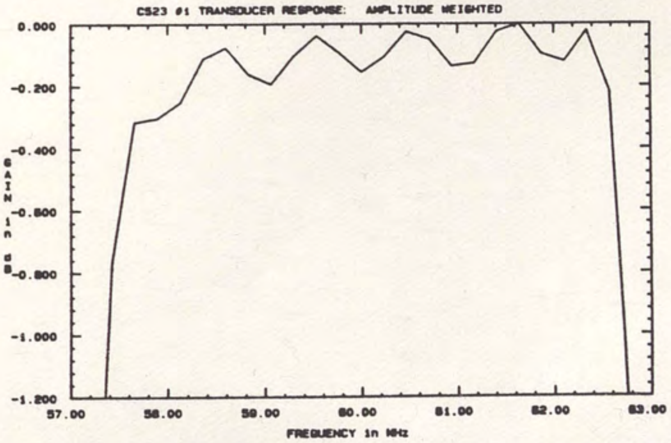


Figure 27. Amplitude Weighted Transducer Frequency Response, First Transducer, Low Shape Factor Filter.

Radiation Profiles

Figure 28 illustrates the radiation profile data for the various transducer structures weighted by the function shown in Figure 26. The beam centered overlap transducer again shows a very nonuniform beam profile. The beam profile for the slant centered overlap transducer, however, is even less uniform. Beam profiles for the beam spread overlap and generalized distance weighted transducers are somewhat more uniform, but because of the presence of time sidelobes, they are not nearly as constant as for the case of the Eigen window transducers.

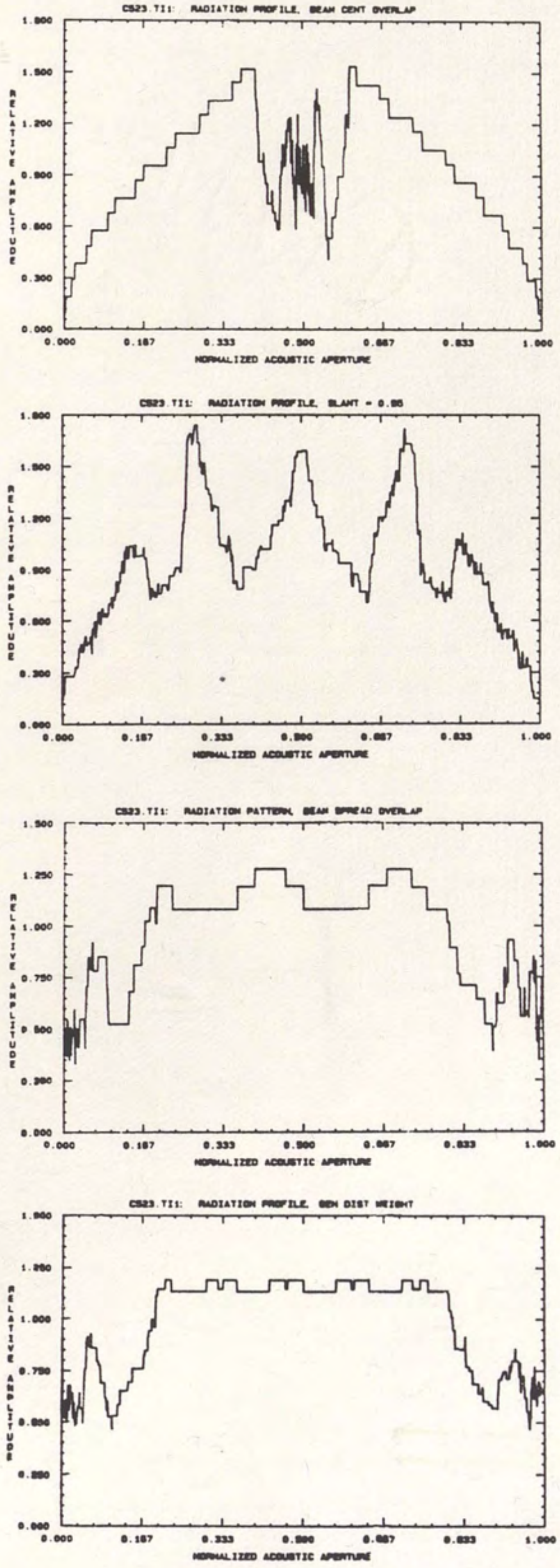
Apodization Loss

Apodization loss data for the first low shape factor filter is given in Table II. Transducer apodization loss is again seen to increase as the beam profiles become nonuniform.

TABLE II

APODIZATION LOSS DATA FOR FIRST TRANSDUCERS OF LOW SHAPE FACTOR FILTERS

TRANSDUCER TYPE	APODIZATION LOSS
Beam Centered Overlap	1.2486 dB.
Slant Centered Overlap	1.6472 dB.
Beam Spread Overlap	0.7344 dB.
Gen. Dist. Weighting	0.6828 dB.



(a) Beam Centered Overlap Transducer.

(b) Slant Centered Overlap Transducer.

(c) Beam Spread Overlap Transducer.

(d) Generalized Distance Weighted Transducer.

Figure 28. Center Frequency Radiation Profiles for First Transducer of Low Shape Factor Filters.

Conductance

Frequency dependent conductance plots for the various transducers under study are shown in Figure 29. Passband acoustic conductance ripple data is presented in Table III. Unlike the case for the Eigen window transducer, there seems to be little correlation between passband conductance and radiation profile uniformity. Appendix D again shows the high conductance sidelobes characteristic of beam spread overlap apodization.

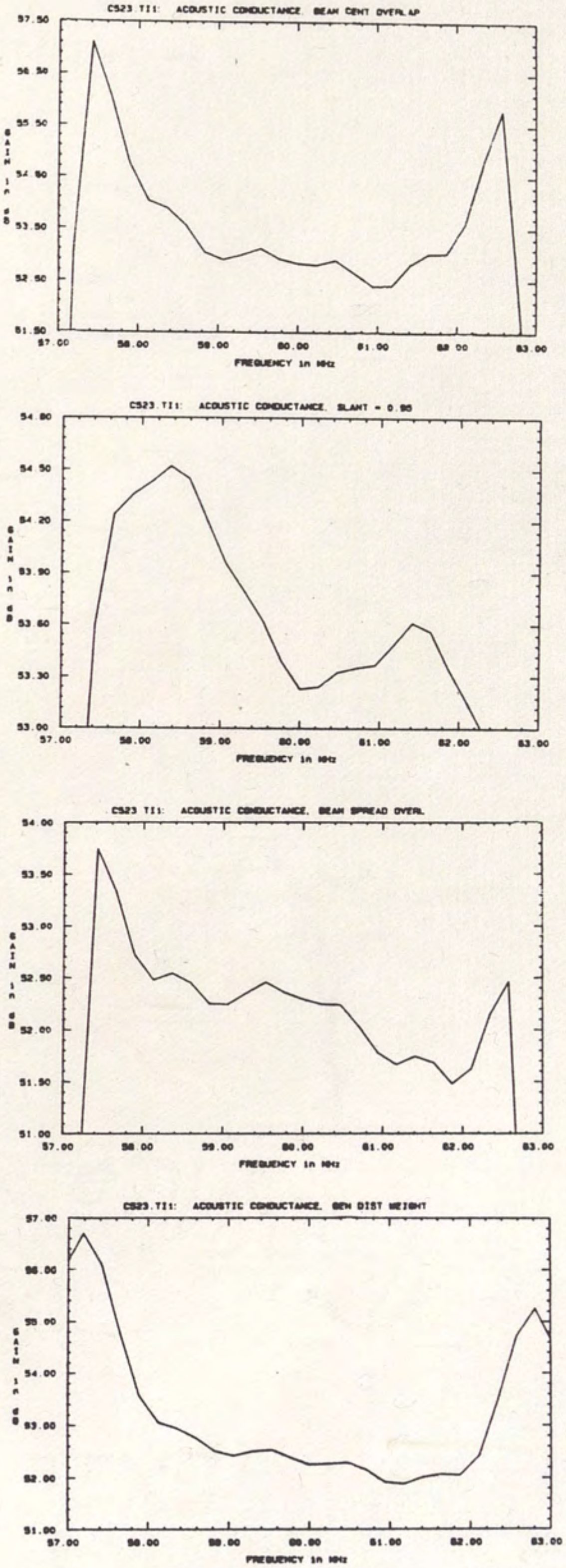
TABLE III

PASSBAND ACOUSTIC CONDUCTANCE RIPPLE DATA
FOR HIGH SHAPE FACTOR FILTER TRANSDUCER.

TRANSDUCER TYPE	PASSBAND CONDUCTANCE RIPPLE
Beam Centered Overlap	4.8 dB.
Slant Centered Overlap	1.8 dB.
Beam Spread Overlap	2.3 dB.
Gen. Dist. Weighting	4.8 dB.

Transducer Responses

Figure 30 shows the resulting narrowband transducer frequency responses associated with each version of the low shape factor filter transducer. Slight passband narrowing



(a) Beam Centered Overlap Transducer.

(b) Slant Centered Overlap Transducer.

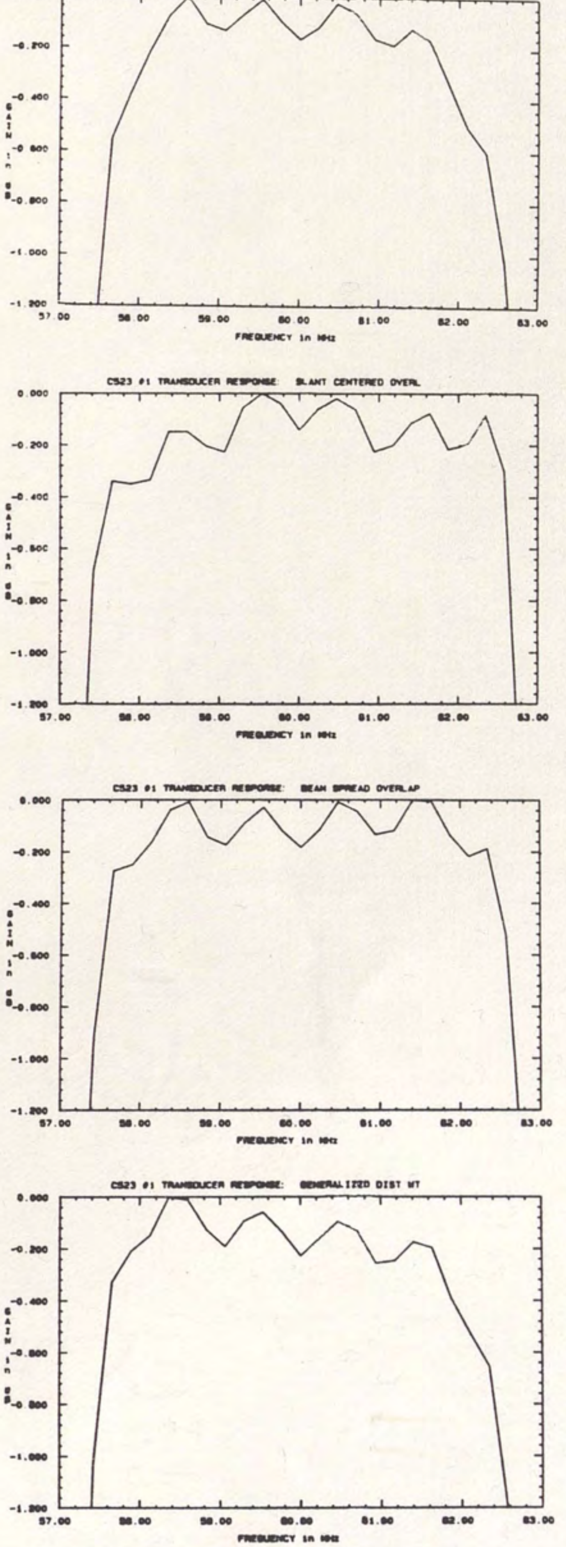
(c) Beam Spread Overlap Transducer.

(d) Generalized Distance Weighted Transducer.

Figure 29. Acoustic Conductance for Low Shape Factor Filter Transducers.

in some transducer responses may be attributed to spikes in the frequency dependent conductance at the edges of the passband. This effect is most pronounced in those transducers with high passband conductance ripple.

Each apodization geometry has the effect of attenuating some portions of the passband response while leaving other portions undisturbed. As expected, all passband transducer responses are very similar in shape to the response of the transducer under amplitude weighting.



(a) Beam Centered Overlap Transducer.

(b) Slant Centered Overlap Transducer.

(c) Beam Spread Overlap Transducer.

(d) Generalized Distance Weighted Transducer.

Figure 30. Transducer Responses for Low Shape Factor Filter Input Transducers.

Summary

Apodization loss for each transducer is clearly a function of the uniformity of that transducer's radiation profile. Ripple in the passband regions of the frequency dependent conductance, however, depends very much on the type of structure used but not necessarily on its radiation profile. Passband shape is altered to some extent as a function of acoustic conductance in all apodized transducers analyzed.

A complete set of data from the analysis of the low shape factor filter is included in Appendix D. Figures 52 and 53 illustrate the increase in accuracy of the minimum track model over the amplitude weighting approximation in predicting transducer acoustic conductance for an experimental filter. Similarly, Figure 62 illustrates how a simulation incorporating this technique for predicting conductance gave results which were more self consistent with actual device performance than the use of the amplitude weighting approximation.

The filter frequency response data included in Appendix D shows an interesting and important phenomenon. According to the analysis, some apodized transducer structure implementations of the low shape factor filter will result in less passband ripple than the amplitude weighting implementation of the same filter.

CHAPTER V

CONCLUSIONS

Various SAW transducer weighting techniques were described. Four weighting techniques were selected for detailed analysis. Two weighting techniques were beam spread overlap apodization and slant centered overlap apodization. Both of these apodization schemes are very popular among SAW filter designers. Two less commonly understood weighting techniques are the beam spread overlap apodization method and the generalized distance weighting technique. All weighting techniques studied closely are varieties of apodization, or periodic spatial weighting.

In order to analyze the transducers under study, the impulse response model was described in terms of finger element superposition. Some deficiencies found in the finger element superposition approach were circumvented by developing overlap element superposition.

Track analysis was introduced as a means of accurately calculating transducer conductance and apodization loss. A minimum track model was developed to minimize calculation time while conserving accuracy under the assumption of negligible diffraction. Calculated data for center frequency acoustic radiation profiles, apodization loss and frequency dependent acoustic conductance are obtained through these methods from "STRUCTURECAD," a computer simulation package developed to analyze the transducer structures of interest through minimum track analysis.

Data extracted from Structurecad was made available to SAWCAD in order to perform the complete filter response analysis, including network responses. Full frequency response simulations were performed on two types of filters. The first type of filter implemented an Eigen window function with no time sidelobes. The second type of filter was a low shape factor bandpass filter with several time sidelobes. Response simulations for the low shape factor slant centered overlap transducer filter were compared to actual device data. Simulations which used track based acoustic conductance data in evaluating the equivalent network responses were found to be much more self consistent with actual device performance than those evaluated on the basis of the amplitude weighting approximation typically assumed in SAWCAD.

The four transducer structures were compared on the basis of several features. No transducer geometry had a constant acoustic radiation profile for transducers implementing time sidelobes. However, both the beam spread overlap and the generalized distance weighted geometries had radiation profiles which were much more uniform than the transducers structured by the beam or slant centered overlap geometries. Transducers with the most uniform radiation profiles were found to have the least apodization loss.

In the transducers implementing no time sidelobes, the shape of the frequency dependent acoustic conductance was found to begin to approach the shape of the transducer frequency response as the center frequency radiation profile began to approach a constant. This effect could be detected only slightly in the case of transducers implementing time sidelobes because the radiation profiles were too nonuniform. It was observed in general that as apodized transducer acoustic radiation profiles approach uniform radiation their behavior approaches that of amplitude weighted transducers.

Transducer acoustic conductance was found to impact the filter frequency response, specifically passband ripple. No transducer had an acoustic conductance which reduced the passband ripple in the transducer response; however, apodized transducer filters were found in some cases to have less filter pass band ripple than their amplitude weighted transducer filter counterparts.

Beam spread overlap and generalized distance weighting are not widely used in the SAW filter industry. They are shown, however, to have merit in producing filters with reduced apodization loss and passband ripple. It is possible that passband ripple may be further reduced in low shape factor SAW filters by combining two transducers of different apodization geometries in a single filter. STRUCTURECAD and SAWCAD remain as analysis tools capable of simulating SAW filter responses under the given assumptions.

Several steps may be taken to improve the simulation capabilities of STRUCTURECAD. Calculation time is at present a factor that limits the practicality of its use. It now takes 20 minutes to calculate 1024 points of wideband acoustic conductance data using STRUCTURECAD for an apodized transducer whose minimum track model has 200 tracks. More complex transducers and more detailed analyses require more calculation time.

Calculation time may be reduced immediately by preparing a more efficient Fast Fourier Transform (FFT) routine [23]. Faster calculation speed may be obtained by storing values of the appropriate sines and cosine terms in order to avoid their repetitive calculation. Further gains in speed may be realized by allowing the FFT to use radices larger than 2. Finally, a very customized Discrete Fourier Transform (DFT) algorithm may be developed. This algorithm could allow the designer to choose the frequencies bands that interest him so as to save the time it takes to calculate the data for frequencies of less importance. The algorithm could be customized to the uniform track model by allowing time points to assume only the values of 0, 1 and -1. Multiplications could in this way be streamlined.

Calculation time may also be reduced by altering the track analysis technique. The minimum track model was shown to be more accurate than the amplitude weighting approximation in predicting acoustic conductance. It may be possible, however, to make fairly accurate predictions of acoustic conductance on the basis of a modified finite track analysis method as well. Tracks in the finite track model are often nonuniform. The accuracy of the finite track model could be improved by treating each track as an amplitude weighted transducer rather than as a uniform transducer. It is possible that simulations for acoustic conductance and apodization loss could very well be carried out to desired accuracies with this technique while using fewer tracks and therefore taking less time. Simulation accuracy could be further improved at the expense of time by modelling the effects of diffraction and triple transit on transducer acoustic conductance [24,25].

Another factor which limits the practical use of STRUCTURECAD is its memory requirements. Track response information is currently stored in variable arrays for easy access. Variable arrays are dimensioned to allow the response of each track to be resolved to the dimension of the FFT, which for the case of the analyses presented was 1024 points. Reducing either the number of frequencies evaluated or the number of tracks analyzed would automatically reduce the memory requirements of the system. However, it is not essential to store all the track information at all times in order to perform the analysis described thus far. The calculations now performed on the two dimensional variable matrices could be performed instead as running summations stored in one dimensional matrices, thereby saving over 99 percent of the memory currently required.

The total response of an apodized transducer SAW filter may be evaluated as the product of the two transducer responses only when the transducers are separated by a multistrip coupler [25]. One last extension of the work presented, however, would be to generalize the track modelling techniques in order to analyze the response of two apodized transducers in direct cascade. Apodized transducers are typically not cascaded directly because they do not radiate uniformly. The less popular geometries analyzed, however, were found to radiate more uniformly than the more common slant and beam centered overlap structures. It may be possible to approach the performance of two apodized transducers separated by a multistrip coupler by joining two apodized transducers in cascade.

The 4 transducer geometries studied provide 16 pairs of structures. Because some apodized transducers are found to radiate nearly uniformly at center frequency, the response of one or more combinations of directly cascaded apodized transducers may be found to approximate the product of the transducer responses for some filters.

APPENDIX A

LISTINGS FOR "STRUCTURECAD"

```
c
c   strcadmain -    STRUCTURE CAD MAIN MODULE
c
c
------------------------------------------------------------
c
c   Date Last Revision:  03-21-86
c
c
------------------------------------------------------------
c
c   This FORTRAN 77 source code file implements the main
menu for
c   STRUCTURE CAD, the Surface Acoustic Wave Transducer
Structure
c   Computer Aided Design and Analysis System.
c
c   Principal Designer:  A.J. Vigil,
c   Solid State Devices and Systems Group,
c   Department of Electrical Engineering and Communication
Sciences,
c   University of Central Florida, Orlando, Florida  32816
c
c-------------------------------------------------------
c
    real*8 tamp(1024), cbtrac(1000), xstart, factor,
slval23
c
    character*1  split, ppad, com, stype, stype23,split23
c
    character*10  basnaml, basnam2, basnam3
c
    character*3  extl, ext2, ext3
c
    character*14  filnaml, filnam2, filnam3
c
c
    common /pat51/ fol, tflol, tfhil, numl, tamp,
tphase(1024),
     2      filnaml, extl, basnaml
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2      num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
    common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
c
```

```
    common / nvalid / nvalid
c
c
c
    ntrigterm = 7
c
    ifilstatsw = -1
c
c
100 call erscrn
c
    write (6,*) ' '
    write (6,*) ' '
    write (6,*) '          <<< STRUCTURE CAD MAIN MENU
>>>'
    write (6,*) ' '
    write (6,127)
127 format(10x,'COMMAND  FUNCTION')
    write (6,*) ' '
    write (6,*) '  1.  PFR     Perform File Read'
    write (6,*) '  2.  SAF     Create Structure Analysis
File'
    write (6,*) '  3.  FTA     Perform Frequency Domain
Track Analysis'
    write (6,*) '  4.  ECP     Derive Equivalent Circuit
Parameters'
    write (6,*) '  5.  OFR     Derive Overall Frequency
Response'
    write (6,*) '  6.  TSA     Transducer Structure
Assembly'
    write (6,*) '  7.  GRP     Graphics'
    write (6,*) '  8.  CFS     Change File Status
Switch'
    write (6,*) '  9.  LSC     Leave Structure CAD'
    write (6,*) ' '
    if (ifilstatsw.gt.0) call filcursys
    write (6,1312)
1312        format('       COMMAND : ==> ',$)
    read  (5,1316)  com
1316        format(a1)
    call erscrn
c
c
c   write (6,1320) com,com,com
1320        format (//6x,a1,' 'a1,/,'   ',a1,//)
c
    if (com.eq.'1'.or.com.eq.'p'.or.com.eq.'P') call
readfiles
    if (com.eq.'2'.or.com.eq.'s'.or.com.eq.'S') call
timearray
```

```
    if (com.eq.'3'.or.com.eq.'f'.or.com.eq.'F') call
freqarray
    if (com.eq.'4'.or.com.eq.'e'.or.com.eq.'E') call
equivcirc
    if (com.eq.'5'.or.com.eq.'o'.or.com.eq.'O') go to 100
    if (com.eq.'6'.or.com.eq.'t'.or.com.eq.'T') call
transtruc
    if (com.eq.'7'.or.com.eq.'g'.or.com.eq.'G') call
showntell
    if (com.eq.'8'.or.com.eq.'c'.or.com.eq.'C') ifilstatsw
= -ifilstatsw
    if (com.ne.'9'.and.com.ne.'l'.and.com.ne.'L') goto 100
c
    stop
c
    end
c
c
c
c   filcursys -     FILES CURRENTLY IN SYSTEM
c
c
------------------------------------------------------------
c
c   Date Last Revision:  03-21-86
c
c
------------------------------------------------------------
c
c   This FORTRAN 77 source code file contains the software
to display
c   a table of files that STRUCTURE CAD is currently
processing.
c
c--------------------------------------------------------
c
    subroutine filcursys
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23
c
    character*1  split, ppad, com, stype, stype23,split23
c
    character*10  basnam1, basnam2, basnam3
c
    character*3  ext1, ext2, ext3
c
    character*14  filnam1, filnam2, filnam3
c
c
```

```
    common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(1024),
     2       filnam1, ext1, basnam1
c
c
    common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
c
    common /pat54/ stype4, slval4, nfft4, cap,
acond(2048), hxsus(2048),
     2       filnam4, basnam4, ext4, split4, fo4, tflo4,
tfhi4
c
    common /pat55/ stype5, slval5, fo5, tflo5, tfhi5,
nfft5, famp(2048),
     2       fphase(2048), filnam5, basnam5, ext5, split5
c
c
c
c
    write (6,1255)
1255         format(20x, 'FILES CURRENTLY IN SYSTEM')
    write (6,1257)
1257         format(10x,'File Type',30x,'Name')
    write (6,1260) filnam1
1260         format('  1.  Time Impulse Response
',a14)
    write (6,1264) filnam2
1264         format('  2.  Structure Time/Track Model
',a14)
    write (6,1268) filnam3
1268         format('  3.  Structure Frequency/Track Model
',a14)
    write (6,1272) filnam4
1272         format('  4.  Equivalent Circuit Parameters
',a14)
    write (6,1276) filnam5
1276         format('  5.  Overall Frequency Response
',a14)
    write (6,1280) filnam6
1280         format('  6.  Transducer Structure
',a14)
c
c
    return
c
c
    end
c
```

```
c
c   readfiles
c
c------------------------------------------------------------
c
c   Date Last Revision:  10-02-85
c
c------------------------------------------------------------
c
c   This FORTRAN 77 source code file reads files needed
for the execution
c   of STRUCTURE CAD as prescribed by the user.
c
c------------------------------------------------------------
c
    subroutine readfiles
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23
c
    integer*4 ihold1
c
    character*1  split, ppad, com, stype, stype23,split23
c
    character*10  basnam1, basnam2, basnam3,
c
    character*3  ext1, ext2, ext3
c
    character*14  filnam1, filnam2, filnam3
c
    character filin*14,filout*14
c
    character basin*10,extin*3
c
c
c
c
    common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(1024),
     2       filnam1, ext1, basnam1
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2       num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
    common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
    common /pat53/ tmamp(600,1024),tmphase(600,1024)
```

```
c
c
c
c
c
    common/names/ filin, filout
c
c
222 call erscrn
c
c
    write (6,*) '   <<< SYSTEM READ >>>'
    write (6,*) ' '
    call filcursys
    write (6,*) '  7.  Return to Main Menu'
333 write (6,*) ' '
    write (6,1005)
1005      format(x,'Enter input file type : ==> ',$)
    read  (5, 999, err=10) itypein
999 format(i1)
    if (itypein.eq.7) return
    if (itypein.lt.1.or.itypein.gt.6) goto 333
    write (6,*) ' '
100 write (6,1010)
1010      format(x,'ENTER input file name : ==> ',$)
    read  (5, 1000, err=10) filin
1000      format(a14)
c
c   separate extension from file name
c
    idotstat = 0
c
    do 2017 ipointvar = 2, 11, 1
c
          if (filin(ipointvar:ipointvar).eq.'.') then
                basin = filin(1:ipointvar-1)
                extin = filin(ipointvar+1:ipointvar+3)
                idotstat = 1
          endif
c
2017      continue
c
    if (idotstat.eq.1) goto 2019
c
    basin = filin
c
    write (6,2025)
2025      format(x,'ENTER extension : ==> ',$)
    read  (5,2035,err=10) extin
2035      format(a3)
```

```
c
2019        filin = basin//'.'//extin
c
    if(filin.eq.'          ') return
    open(10,file=filin,status='unknown',err=10)
    goto 20
10  write(6,*)' '
    write(6,*) '*** ERROR in file or filename ***'
    write(6,*) ' '
    goto 222
c
c   read in file data
c
20  rewind (10)
c
    read (10,1900,err=10,end=10) icont
1900        format(x,6x,i5)
c
c
    if ((icont.eq.10.or.icont.eq.1.or.icont.eq.11) .and.
itypein.eq.1) then
            filnam1 = filin
            basnam1 = basin
            ext1 = extin
            read(10,1902,err=10,end=10) itype
            if (itype.ne.-1) goto 1876
            read(10,1901,err=10,end=10) fo1
            read(10,1901,err=10,end=10) tflo1
            read(10,1901,err=10,end=10) tfhi1
            read(10,1902,err=10,end=10) num1
c
1901                format(x,6x,e17.9)
1902                format(x,6x,i4)
1903                format(x,6x,a1)
c
            do 300 i = 1, num1, 1
                    read (10,*,err=10,end=10)
tamp(i),tphase(i)
300         continue
c
    else if (itypein.eq.2.and.icont.eq.2) then
c
            filnam3 = ' '
            basnam3 = ' '
            ext3    = ' '
            filnam2 = filin
            basnam2 = basin
c
            read  (10,1903,err=10,end=10) stype23
            ext2 = stype23//'ta'
            read  (10,1903,err=10,end=10) split23
            read  (10,1901,err=10,end=10) slval23
            read  (10,1901,err=10,end=10) fo23
            read  (10,1901,err=10,end=10) tflo23
            read  (10,1901,err=10,end=10) tfhi23
            read  (10,1902,err=10,end=10) num23
            read  (10,1902,err=10,end=10) nfft23
            read  (10,1902,err=10,end=10) numtrac23
c
            do 755  i = 1, numtrac23, 1
                    read  (10,1901,err=10,end=10)
cbtrac(numtrac2)
755         continue
c
            do 766  i = 1, numtrac23, 1
                    do 760  j = 1, nfft23, 1
                        read  (10,*,err=10,end=10)
tmamp(i,j), tmphase(i,j)
760                 continue
766         continue
c
c
    else if (icont.eq.3.and.itypein.eq.3) then
c
            filnam2 = ' '
            ext2    = ' '
            basnam2 = ' '
            filnam3 = filin
            basnam3 = basin
c           read  (10,1903,err=10,end=10) stype23
            ext3 = stype23//'ta'
            read  (10,1903,err=10,end=10) split23
            read  (10,1901,err=10,end=10) slval23
            read  (10,1901,err=10,end=10) fo23
            read  (10,1901,err=10,end=10) tflo23
            read  (10,1901,err=10,end=10) tfhi23
            read  (10,1902,err=10,end=10) num23
            read  (10,1902,err=10,end=10) nfft23
            read  (10,1902,err=10,end=10) numtrac23
c
            do 855  i = 1, numtrac23, 1
                    read  (10,1901,err=10,end=10)
cbtrac(numtrac2)
855         continue
c
            do 866  i = 1, numtrac23, 1
                    do 860  j = 1, nfft23, 1
                        read  (10,*,err=10,end=10)
tmamp(i,j), tmphase(i,j)
860                 continue
```

```
866         continue
c
            do 890  i = 1, nfft3, 1
               read  (10,*,err=10,end=10)
sumamp(i),sumphase(i)
890         continue
c
   else
c
1877           write (6,777) filin, icont, itypein
777    format(x, a14, ' is a type ', i1, ' file, not a
type ',i1,'.')
c
   endif
c
   goto 1888
c
1876        write(6,*) 'This is not a time file. '
c
1888        close (unit=10)
c
   goto 222
c
   end
c
c
c  timearray -    CREATE STRUCTURE ANALYSIS FILE - TIME
c
c-----------------------------------------------------
c
c  Date Last Revision:     03-22-86
c
c-----------------------------------------------------
c
c  This FORTRAN 77 source code file will create a two
dimensional array
c  to model a transducer given its time impulse response
and type.
c  The array's dimensions are time and acoustic beamwidth
position.
c
c-----------------------------------------------------
c
   subroutine timearray
c
   real*8  tamp(1024), cbtrac(1000), xstart, factor,
slval23
c
   character*1
split,ppad,com,stype,stype23,split23,strucans,aans
```

```
c
      character*10  basnam1, basnam2, basnam3
c
      character*3  ext1, ext2, ext3
c
      character*14  filnam1, filnam2, filnam3
c
      common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(1024),
     2       filnam1, ext1, basnam1
c
      common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2       num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
      common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
      common /pat53/ tmamp(600,1024),tmphase(600,1024)
c
      common / nvalid / nvalid
c
      common /pat521/ isplitnum
c
      call erscrn
c
      write (6,*) '           CREATE STRUCTURE TIME ARRAY'
      write (6,*) ' '
      write (6,*) '   Please Specify Structure Type.'
      write (6,*) ' '
      write (6,*) '1. Slant apodization'
      write (6,*) '2. Meandering Ground Apodization'
      write (6,*) '3. Generalized Distance Weighting
Apodization'
      write (6,*) '4. Return to Main Menu'
      write (6,*) ' '
      write (6,1011)
1011         format(x,'   Structure Type : ==> ',$)
      read  (5,1013) strucans
1013         format(a1)
c
      iarraystat = 1
c
      if
(strucans.eq.'1'.or.strucans.eq.'s'.or.strucans.eq.'S')
then
             stype23 = 's'
             call pretarray
             call slatarray
```

```
    else if
(strucans.eq.'2'.or.strucans.eq.'m'.or.strucans.eq.'M')
then
            stype23 = 'm'
            call pretarray
            call mgatarray
    else if
(strucans.eq.'3'.or.strucans.eq.'g'.or.strucans.eq.'G')
then
            stype23 = 'g'
            call pretarray
            call gdwtarray
    else
            return
    endif
c
    nvalid = isplitnum*num23 + 2
c
889 noverlap = 0
c
900 write (6,979)
979 format( ' View time array? (Y/N) ',$)
    read  (5,1013,err=989) aans
    if (aans.eq.'y'.or.aans.eq.'Y') call disptimarry
c
    if (noverlap.eq.0) then
c
            call erscrn
            write (6,879)
879         format( ' Go to overlap array factor ? (Y/N)
',$)
            read  (5,1013,err=889) aans
c
            if (aans.eq.'y'.or.aans.eq.'Y') then
c
                    do  710  i = 1, numtrac23
c
                            do  705  j = 2, nvalid
c
                                 tmamp(i,j-1) = tmamp(i,j)
- tmamp(i,j-1)
c
705                         continue
c
710                 continue
c
                    noverlap = 1
                    nvalid = nvalid - 1
c
                    goto 900
```

```
c
            endif
c
    endif
c
989 return
c
    end
c
c
c   freqarray -    PERFORM FREQUENCY DOMAIN TRACK ANALYSIS
c
c-------------------------------------------------------
c
c   Date Last Revision:  03-22-86
c
c-------------------------------------------------------
c
c   This FORTRAN 77 source code file, given a time
transducer model array,
c   will perform a track by track Fast Fourier Transform,
resulting in
c   both amplitude and phase arrays whose dimensions are
frequency
c   and acoustic beamwidth position.  The amplitude array
is normalized
c   to the square root of the weighted (by track width)
sum of energy
c   in the acoustic beamwidth at center frequency.  Net
overall amplitude
c   and phase response is calculated for each frequency
point.
c
c-------------------------------------------------------
c
    subroutine freqarray
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23, rnorm,
     2      inorm, normden, normconst, rnfft23
c
    real*8 abw
c
    character*1
split,ppad,com,stype,stype23,split23,aans,bans
c
    character*10  basnam1, basnam2, basnam3
c
    character*3  ext1, ext2, ext3
c
```

```
      character*14  filnam1, filnam2, filnam3
c
      character*20  filout
c
      common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(1024),
     2       filnam1, ext1, basnam1
c
      common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2       num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
      common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
      common /pat53/ tmamp(600,1024),tmphase(600,1024)
c
      common /file/ amp(4096),phase(4096),nfft,itype
      common /dat/  fo,tflo,tfhi,num
c
      common /conduct/
acouscon(4096),rksq,vo,abw,cs,eta,splfcond,
     2                aploss
c
      common /original/ origamp(4096),origphase(4096)
c
      common / nvalid / nvalid
c
      common / noverlap / noverlap
c
      common /dpar/ ca,qo,rp,cp,gao,aneff
c
          dimension bc(10)
          complex cmplx,cexp,yc,zc,z1,s11
          character*1 cpause,splfcond,structype
c
      data r1 /50.0/
c
      rnorm = 0.0
      inorm = 0.0
c
      rnfft23 = float(nfft23)
c
      write (6,531)
531   format(x,'Display track by track Fast Fourrier
Transform Process?
     2        (Y/N) ==> ',$)
      read  (5,251,err=252) aans
251   format(a1)
```

```
    go to 243
252 aans = 'n'
c
243 rmaxamp = 0.0
c
    do 989  i = 1, 2048, 1
c
            acond(i) = 0.0
c
989 continue
c
    normconst = (float(nfft23))*fo23*(tfhi23-tflo23)
     2              /float(nfft23-1) + 1.0
    nfo = normconst
c
cccc        MATRIX TRANSFORM
c
    basnam3 = basnam2
    ext3 = ext2(1:1)//'fa'
    filnam3 = basnam3//'.'//ext3
c
    filnam2 = ' '
    basnam2 = ' '
    ext2 = ' '
c
    fo = fo23
    num = nfft23
    nfft = nfft23
c
    do 1999  i = 1, numtrac23, 1
c
            itype = -1
            tflo = tflo23
            tfhi = tfhi23
c
            do 2001  j = 1, nvalid, 1
c
                    amp(j) = tmamp(i,j)
                    phase(j) = tmphase(i,j)
c
2001                continue
c
    do  2003  j  =  nvalid + 1,  num,  1
c
            amp(j)  = 0.0
            phase(j) = 0.0
c
2003        continue
c
    if (aans.eq.'y'.or.aans.eq.'Y') then
```

```
   call sinit
   call gplot
   endif
c
         call erscrn
         write (6,2702) i, numtrac23
2702           format( ' Now transforming track
',i4,' of ',i4,'.')
cccc
         call xform
cccc
   if (aans.eq.'y'.or.aans.eq.'Y') then
   call sinit
   call gplot
   endif
c
         if (i.eq.1) then
c
               rnorm =
cbtrac(i)*amp(nfo)*cos(phase(nfo))
               inorm =
cbtrac(i)*amp(nfo)*sin(phase(nfo))
               normden = (amp(nfo))**2*cbtrac(i)
c
         else
c
               rnorm = rnorm + (cbtrac(i) -
cbtrac(i-1))
    2
      *amp(nfo)*cos(phase(nfo))
               inorm = inorm + (cbtrac(i) -
cbtrac(i-1))
    2
      *amp(nfo)*sin(phase(nfo))
               normden  = normden
    2                              + (cbtrac(i) -
cbtrac(i-1))
    3                              *(amp(nfo))**2
c
         endif
c
         do  2010  j = 1, num, 1
c
               tmamp(i,j) = amp(j)
               tmphase(i,j) = phase(j)
c
2010           continue
c
   if (amp(nfo).gt.rmaxamp)  rmaxamp = amp(nfo)
c
```

```
1999        continue
c
    aploss  =  (rnorm**2.0 + inorm**2.0) / normden
c
    tflo23 = tflo
    tfhi23 = tfhi
c
    normconst = normden**(-1.0/2.0)
c
    if (normconst.le.1e-12) then
c
    write (6,*) ' Normalization Constant is zero!!'
c
    else
c
    do 2020  j = 1, nfft23, 1
c
            do 2030  i = 1, numtrac23, 1
c
                    tmamp(i,j) =  normconst*tmamp(i,j)
c
                    if (i.eq.1) then
c
                            rnorm =
tmamp(i,j)*cos(tmphase(i,j))*cbtrac(i)
                            inorm =
tmamp(i,j)*sin(tmphase(i,j))*cbtrac(i)
                            acond(j) =
tmamp(i,j)**2.0*cbtrac(i)
c
                    else
c
                            rnorm = rnorm +
tmamp(i,j)*cos(tmphase(i,j))
     2                                         *(cbtrac(i) -
cbtrac(i-1))
                            inorm = inorm +
tmamp(i,j)*sin(tmphase(i,j))
     2                                         *(cbtrac(i) -
cbtrac(i-1))
                            acond(j) = acond(j) +
tmamp(i,j)**2.0
     2                                         *(cbtrac(i) -
cbtrac(i-1))
c
                    endif
c
2030                continue
c
            sumamp(j)  =  (rnorm**2.0 + inorm**2.0)**0.5
```

```
c
            if (inorm.ne.0.0.or.rnorm.ne.0.0)
     2                          sumphase(j) =
atan2(inorm,rnorm)
c
2020        continue
c
    endif
c
c
    do 3020  k = 1, nfft23, 1
c
c
    acouscon(k) = rnfft23*normden*acond(k)
c
    acond(k) = acouscon(k)
c
3020        continue
c
    gao  =  acouscon(nfo)
c
    write(6,349) nfo, acouscon(nfo), gao
349 format('  acouscon(',i4,') = ',f15.7,',  gao =
',f15.7,'.')
c
100     write(6,1005)
1005    format(x,'ENTER output file name [<ret> to abort]:
==> ',$)
        read(5,1000,err=10) filout
1000    format(a20)
c
1001        format(a1)
c
c   determine if file already exists
c
        open(10,file=filout,status='old',err=11)
c
    write(6,*)'  '
    write(6,*)'ERROR-WRITEO-FIALEX: file already exists'
    write(6,*)'   '
    write(6,*)' OPTIONS: (N)ew name, enter new name'
    write(6,*)'          (C)ontinue and over-write file'
    write(6,*)'          (<return>) to abort'
13  write(6,12)
12  format(x,' ==>',$)
    read(5,1001) answ
    if (answ.eq.' ') then
         return
    else if (answ.eq.'N'.or.answ.eq.'n') then
        call erscrn
        goto 100
    else if (answ.eq.'C'.or.answ.eq.'c') then
        close(10,status='delete')
    else
        write(6,*) '  '
        write(6,*) 'ERROR-WRITEO-INVALEN: invalid entry,
retry'
        write(6,*) '  '
        goto 13
    endif
c
11      open(10,file=filout,status='new',err=10)
        goto 20
10      write(6,*)' '
        write(6,*) ' *** ERROR in file or filename ***'
        write(6,*)' '
        goto 100
c
c       write out file data
c
20  write(10,2301) itype
2301        format(x,'itype=',i4)
    write(10,2303) fo
2303    format(x,'fo   =',e17.9)
        write(10,2002) tflo
2002    format(x,'tflo =',e17.9)
    write(10,2304) tfhi
2304        format(x,'tfhi =',e17.9)
    write(10,2305) num
2305        format(x,'num  =',i4)
c
c
    write (10,*) acouscon(1), gao
c
    write (10,*) acouscon(2), aploss
c
        do 300 i=3,nfft23,1
c
                write(10,*) acouscon(i), acouscon(i)
c
300     continue
c
        close (10)
c
    read  (5,293) i
293 format(i1)
c
    call erscrn
c
        return
```

```
c
      end
c
c
c   showntell -    GRAPHICS
c
c-----------------------------------------------------------------
c
c   Date Last Revision:  03-22-86
c
c-----------------------------------------------------------------
c
c   This FORTRAN 77 source code file contains the
subroutine which will
c   generate graphical displays as desired by the
STRUCTURE CAD user.
c
c-----------------------------------------------------------------
c
    subroutine showntell
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23
c
    character*1  split, ppad, com, stype, stype23,split23
c
    character*10  basnaml, basnam2, basnam3
c
    character*3  extl, ext2, ext3
c
    character*14  filnaml, filnam2, filnam3
c
    common /pat51/ fol, tflol, tfhil, numl, tamp,
tphase(1024),
     2      filnaml, extl, basnaml
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2      num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
    common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
    common /pat53/ tmamp(600,1024),tmphase(600,1024)
c
    common /file/ amp(4096),phase(4096),nfft,itype
c
    common /dat/ fo,tflo,tfhi,num
c
```

```
395 call erscrn
c
    call filcursys
c
    write (6,*) '  7.  Return to Main Menu'
    write (6,*) ' '
    write (6,400)
400 format(x,'Which file would you like to view?')
    write (6,420)
420 format(x,'Please specify number.  ===> ',$)
    read  (5,440) ians
440 format(i1)
c
    if (ians.eq.1) then
c
            fo = fo1
            tflo = tflo1
            tfhi = tfhi1
            num  = num1
            itype = -1
c
            do 460 i = 1, num, 1
c
                    amp(i) = tamp(i)
                    phase(i) = tphase(i)
c
460         continue
c
            call sinit
            call gplot
c
    else if (ians.eq.2) then
c
            call disptimarry
c
    else if (ians.eq.3) then
c
            call dispfreqary
c
    else if (ians.eq.7) then
c
            return
c
    endif
c
    go to 395
c
    end
c
c
```

```
c
c   slatarray.for   SLANT APODIZATION TIME ARRAY
c
c-------------------------------------------------------
c
c   Date Last Revision:  03-22-86
c
c-------------------------------------------------------
c
c   This FORTRAN 77 source code file contains the routine
which will
c   assemble a time/acoustic beamwidth coordinate array
file representing
c   a slant apodized transducer for analysis.
c
c-------------------------------------------------------
c
    subroutine slatarray
c
    integer*4 idouble
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23, cutoff
c
    character*1  split, ppad, com, stype, stype23,split23
c
    character*10  basnaml, basnam2, basnam3
c
    character*3  extl, ext2, ext3
c
    character*14  filnaml, filnam2, filnam3
c
    common /pat51/ fol, tflol, tfhil, numl, tamp,
tphase(1024),
     2       filnaml, extl, basnaml
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2      num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
    common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
    common /pat53/ tmamp(600,1024),tmphase(600,1024)
c
    common /pat521/ isplitnum
c
c   Slope Entry (Slope of Slant)
c
```

```
790 write (6,*) ' '
    write (6,*) 'Please enter Slope of Slant.'
    write (6,*) '   -1.0 <= Slope <= 1.0'
    write (6,800)
800 format(30x,' ==> ',$)
    read  (5,810,err=790) slval23
810 format(f9.7)
    if (slval23.gt.1.0.or.slval23.lt.-1.0) goto 790
c
c   Assemble Track Matrix
c
    cbtrac(1) = 1.0
    factor = slval23/(num23-1)
    xstart = 0.5*(1.0 - slval23)
    cbtrac(2) = xstart
    cbtrac(3) = 0.5*(1.0+slval23)
    numtrac23 = 3
c
    do 1962  i = 1, num23, 1
c
            cbtrac(numtrac23+1) = xstart + (i-1)*factor
     2                        +
0.5*tamp(i)*cos(tphase(i))/abs(cos(tphase(i)))
c
            numtrac23 = numtrac23 + 1
c
1962        continue
c
    call sortme(cbtrac,numtrac23)
c
c   Track Matrix Assembly
c
    do 140   i  =   1, numtrac23,  1
c
            if (cbtrac(i).le.xstart+1d-15) then
c                   datapt = 1.0
                    tmamp(i,1) = 1.0
            else
c                   datapt = 0.0
                    tmamp(i,1) = 0.0
            endif
c
                    tmphase(i,1) = 0.0
c
140         continue
c
    call erscrn
    write (6,*) 'Left Augmenting Elements Finished.'
    do 1979  j = 1, num23, 1
c
```

```
      cutoff = xstart + (j-1)*factor + 1d-15
   2              +
0.5*tamp(j)*cos(tphase(j))/abs(cos(tphase(j)))
c
      do 1980  i = 1, numtrac23, 1
c
          if (cbtrac(i).le.cutoff) then
              tmamp(i,(j-1)*isplitnum+2) =
1.0
              tmamp(i,j*isplitnum+1)   = 1.0
          else
              tmamp(i,(j-1)*isplitnum+2) =
0.0
              tmamp(i,j*isplitnum+1)   = 0.0
          endif
c
          tmphase(i,(j+2)*isplitnum+1) = 0
          tmphase(i,(j+3)*isplitnum)   = 0
c
1980      continue
c
  if (mod(j,10).eq.0) then
c
  call erscrn
  write (6,1798) j, num23
1798  format(' Time Point Number ',i4,' of ',i4,' is
finished.')
c
  endif
c
1979  continue
c
  call erscrn
  write (6,*) ' Now entering right augmenting elements.'
c
  cutoff = 5d-1*(1.0+slval23) + 1d-15
c
  do 1984  i = 1, numtrac23, 1
c
      if (cbtrac(i).le.cutoff) then
c         datapt = 1.0
          tmamp(i,num23*isplitnum+2) = 1.0
      else
c         datapt = 0.0
          tmamp(i,num23*isplitnum+2) = 0.0
      endif
c
          tmphase(i,num23*isplitnum+2) = 0.0
c
1984      continue
```

```
c
    call erscrn
c
    return
c
    end
c
c
c
c
c   mgatarray        MEANDERING GROUND APODIZATION TIME
ARRAY
c
c-----------------------------------------------------------
c
c   Date Last Revision:     03-22-86
c
c-----------------------------------------------------------
c
c   This FORTRAN 77 source code file contains the code
necessary to
c   assemble a time/acoustic beamwidth coordinate array
file which
c   represents a meandering ground apodized SAW transducer
for analysis.
c
c-----------------------------------------------------------
c
    subroutine mgatarray
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23
c
    character*1  split, ppad, com, stype, stype23,split23
c
    character*10  basnam1, basnam2, basnam3
c
    character*3  ext1, ext2, ext3
c
    character*14  filnam1, filnam2, filnam3
c
    common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(1024),
     2      filnam1, ext1, basnam1
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2      num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
```

```
      common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
      common /pat53/ tmamp(600,1024),tmphase(600,1024)
c

      common /pat521/ isplitnum
c
c     cbtrac array
c
      cbtrac(1) = 1.0
      numtrac23 = 1
      iafstat = 1
      tamp(num1+1) = 1.0
      tphase(num1+1) = 180
c
      do 1776  i = 1, num23, 1
c
                  if (tamp(i)*cos(tphase(i)).gt.0.0) then
c                             ACTIVE WEIGHTING
                              numtrac23 = numtrac23 + 1
c
                              if (iafstat.eq.1) then
c
                                          cbtrac(numtrac23) =
tamp(i)*cos(tphase(i))/
     2
            abs(cos(tphase(i)))
c
                              else
c
                                          cbtrac(numtrac23) = 1.0
     2                                                            -
tamp(i)*cos(tphase(i))/
     3
            abs(cos(tphase(i)))
c
                              endif
c
                              if (tamp(i+1)*cos(tphase(i+1)).ne.
     2
            abs(tamp(i+1)*cos(tphase(i+1))))
     3                                                      iafstat = - iafstat
c
                  endif
c
1776              continue
c
      call sortme(cbtrac,numtrac23)
c
```

```
    iafstat = 1
c
    j = 1
c
    do 1780  i = 1, numtrac23, 1
c
                    tmamp(i,j) = 0.0
                    tmphase(i,j) = 0.0
c
1780        continue
c
c   call disptimarry
c
    do 1788  j = 1, num23, 1
c
            if (tamp(j)*cos(tphase(j)).gt.0.0) then
c                   ACTIVE WEIGHTING
c
                    if (iafstat.eq.1) then
c
                            do 1792  i = 1, numtrac23, 1
c
                                    if
(cbtrac(i).le.abs(tamp(j))+1d-15)
     2                                                then
c

tmamp(i,(j-1)*isplitnum+2) = 1.0

tmamp(i,j*isplitnum+1)   = 1.0
                                    else

tmamp(i,(j-1)*isplitnum+2) = 0.0

tmamp(i,j*isplitnum+1)   = 0.0
                                    endif
c

        tmphase(i,(j-1)*isplitnum+2) = 0

        tmphase(i,j*isplitnum+1)   = 0
c
1792                                continue
c
                    else
c
                            do 1796  i = 1, numtrac23, 1
c
                                   if
(cbtrac(i).le.1.0-abs(tamp(j))+1d-15)
```

```
     2                                                          then
c
tmamp(i,(j-1)*isplitnum+2) = 0.0

tmamp(i,j*isplitnum+1)     = 0.0
                                          else

tmamp(i,(j-1)*isplitnum+2) = 1.0

tmamp(i,j*isplitnum+1)     = 1.0
                                          endif
c

tmphase(i,(j-1)*isplitnum+2) = 0

tmphase(i,j*isplitnum+1)     = 0
c
1796                                       continue
c
                    endif
c
                    if (tamp(j+1)*cos(tphase(j+1)).ne.
     2        abs(tamp(j+1)*cos(tphase(j+1))))
     3                                     iafstat = - iafstat
c
            else
c
                    do 1800 i = 1, numtrac23, 1
c
                            tmamp(i,(j-1)*isplitnum+2) =
0.0
                            tmamp(i,j*isplitnum+1)     = 0.0
                            tmphase(i,(j-1)*isplitnum+2) =
0

                            tmphase(i,j*isplitnum+1)     = 0
c
1800                        continue
c
            endif
c
1788        continue
c
    j = isplitnum*num23 + 2
c
            do 1808  i = 1, numtrac23, 1
c
                    tmamp(i,j) = 0.0
                    tmphase(i,j) = 0
```

```
1808              continue
c
    return
c
    end
c
c
c
c   gdwtarray.for   GENERALIZED DISTANCE WEIGHTING TIME
ARRAY
c
c---------------------------------------------------
c
c   Date Last Revision:     03-22-86
c
c---------------------------------------------------
c
c   This FORTRAN 77 source code file contains the
subroutine which creates
c   a time/acoustic beamwidth array file to model a
generalized distance
c   weighted SAW transducer.
c
c---------------------------------------------------
c
    subroutine gdwtarray
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23, cutoff
c
    character*1  split, ppad, com, stype, stype23,split23
c
    character*10  basnam1, basnam2, basnam3
c
    character*3  ext1, ext2, ext3
c
    character*14  filnam1, filnam2, filnam3
c
    common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(1024),
     2       filnam1, ext1, basnam1
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2       num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
    common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
```

```
      common /pat53/ tmamp(600,1024),tmphase(600,1024)
c
      common /pat521/ isplitnum
c
      iafstat = -1
      cbtrac(1) = 1.0
      numtrac23 = 1
c
      do 1812  i = 1, num23, 1
c
            if (abs(tamp(i)*cos(tphase(i))).le.1.0e-12)
then
              itphs = -1
            else
              itphs =
tamp(i)*cos(tphase(i))/abs(tamp(i)*cos(tphase(i)))
            endif
c
            if
(abs(tamp(i)).eq.1.0.and.abs(tamp(i+1)).ne.1.0) then
c
                    iafstat = -iafstat
c
            else
c
                    if (iafstat*itphs.eq.1) then
c
                            numtrac23 = numtrac23 + 1
c
                            if (itphs.eq.1) then
c
                                    cbtrac(numtrac23) =
abs(tamp(i))
c
                            else
c
                                    cbtrac(numtrac23) =
1.0 - abs(tamp(i))
c
                            endif
c
                    endif
c
            endif
c
1812        continue
c
      call sortme(cbtrac,numtrac23)
c
c     track matrix
```

```
c
      iafstat = -1
c
      j = 1
c
      do 1816  i = 1, numtrac23, 1
c
                    tmamp(i,j) = 1.0
                    tmphase(i,j) = 0
c
1816        continue
c
      do 1824  j = 1, num23, 1
c
            if (abs(tamp(j)*cos(tphase(j))).le.1.0e-12)
then
              itphs = -1
            else
              itphs =
tamp(j)*cos(tphase(j))/abs(tamp(j)*cos(tphase(j)))
            endif
c
            if
(abs(tamp(j)).eq.1.0.and.abs(tamp(j+1)).ne.1.0)
     2                          iafstat = - iafstat
c
            if (iafstat*itphs.eq.1) then
c
                    if (itphs.eq.1) then
c
                            cutoff = abs(tamp(j)) + 1d-09
c
                    else
c
                            cutoff = 1.0 - abs(tamp(j)) +
1d-09
c
                    endif
c
            else
c
                    if (itphs.eq.1) then
c
                            cutoff = 1.0
c
                    else
c
                            cutoff = 0.0
c
                    endif
```

```
c
            endif
c
            do 1828  i = 1, numtrac23, 1
c
                    if (cbtrac(i).le.cutoff) then
c
                            tmamp(i,(j-1)*isplitnum+2) =
1.0
                            tmamp(i,j*isplitnum+1)    = 1.0
                    else
                            tmamp(i,(j-1)*isplitnum+2) =
0.0
                            tmamp(i,j*isplitnum+1)    = 0.0
                    endif
c
                    tmphase(i,(j-1)*isplitnum+2) = 0
                    tmphase(i,j*isplitnum+1)     = 0
c
1828                continue
c
1824        continue
c
    j = isplitnum*num23 + 2
c
            do 1836 i = 1, numtrac23, 1
c
                    tmamp(i,j) = 0.0
                    tmphase(i,j) = 0
c
1836                continue
c
    return
c
c
    end
c
c
c
c   sortme.for      SORTING ROUTINE
c
c--------------------------------------------------------
c
c   Date Last Revision:     03-22-86
c
c--------------------------------------------------------
c
c   This FORTRAN 77 source code file contains a sorting
subroutine which
c   will put the elements of an array in numerical order
```

```
and eliminate
c     double occurrences of the same element.
c
c----------------------------------------------------------------------
c
      subroutine sortme(sortarray,isize)
c
      real*8 sortarray(4096),temphold
c
      integer*4 idouble
c
      do 1771  i = 1, isize, 1
c
                  if (sortarray(i).lt.-1d-15.or
        2                             .sortarray(i).gt.1.0+1d-15)
then
c
                              write (6,*) '         WARNING'
                              write (6,*) 'A tap has extended
beyond'
                              write (6,*) ' the acoustic beamwidth.'
595                           read  (5,559,err=595) itypein
559                           format(i1)
c
                  endif
c
                  if (sortarray(i).lt.1d-15.or
        2                             .sortarray(i).gt.1.0-1d-15) go
to 1771
c
                  idouble = sortarray(i)*10**9 + 0.5
                  sortarray(i) = idouble
                  sortarray(i) = (sortarray(i) +
0.00000001)/10**9
c
1771              continue
c
      do 1832  i = 2, isize, 1
c
                  do 1836  j = i, 2, -1
c
                              if (sortarray(j-1).gt.sortarray(j))
then
c
                                          temphold = sortarray(j-1)
                                          sortarray(j-1) = sortarray(j)
                                          sortarray(j) = temphold
c
                              endif
c
```

```
1836                continue
c
1832        continue
c
1671        newsize = 0
c
      do 1840  i = 1, isize-1, 1
c
            if (newsize.ne.0) go to 1840
c
            if (sortarray(i).ge.1.0-1d-15) then
c
                    newsize = i
            else if (sortarray(i).eq.sortarray(i+1).
     2                      or.sortarray(i).le.1d-15) then
c
1846                        do 1844  j = i, isize-1, 1
c
                            sortarray(j) = sortarray(j+1)
c
1844                        continue
c
            if (sortarray(i).eq.sortarray(i+1).
     2                      or.sortarray(i).le.1d-15) go
to 1846
c
            endif
c
1840        continue
c
      if (newsize.ne.0) isize = newsize
c
      write (6,1253,err=1261) isize
1253        format(' There are ',i3,' tracks.')
      go to 1721
1261        write (6,*) ' There are too many tracks.'
1721        write (6,*) ' Display track-bottom
coordinates? (Y/N)'
      read  (5,1391,err=1445) aans
1391        format(a1)
      if (aans.eq.'y'.or.aans.eq.'Y')  call
displaysort(sortarray,isize,i,j)
c
1445        return
c
      end
c
      subroutine displaysort(sortarray,isize,i,j)
c
      character*1 aans
```

```
    real*8 sortarray(4096)
c
    call erscrn
c
    write (6,1232) isize,i,j
1232         format(3x,'isize = ',i3,3x,'i = ',i3,3x,'j =
',i3)
c
    do 1111  l = 1, isize, 1
c
    write (6,2222) l, sortarray(l)
2222         format(5x,i3,5x,f26.22)
c
1111         continue
c
    write (6,*) ' "Return" to continue. '
c
3393         read (5,3333,err=3393) itypein
3333         format (i1)
    return
c
    end
c
c
c
c   pretarray.for   PRELIMINARY TIME ARRAY ASSEMBLY
COMMANDS
c
c-------------------------------------------------------
c
c   Date Last Revision:     10-02-85
c
c-------------------------------------------------------
c
c   This FORTRAN 77 source code file contains several
commands common to
c   the assembly of all transducer type time/acoustic
beamwidth coordinate
c   models.
c
c-------------------------------------------------------
c
    subroutine pretarray
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23
c
    character*1  split, ppad, com, stype, stype23,split23
c
    character*10  basnam1, basnam2, basnam3
```

```
c
    character*3  ext1, ext2, ext3
c
    character*14  filnam1, filnam2, filnam3
c
    common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(1024),
     2      filnam1, ext1, basnam1
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2      num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
    common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
    common /pat53/ tmamp(600,1024),tmphase(600,1024)
c
    common /pat521/ isplitnum
c
    common / nvalid / nvalid
c
    real testpower2
    integer testflag
c
    num23   = num1
    fo23    = fo1
    tflo23  = tflo1
    tfhi23  = tfhi1
    basnam2 = basnam1
    ext2    = stype23//'ta'
    filnam2 = basnam2//'.'//ext2
c
    filnam3 = ' '
    basnam3 = ' '
    ext3    = ' '
c
999 write (6,*) ' '
    write (6,998)
998 format(x,'Split or Unsplit Electrodes? ==> ',$)
    read  (5, 997, err=999) split23
997 format(a1)
c
    if (split23.eq.'s'.or.split23.eq.'S') then
            isplitnum = 2
    else if (split23.eq.'u'.or.split23.eq.'U') then
            isplitnum = 1
    else
            isplitnum = 0
    endif
    if (isplitnum.eq.0) goto 999
c
995 write (6,*) ' '
    write (6,980) num23, filnam1
980 format(x,'There are ',i3,' samples in ',a14,'.  Please
enter')
    write (6,970) filnam2
970 format(x,'number of time samples per track desired for
',a14,'.')
    write (6,960) isplitnum*num23 + 2
960 format(x,'Number must be a power of 2 and greater than
',i3,'.')
    write (6,950)
950 format(x,'Enter : ==>',$)
    read  (5,940,err=995) nfft23
940 format(i4)
c
    if (nfft23.lt.isplitnum*(num23+6)) goto 995
c
    testpower2 = nfft23
    testflag   = 0
c
    do 930  i = 1, 14, 1
c
            testpower2 = testpower2/2.0
c
            if (testpower2.eq.1.0) testflag = 1
c
930 continue
c
    if (testflag.ne.1) goto 995
c
    tfhi23 = tflo23 +
abs(tfhi23-tflo23)*(nfft23-1)/((num23-1)*isplitnum)
c
c
    return
c
c
    end
c
c
c
c   disptimarry.for          DISPLAY TIME ARRAY
c
c-----------------------------------------------------
c
c   Date Last Revision:     03-22-86
c
```

```
c-----------------------------------------------------------
c
c   This FORTRAN 77 source code file contains the
subroutine used
c   to display the data contained in the system time array
file.
c
c-----------------------------------------------------------
c
    subroutine disptimarry
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23
c
    character*1  split,ppad,com,stype,stype23,split23,aans
c
    character*10  basnam1, basnam2,
basnam3,aslantstat,antracks
c
    character*11 chstr
c
    real*8 chnum
c
    character*3  ext1, ext2, ext3
c
    character*14  filnam1,filnam2,filnam3,apodstyle
c
    character*5 tilt,apage
c
    character*8 aorig,atilo,atihi
c
    character*12 asplitstat
c
    character*13 atimepts
c
    character*18 acenter
c
    character*32 atimspan
c
    character*72 datastring,blank,data2,data3
c
    common /chconv/ chstr,chnum
c
    common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(1024),
     2      filnam1, ext1, basnam1
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2      num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
      common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
c
      common /pat53/ tmamp(600,1024),tmphase(600,1024)
c
      common /file/ amp(4096),phase(4096),nfft,itype
c
      common /dat/  fo,tflo,tfhi,num
c
      common /pat521/ isplitnum
c
      common / nvalid / nvalid
c
      blank = ' '
c
      if (basnam2.eq.' ') then
              write (6,*) ' '
              write (6,*) 'A time file does not exist in the
system.'
              write (6,*) ' '
              return
      endif
c
c     apodization style
c
      if (stype23.eq.'s'.or.stype23.eq.'S') then
              apodstyle = 'SLANT OVERLAP'
              chnum = 1.000001*slval23
              intval = 0
              istrlen = 5
              call chconv(intval,istrlen)
              tilt = chstr
      else if (stype23.eq.'m'.or.stype23.eq.'M') then
              apodstyle = 'MEAND GND APOD'
              tilt = ' N/A '
      else if (stype23.eq.'g'.or.stype23.eq.'G') then
              apodstyle = 'GEN DIST WEIGH'
              tilt = ' N/A '
      else if (stype23.eq.'u'.or.stype23.eq.'U') then
              apodstyle = 'UNILAT OVERLAP'
              tilt = ' N/A '
      else
              write (6,*) ' '
              write (6,*) 'Incompatible Structure Type.'
              write (6,*) ' '
924           read  (5,950,err=924) itypein
950           format(i1)
              return
      endif
c
      aslantstat = 'SL = '//tilt
c
      if (split23.eq.'s'.or.split23.eq.'S') then
              asplitstat = ' SPLIT TAPS '
      else if (split23.eq.'u'.or.split23.eq.'U') then
              asplitstat = 'UNSPLIT TAPS'
      else
              write (6,*) ' '
              write (6,*) 'Incompatible Split Status'
              write (6,*) ' '
934           read  (5,950,err=934) itypein
      endif
c
      chnum = fo23
      intval = 0
      istrlen = 7
      call chconv(intval,istrlen)
      acenter = 'CENTER '//chstr(1:7)//' MHz'
c
      chnum = nfft23
      intval = 1
      istrlen = 4
      call chconv(intval,istrlen)
      atimepts = chstr(1:4)//' TIME PTS'
c
      chnum = num23
      intval = 1
      istrlen = 3
      call chconv(intval,istrlen)
      aorig = chstr(1:3)//' ORIG'
c
      chnum = numtrac23
      intval = 1
      istrlen = 3
      call chconv(intval,istrlen)
      antracks = chstr(1:3)//' TRACKS'
c
      intval = 0
      istrlen = 8
      chnum = tflo23
      call chconv(intval,istrlen)
      atilo = chstr
      chnum = tfhi23
      call chconv(intval,istrlen)
      atihi = chstr
      atimspan = atilo//' TO '//atihi//' MCRSEC TIME'
c
      ipagenum = 0
```

```
c
    do 801  ifirsttime = 1, nvalid, 72
c
            lasttime = ifirsttime + 71
c           if (lasttime.gt.(num23+10)*isplitnum)
            if (lasttime.gt.nvalid)
c    2                      lasttime =
(num23+10)*isplitnum
     2                      lasttime = nvalid
c
            do 802  ifirsttrack = 1, numtrac23, 20
c
                    lasttrack = ifirsttrack + 19
                    if (lasttrack.gt.numtrac23) lasttrack
= numtrac23
c
    call erscrn
    ipagenum = ipagenum + 1
    chnum = ipagenum
    intval = 1
    istrlen = 2
    call chconv(intval,istrlen)
    apage = 'PG '//chstr
    write
(6,1901)filnam2,apodstyle,asplitstat,aslantstat,acenter
1901        format(3x,a14,2x,a14,2x,a12,2x,a10,2x,a18)
    write (6,1902)atimepts,aorig,antracks,atimspan,apage
1902        format(3x,a13,2x,a8,2x,a10,2x,a32,2x,a5)
c
    write (6,*) ' '
    write (6,1903)
1903        format(3x,'BOT',58x,'TIME COORDINATE')
c
                    do 1951  i = 1, 1 + lasttime -
ifirsttime, 1
c
                            chnum = i
                            intval = 1
                            istrlen = 3
                            call chconv(intval,istrlen)
                  if (i.eq.1) then
                    datastring = chstr(1:1)
                    data2 = chstr(2:2)
                    data3 = chstr(3:3)
                  else
                    datastring =
datastring(1:i-1)//chstr(1:1)//blank
                    data2 =
data2(1:i-1)//chstr(2:2)//blank
                    data3 =
data3(1:i-1)//chstr(3:3)//blank
                  endif
c
1951                        continue
c
                    write (6,1904) datastring
1904                        format(2x,'TRACK',x,a72)
                    write (6,1905) data2
1905                        format(2x,'COORD',x,a72)
                    write (6,1906) data3
1906                        format(9x,a72)
                    write (6,*) ' '
c
                    intval = 1
                    istrlen = 1
c
                    do 1969  i = ifirsttrack, lasttrack, 1
c
                            if
(tmamp(i,ifirsttime).eq.0.0) then
                                    chstr = '0'//blank
                            else if
(tmamp(i,ifirsttime).eq.1.0) then
                                    chstr = '+'//blank
                            else if
(tmamp(i,ifirsttime).eq.-1.0) then
                                    chstr = '-'//blank
                            else
                                    chstr = '2'//blank
                            endif
c
                            datastring = chstr(1:1)//blank
c
                            do 1973  j = 1, lasttime -
ifirsttime, 1
c
                               if
(tmamp(i,ifirsttime+j).eq.0.0) then
                                            chstr = '0'
                               else if
(tmamp(i,ifirsttime+j).eq.1.0) then
                                            chstr = '+'
                               else if
(tmamp(i,ifirsttime+j).eq.-1.0) then
                                            chstr = '-'
                               else
                                            chstr = '2'
                               endif
c
                            datastring =
datastring(1:j)//chstr(1:1)//blank
c
1973                                continue
c
                    write (6,1907) cbtrac(i),datastring
1907                        format(x,f6.4,x,a72)
c
1969                        continue
c
972                 read  (5,950,err=972) itypein
c
802         continue
c
801 continue
c
895 call erscrn
    write (6,*) ' View a track? (Y/N) '
884 read  (5,874,err=884) aans
874 format (a1)
    if (aans.eq.'y'.or.aans.eq.'Y') then
c
856         write (6,857) numtrac23
857         format(x,'Enter Track Number ( 1 < n < ',i3,'
)')
            write (6,*) ' or "c" to continue :==>'
            read  (5,864,err=854) itracknum
864         format(i3)
            if (itracknum.gt.numtrac23.or.itracknum.lt.1)
go to 856
            fo = fo23
            tflo = tflo23
            tfhi = tfhi23
            num = nfft23
            itype = -1
c
            do 844  j = 1, nfft23, 1
c
                    amp(j) = tmamp(itracknum,j)
                    phase(j) = tmphase(itracknum,j)
c
844         continue
c
            call sinit
            call gplot
c
            go to 895
c
854 endif
c
795 call erscrn
```

```
    write (6,*) ' View a section of time beam profile?
(Y/N)'
784 read  (5,874,err=784) aans
    if (aans.eq.'y'.or.aans.eq.'Y') then
c
            write (6,713) nfft23
713         format(x,'Enter time column number for section
desired
     2                ( 1 < t < ',i4,' )')
            write (6,*) ' or "c" to continue :==>'
            read  (5,764,err=754) itimenum
764         format(i4)
            fo = fo23
            tflo = 0
            tfhi = 1
767         write (6,*) ' Please enter desired graph
resolution,'
            write (6,*) ' an integer less than 4096.'
            read  (5,764,err=767) num
            if (num.gt.4096.or.num.lt.1) go to 767
c
            itype = -2
c
            do 744  j = 1, num, 1
c
                    cutoff = float(j)/float(num)
c
                    iswitch = 0
c
                    do 749  i = 1, numtrac23, 1
c
                            if
(cbtrac(i).ge.cutoff.and.iswitch.eq.0) then
c
                                    amp(j) =
tmamp(i,itimenum)
                                    phase(j) =
tmphase(i,itimenum)
                                    iswitch = 1
c
                            endif
c
749                 continue
c
744         continue
c
            call sinit
            call gplot
c
            go to 795
```

```
c
754 endif
c
    return
c
    end
c
c
c
c   dispfreqary.for          DISPLAY FREQUENCY ARRAY
c
c-----------------------------------------------------------
c
c   Date Last Revision:      10-03-85
c
c-----------------------------------------------------------
c
c   This FORTRAN 77 source code file contains the
subroutine which makes
c   any structurecad transducer frequency array model
visually accessible
c   to the user.
c
c-----------------------------------------------------------
c
    subroutine dispfreqary
c
    real*8 tamp(1024),cbtrac(1000), xstart, factor,
slval23, rnorm,
     2      inorm, normden, normconst
c
    character*1  split,ppad,com,stype,stype23,split23,aans
c
    character*10  basnam1, basnam2, basnam3
c
    character*3  ext1, ext2, ext3
c
    character*14  filnam1, filnam2, filnam3
c
    common /pat51/ fo1, tflo1, tfhi1, num1, tamp,
tphase(4096),
     2      filnam1, ext1, basnam1
c
    common /pat52/
stype23,split23,slval23,fo23,tflo23,tfhi23,
     2      num23,nfft23,numtrac23,cbtrac, sumamp(2048),
sumphase(2048)
c
    common /pat525/
filnam2,basnam2,ext2,filnam3,basnam3,ext3
```

```
c
      common /pat53/ tmamp(600,1024),tmphase(600,1024)
c
      common /file/ amp(4096),phase(4096),nfft,itype
      common /dat/  fo,tflo,tfhi,num
c
      write (6,*) 'Return for sum of track responses
display'
235 read  (5,225,err=235) itypein
225 format(i1)
c
      fo = fo23
      tflo = tflo23
      tfhi = tfhi23
      num = nfft23
      nfft = nfft23
      itype = 1
c
      do 245  i = 1, nfft23, 1
              amp(i) = sumamp(i)
              phase(i) = sumphase(i)
245 continue
c
      call sinit
      call gplot
c
      call erscrn
613 write (6,*) ' Return for sum of squares of amplitudes
display'
      read  (5,225,err=613) itypein
      do 644  i = 1, nfft23, 1
      amp(i) = acond(i)
      phase(i) = 0.0
644 continue
      call sinit
      call gplot
c
895 call erscrn
      write (6,*) ' View a track? (Y/N) '
884 read  (5,874,err=884) aans
874 format (a1)
      if (aans.eq.'y'.or.aans.eq.'Y') then
c
856           write (6,857) numtrac23
857           format(x,'Enter Track Number ( 1 < n < ',i3,'
)')
              write (6,*) ' or "c" to continue :==>'
              read  (5,864,err=854) itracknum
864           format(i3)
              if (itracknum.gt.numtrac23.or.itracknum.lt.1)
```

```
go to 856
          fo = fo23
          tflo = tflo23
          tfhi = tfhi23
          num = nfft23
          itype = 1
c
          do 844  j = 1, nfft23, 1
c
                amp(j) = tmamp(itracknum,j)
                phase(j) = tmphase(itracknum,j)
c
844       continue
c
          call sinit
          call gplot
c
          go to 895
c
854 endif
c
795 call erscrn
   write (6,*) ' View beam profile for a cw input ?
(Y/N)'
784 read  (5,874,err=784) aans
   if (aans.eq.'y'.or.aans.eq.'Y') then
c
          write (6,713) tfhi23
713       format(x,'Enter approximate frequency desired
    2                ( 0 < t < ',f9.4,' )')
          write (6,*) ' or "c" to continue :==>'
          read  (5,764,err=754) aprxfreq
764       format(f9.4)
          ifreqnum = aprxfreq*(nfft23-1)/tfhi23
          fo = fo23
          tflo = 0
          tfhi = 1
767       write (6,*) ' Please enter desired graph
resolution,'
          write (6,*) ' an integer less than 4096.'
          read  (5,723,err=767) num
          if (num.gt.4096.or.num.lt.1) go to 767
c
723       format(i4)
c
          itype = 2
c
          do 744  j = 1, num, 1
c
                cutoff = float(j)/float(num)
```

```
c
                    iswitch = 0
c
                    do 749  i = 1, numtrac23, 1
c
                            if
(cbtrac(i).ge.cutoff.and.iswitch.eq.0) then
c
                                    amp(j) =
tmamp(i,ifreqnum)
                                    phase(j) =
tmphase(i,ifreqnum)
                                    iswitch = 1
c
                            endif
c
749                 continue
c
744         continue
c
            call sinit
            call gplot
c
            go to 795
c
754 endif
c
    write (6,11) num, nfft
11  format (' num = ',i4,', nfft = ',i4,'.')
c
    num = nfft23
    nfft = nfft23
c
    write (6,11) num, nfft
c
    return
c
    end
c
c
c
```

APPENDIX B

LISTING FOR "CALCOND"

```
c
c   CALCOND.FOR     "CALculate CONDuctance"
c
c ---------------------------------------------------------
c
c   Date Last Revision:     3-16-86
c
c ---------------------------------------------------------
c
c   Written by A.J. Vigil
c
c ---------------------------------------------------------
c
        subroutine  calcond(gao)
c
        real dummy
c
        character filin*20,txtdat*6,cpause*1
c
        common /file/ amp(4096), phase(4096), nfft, itype
c
        common /dat/  fo, tflo, tfhi, num
c
        common /conduct/ acouscon(4096), rksq, vo, bw, cs,
eta, splfcond,
     2              aploss, rho
c
        racouscoef = 2.0*rksq*cs*bw*vo
c
                call erscrn
c
c       terminal input
c
        write(6,*) '  <*****************>'
        write(6,*) '  <   SYSTEM READ   >'
        write(6,*) '  <*****************>'
        write(6,*)'    '
100     write(6,1010)
1010    format(x,'ENTER Acoustic Conductance input file
name : ==> ',$)
        read(5,1000,err=10) filin
1000    format(a20)
        if(filin.eq.'          ') return
        open(10,file=filin,status='old',err=10)
        goto 20
10      write(6,*)' '
        write(6,*) 'ERROR-READIN-FINOFO: file not found'
        write(6,*) ' '
        goto 100
c
```

```
c       read in file data
c
20      rewind (10)
c
c   test for file format, if the file comes from
c   the structure programs it will have "icont= XX"
c   as its first entry.  If so ignore this record
c
        read(10,1991,err=10,end=10) txtdat
1991    format(x,a6)
c
        if (txtdat.eq.'icont='.or.txtdat.eq.'ICONT=') then
            continue ! this read sets pointer to itype
        else
            rewind(10) ! regular file, reset and go
        endif
c
        read(10,1902,err=10,end=10) idum
        read(10,1901,err=10,end=10) dummy
        read(10,1901,err=10,end=10) dummy
        read(10,1901,err=10,end=10) dummy
        read(10,1902,err=10,end=10) idum
c
1901    format(x,6x,e17.9)
1902    format(x,6x,i4)
c
        read (10,*,err=10,end=10) acouscon(1), gao
c
        read (10,*,err=10,end=10) acouscon(2), aploss
c
123     write(6,1005) filin
        write(6,1003) gao,aploss
1005    format(x,///,x,'File Parameters for ',a10,' :')
1003    format(x,/,x,'Acoustic Conductance file -
Gao=',f11.6,
     2          /,'         APLOSS=',f11.6)
1006    format(x,///)
c
        do 300 i=3,nfft
        read(10,*,err=10,end=10) acouscon(i), dummy
        acouscon(i) = racouscoef*acouscon(i)
300     continue
c
        close (unit=10)
c
        gao = racouscoef*gao
c
        write(6,1003) gao,aploss
        write(6,*) ' racouscoef = ',racouscoef
c
```

```
        read (5,295) iins
295     format(il)
c
        return
        end
c
```

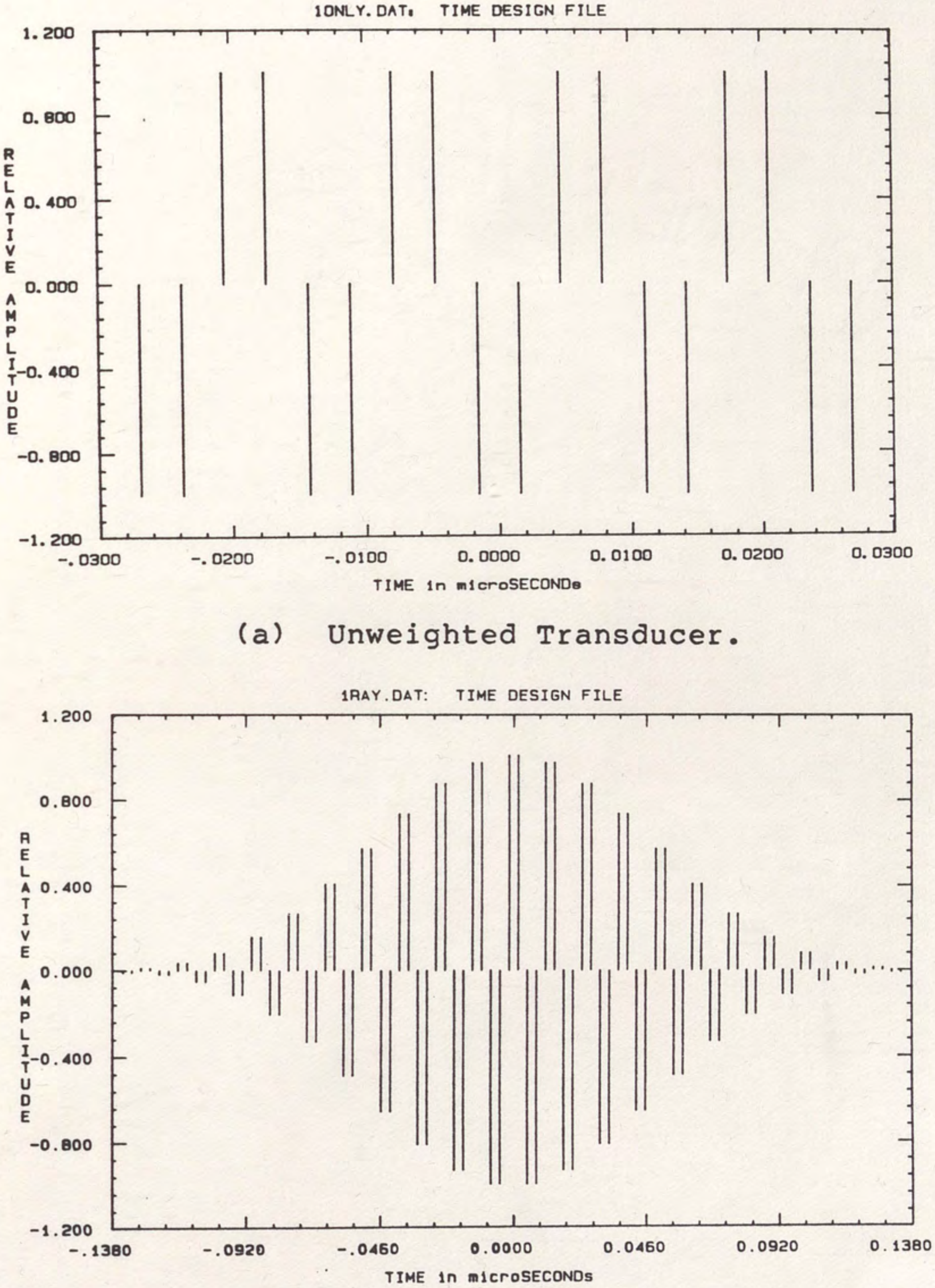
APPENDIX C

EIGEN WINDOW FILTER DATA

TABLE IV

EIGEN WINDOW FILTER PARAMETERS

Coupling Coefficient	.056
Acoustic Velocity	3927 meters/sec.
Acoustic Beamwidth	50 λ_o
Finger Capacitance	5.4 pF./cm.
Metal Resistivity	0.2 Ohms/sq.
Center Frequency	79 MHz.



(a) Unweighted Transducer.

(b) Eigen Window Transducer.

Figure 31. Eigen Window Filter Time Design Files.

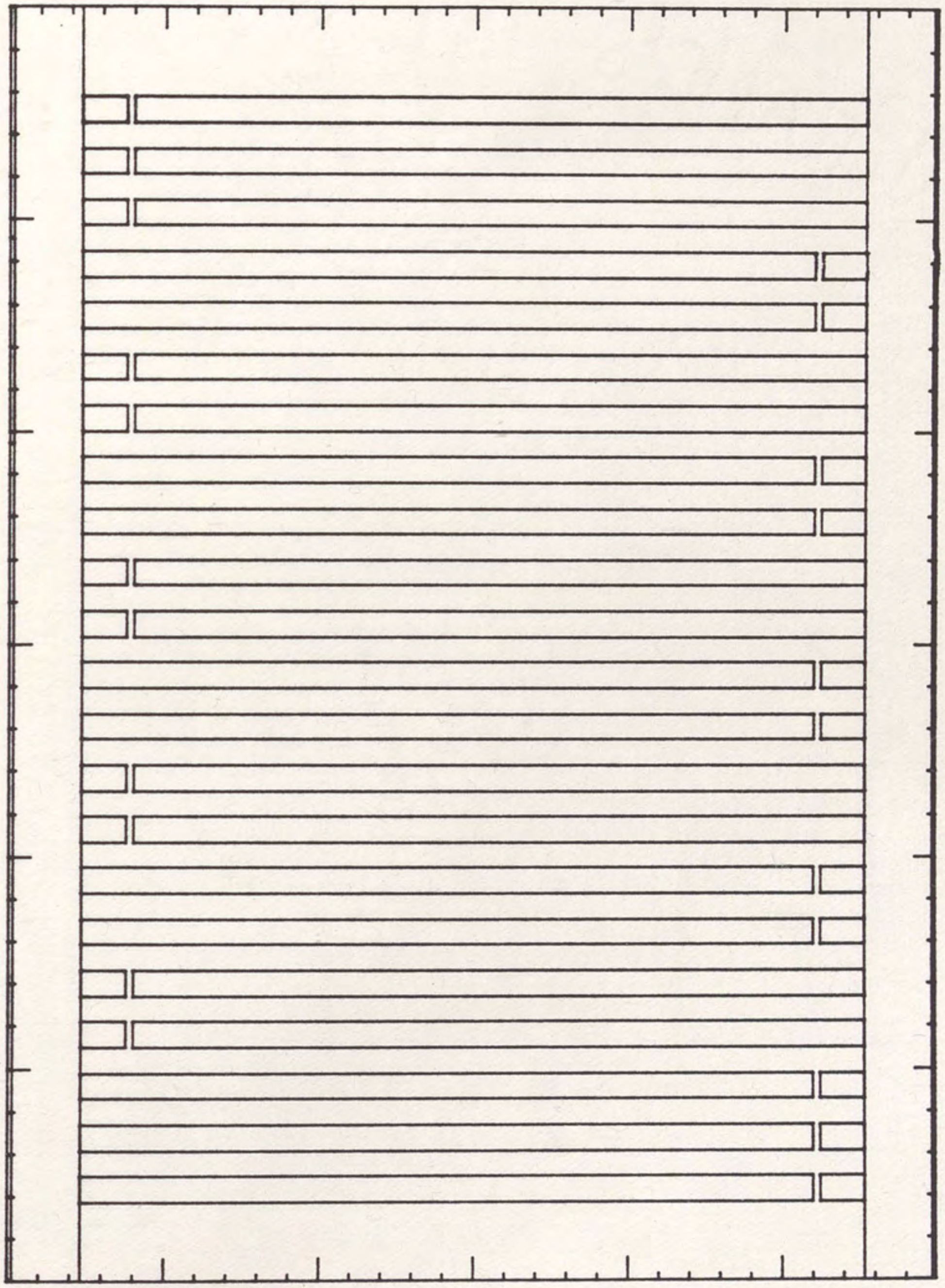


Figure 32. Eigen Window Filter Unweighted Transducer Structure.

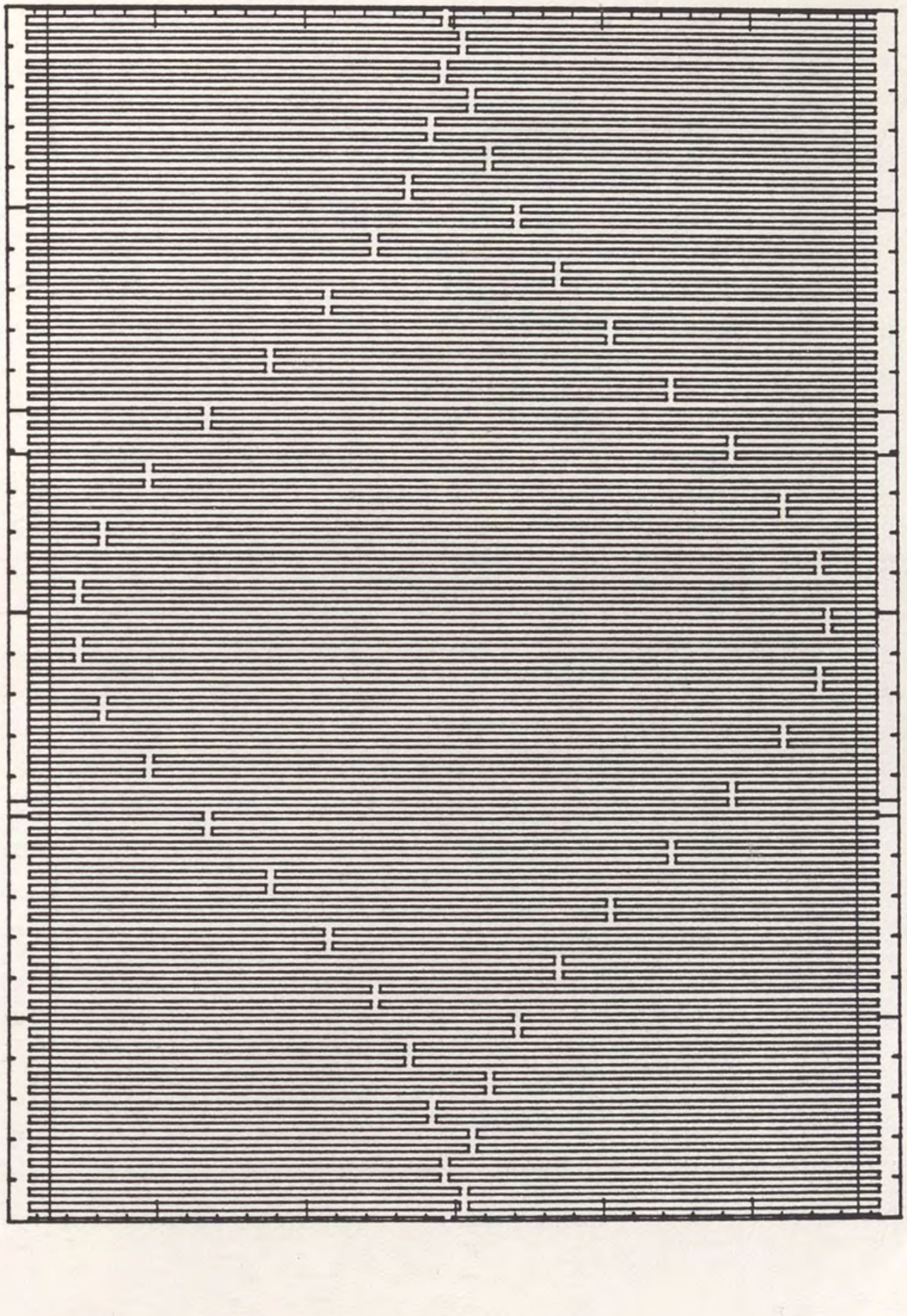


Figure 33. Eigen Window Transducer Beam Centered Overlap Structure.

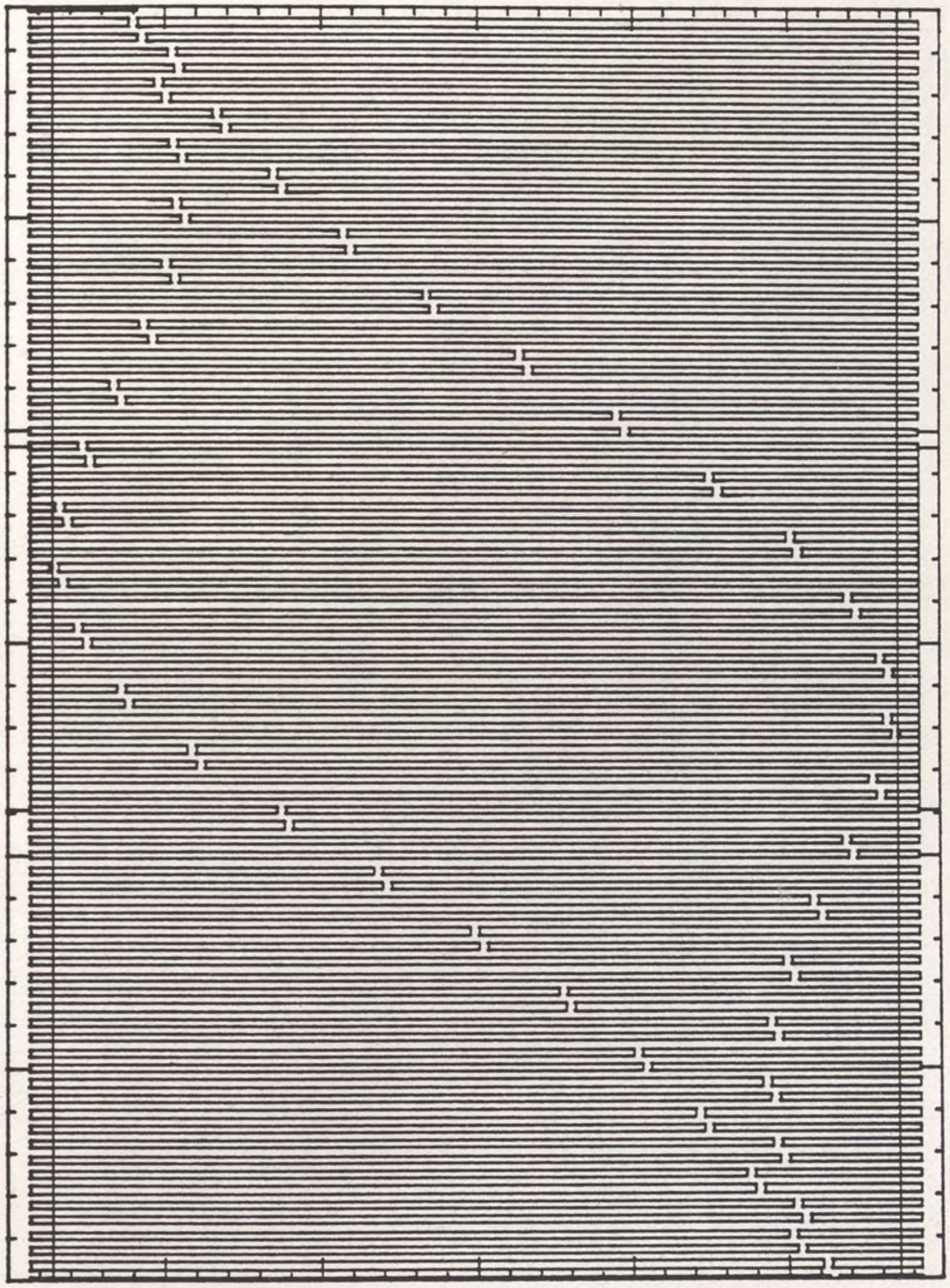


Figure 34. Eigen Window Transducer Slant Centered Overlap Structure.

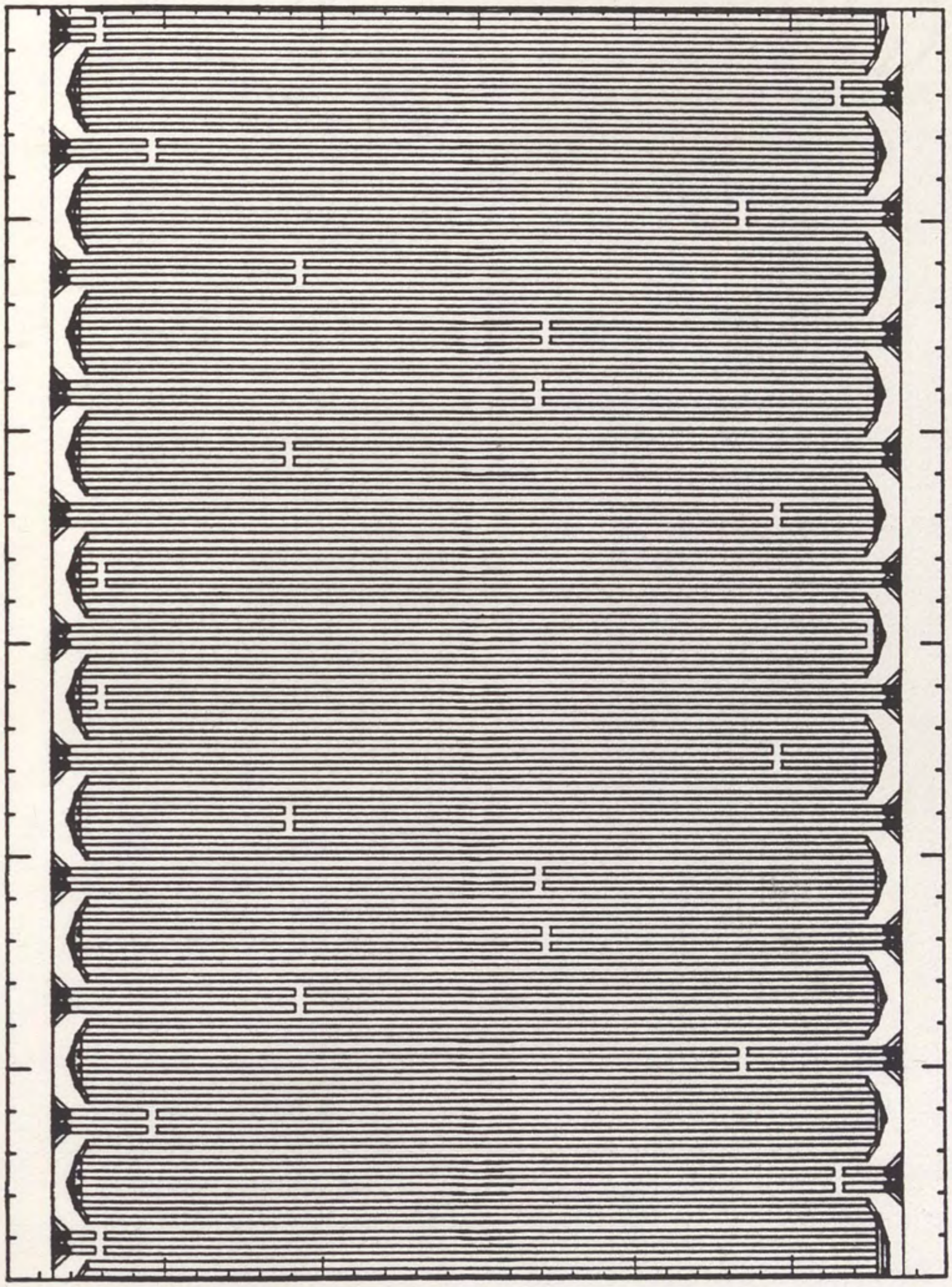


Figure 35. Eigen Window Transducer Beam Spread Overlap Structure.

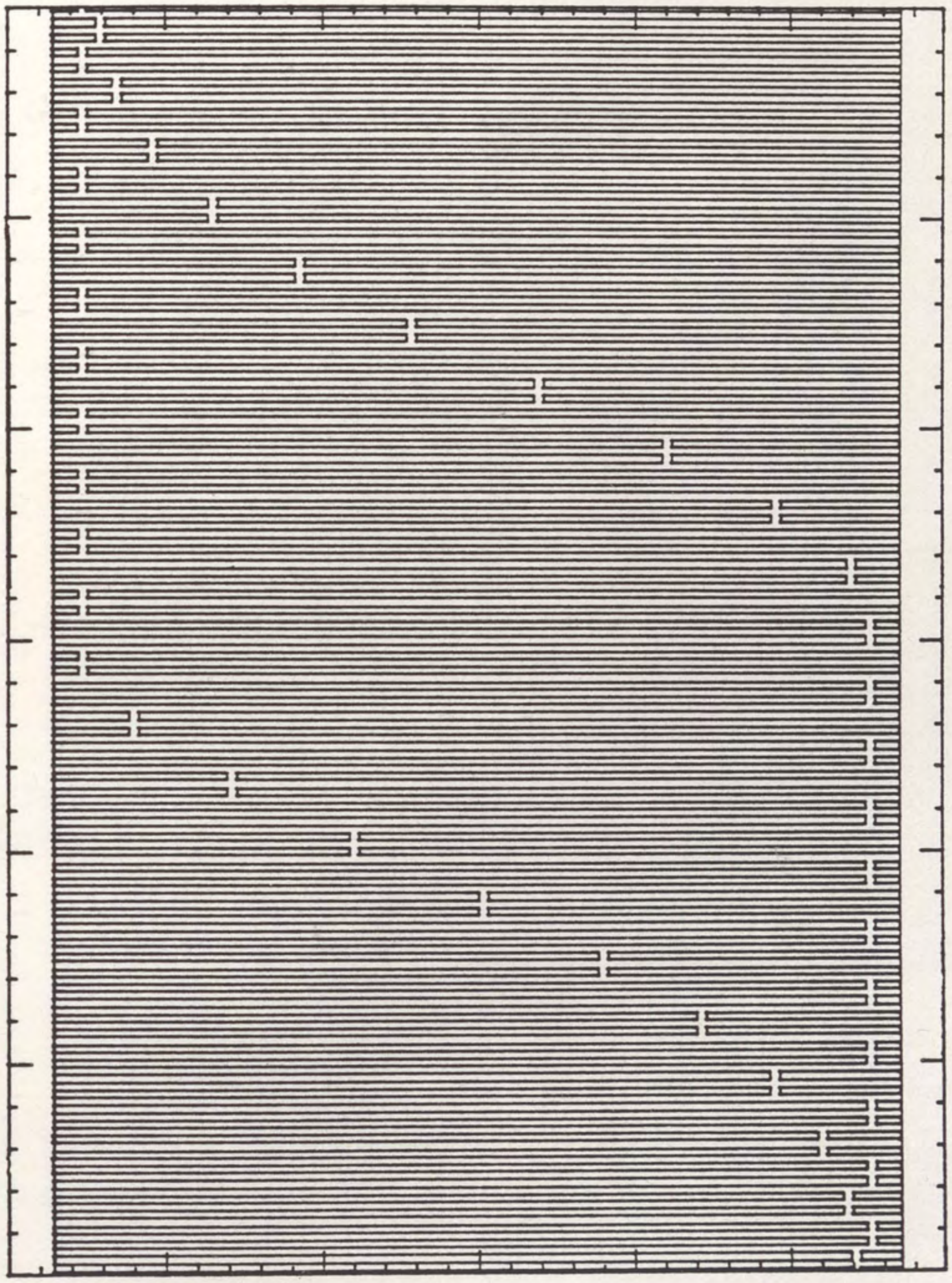
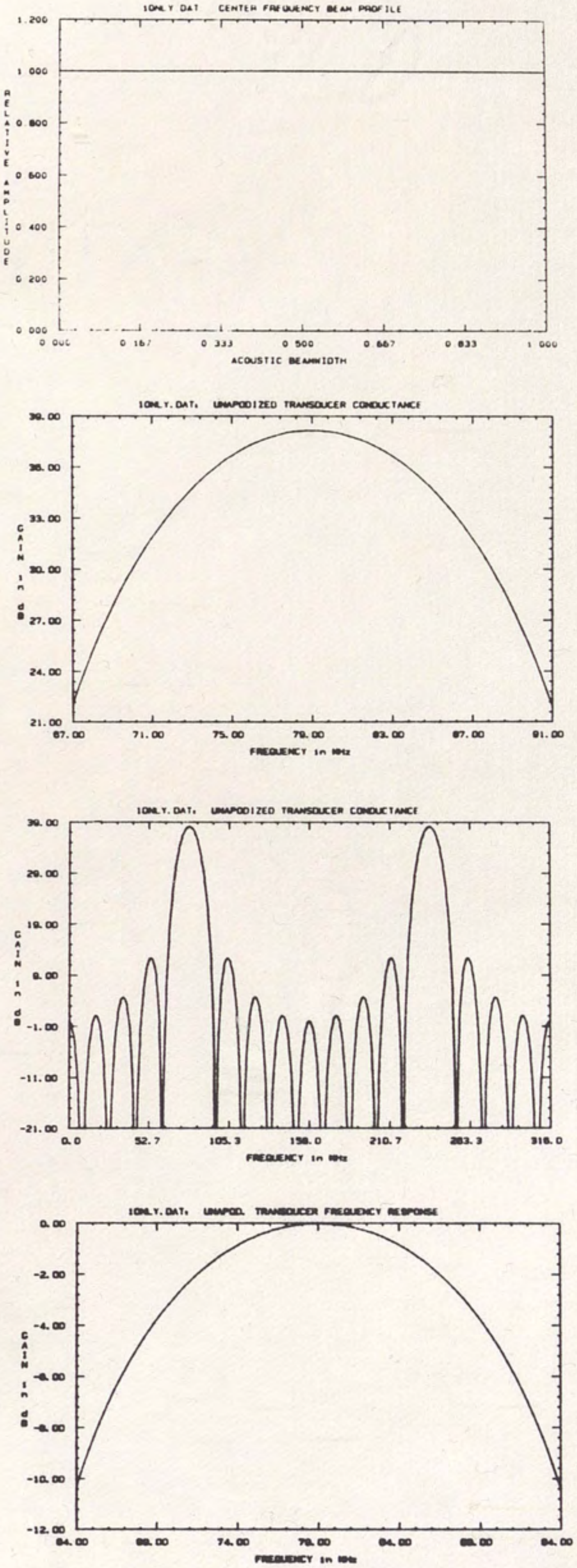


Figure 36. Eigen Window Transducer Generalized Distance Weighted Structure.



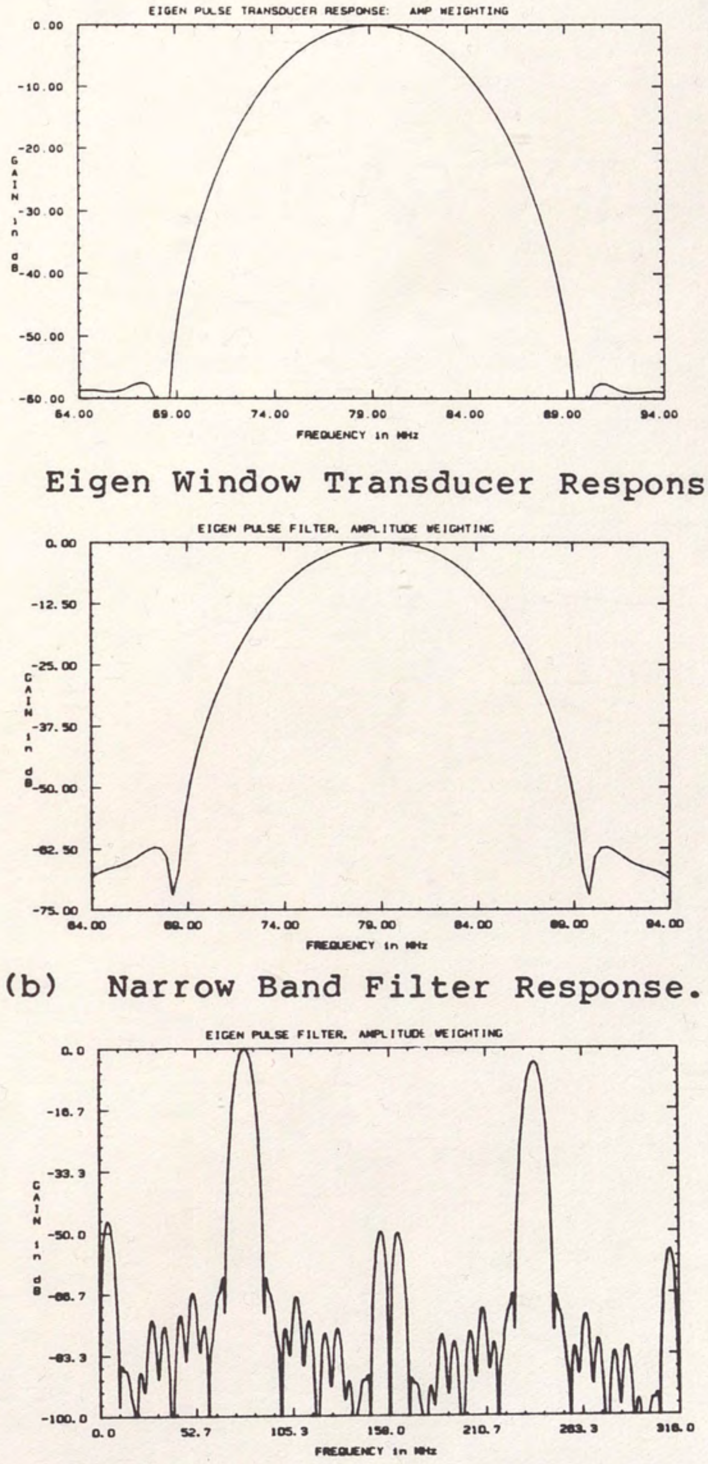
(a) Center Frequency Radiation Profile.

(b) Narrow Band Conductance.

(c) Wide Band Conductance.

(d) Frequency Response.

Figure 37. Eigen Window Filter Unweighted Transducer Data.

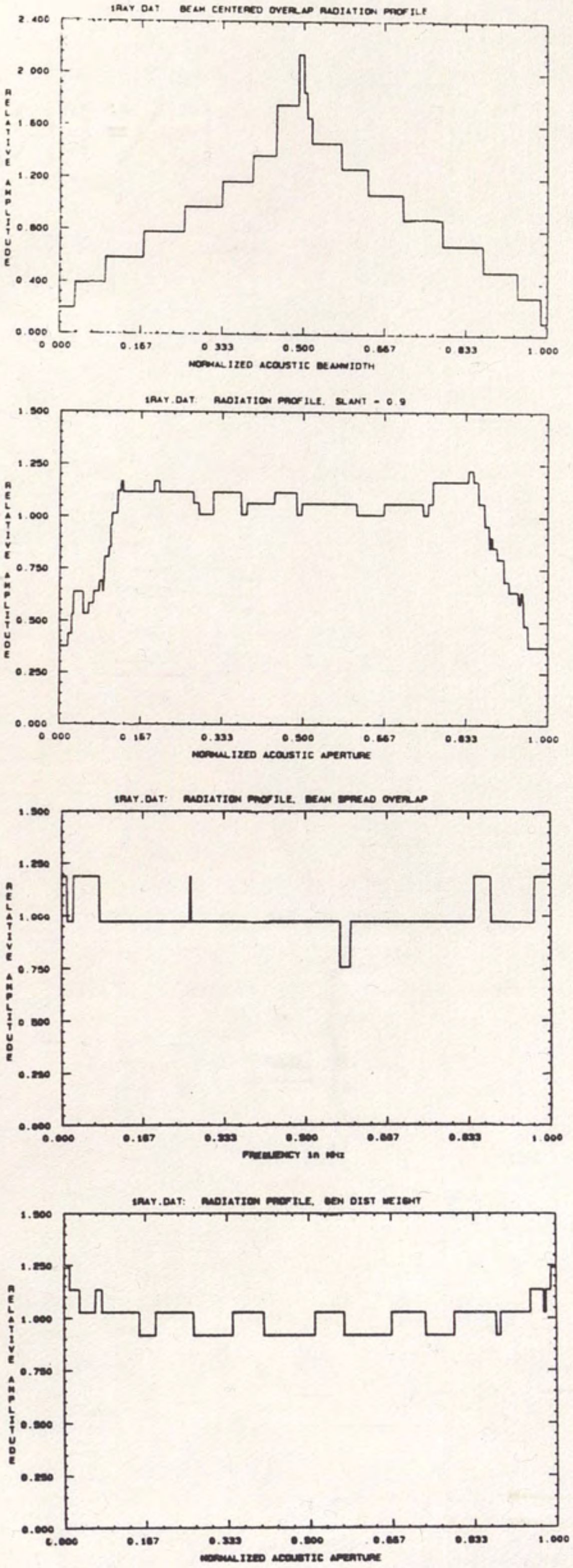


(a) Eigen Window Transducer Response.

(b) Narrow Band Filter Response.

(c) Wide Band Filter Response.

Figure 38. Amplitude Weighted Eigen Window Filter Data.



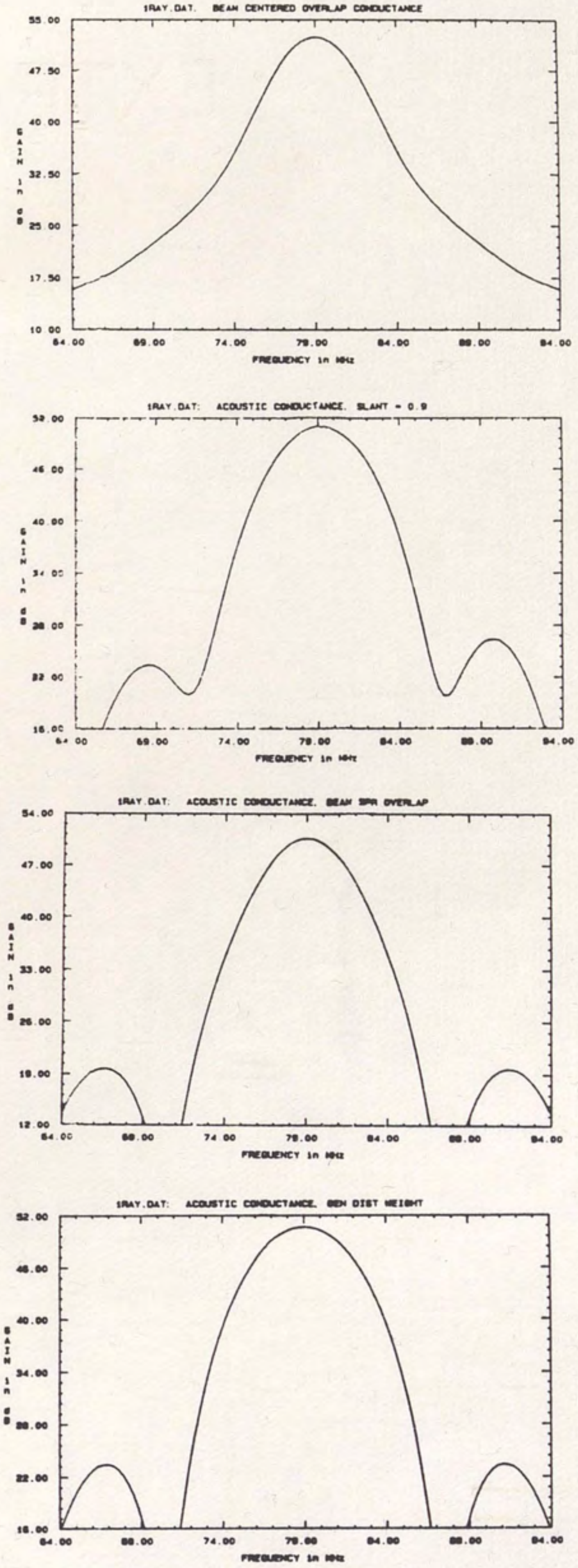
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 39. Eigen Window Transducer Center Frequency Radiation Profiles.



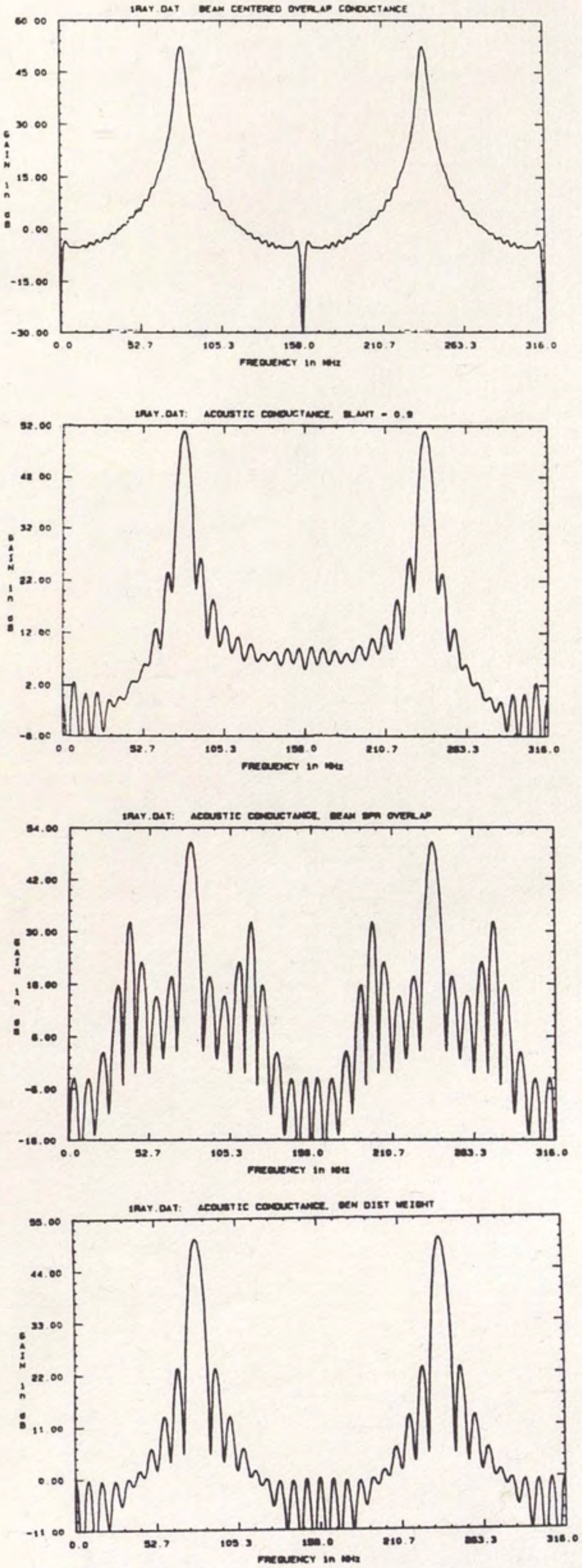
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 40. Eigen Window Transducer Passband Acoustic Conductance.



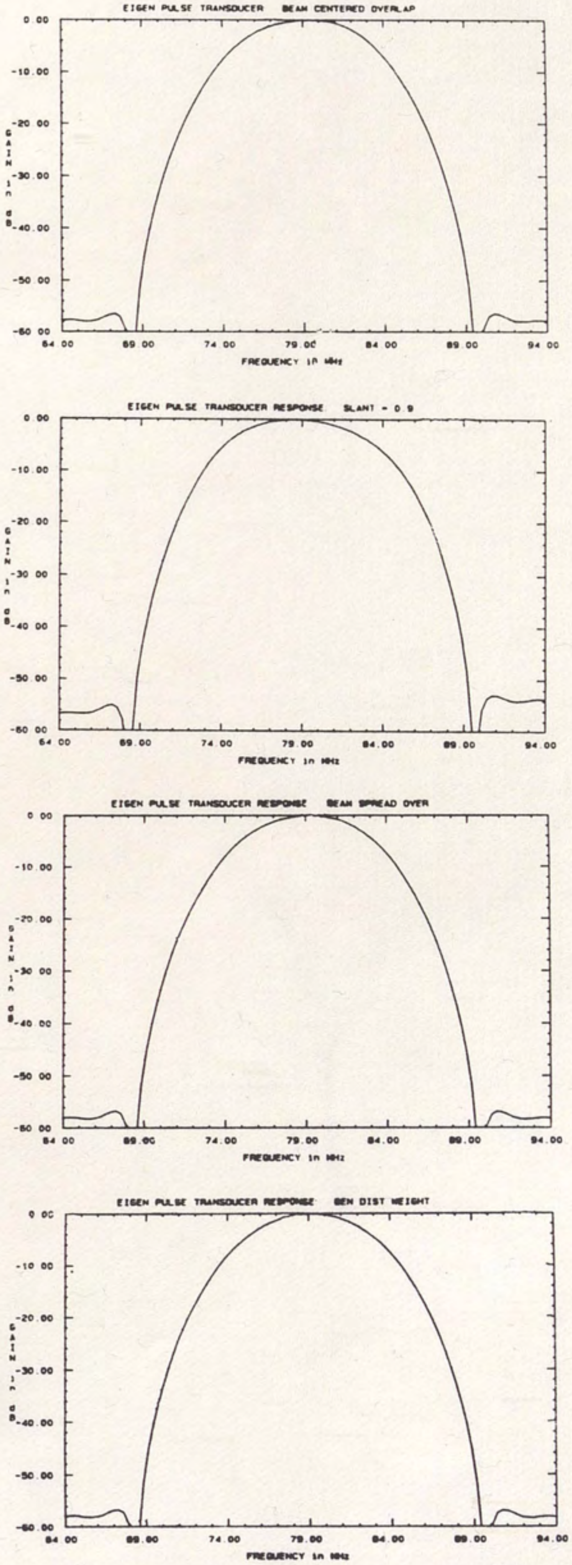
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 41. Eigen Window Transducer Wideband Acoustic Conductance.



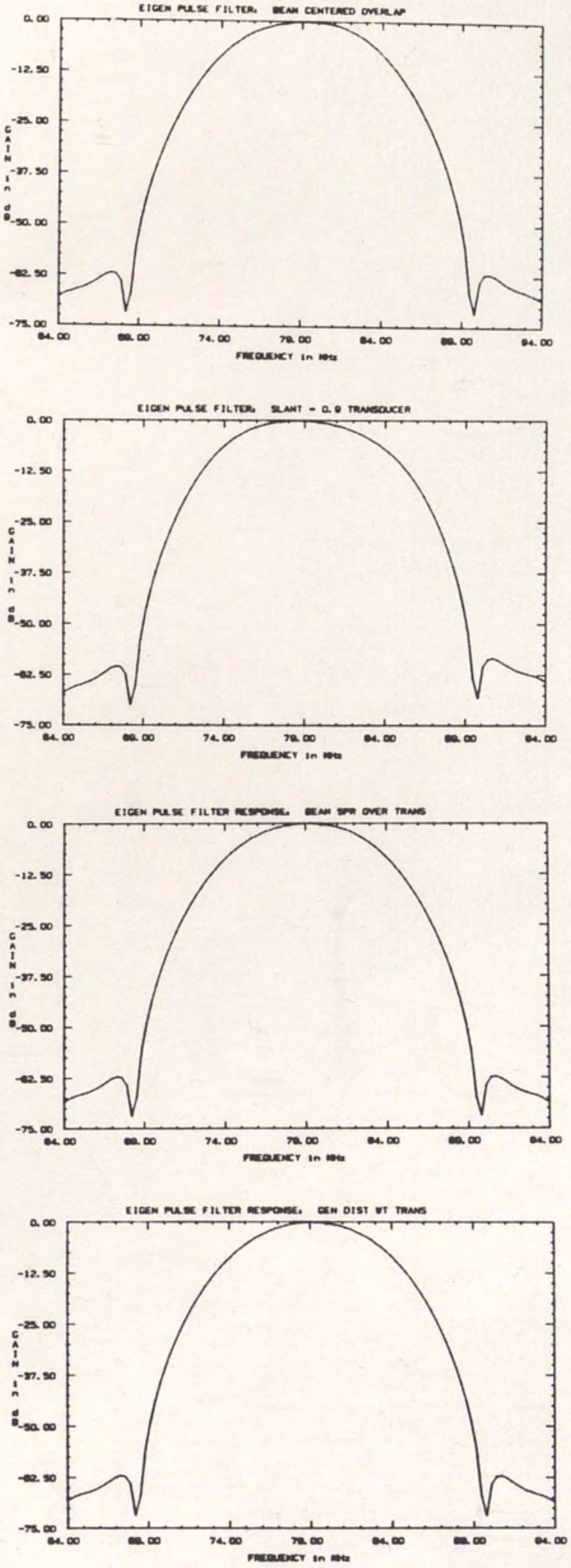
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 42. Eigen Window Transducer Passband Responses.



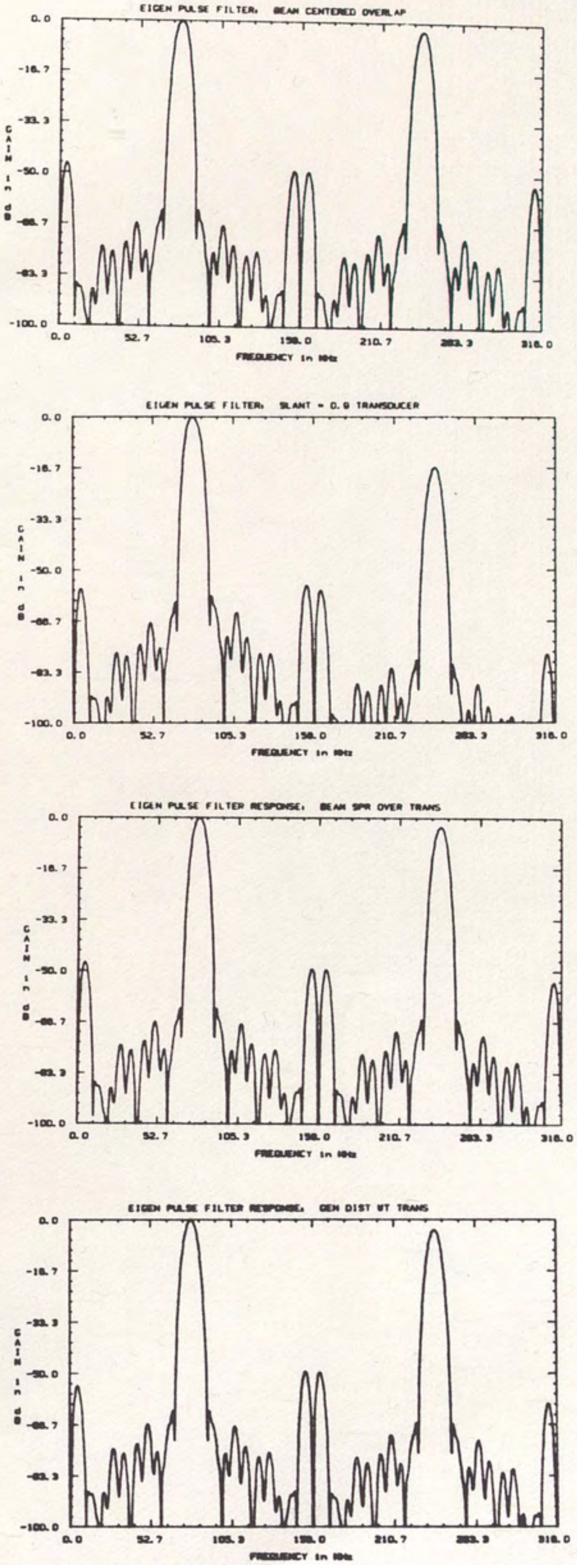
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 43. Eigen Window Filter Passband Responses.



(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 44. Eigen Window Filter Wideband Responses.

APPENDIX D

LOW SHAPE FACTOR FILTER DATA

TABLE V

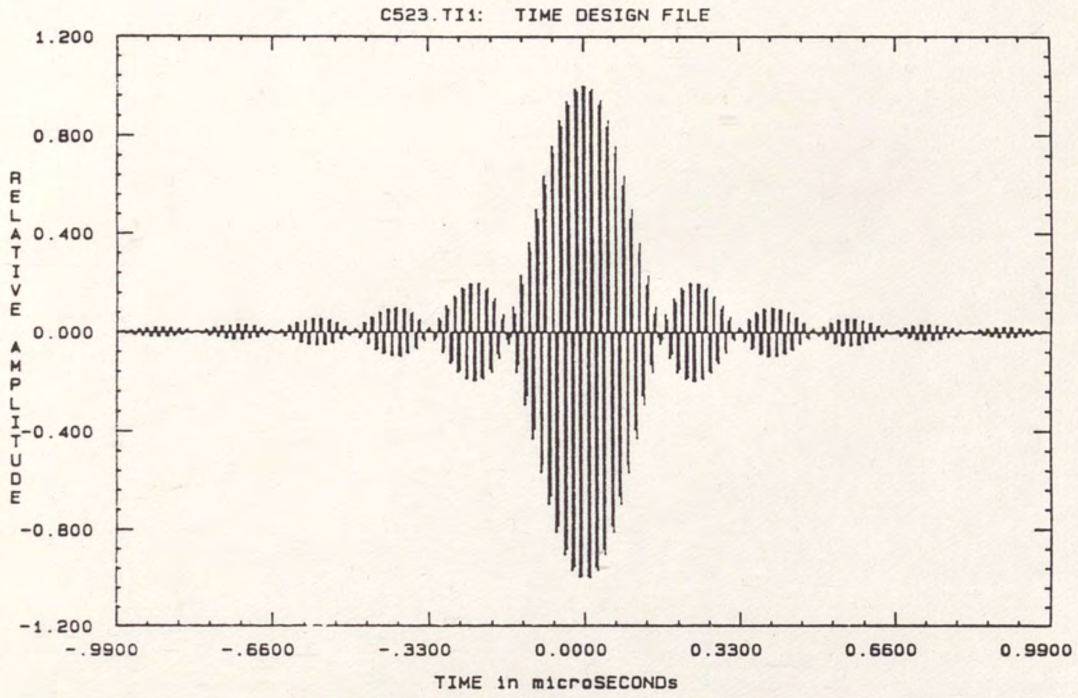
LOW SHAPE FACTOR FILTER PARAMETERS

Coupling Coefficient	.056
Acoustic Velocity	3925 meters/sec.
Acoustic Beamwidth	35 λ_o
Electrode Capacitance	5.7 pF./cm.
Metal Resistivity	0.18 Ohms/sq.
Center Frequency	60 MHz.

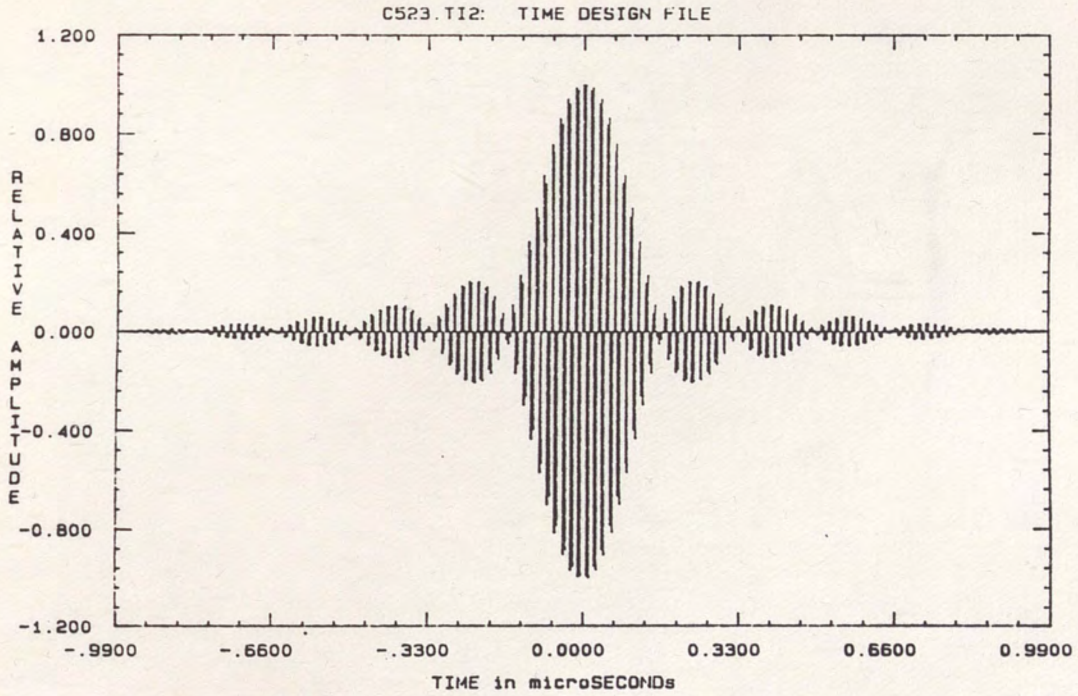
TABLE VI

APODIZATION LOSS DATA FOR SECOND TRANSDUCERS OF LOW SHAPE FACTOR FILTERS.

TRANSDUCER TYPE	APODIZATION LOSS
Beam Centered Overlap	1.1907 dB.
Slant Centered Overlap	1.4944 dB.
Beam Spread Overlap	0.6329 dB.
Gen. Dist. Weighting	0.5869 dB.



(a) Input Transducer.



(b) Output Transducer.

Figure 45. Low Shape Factor Filter Time Design Data.

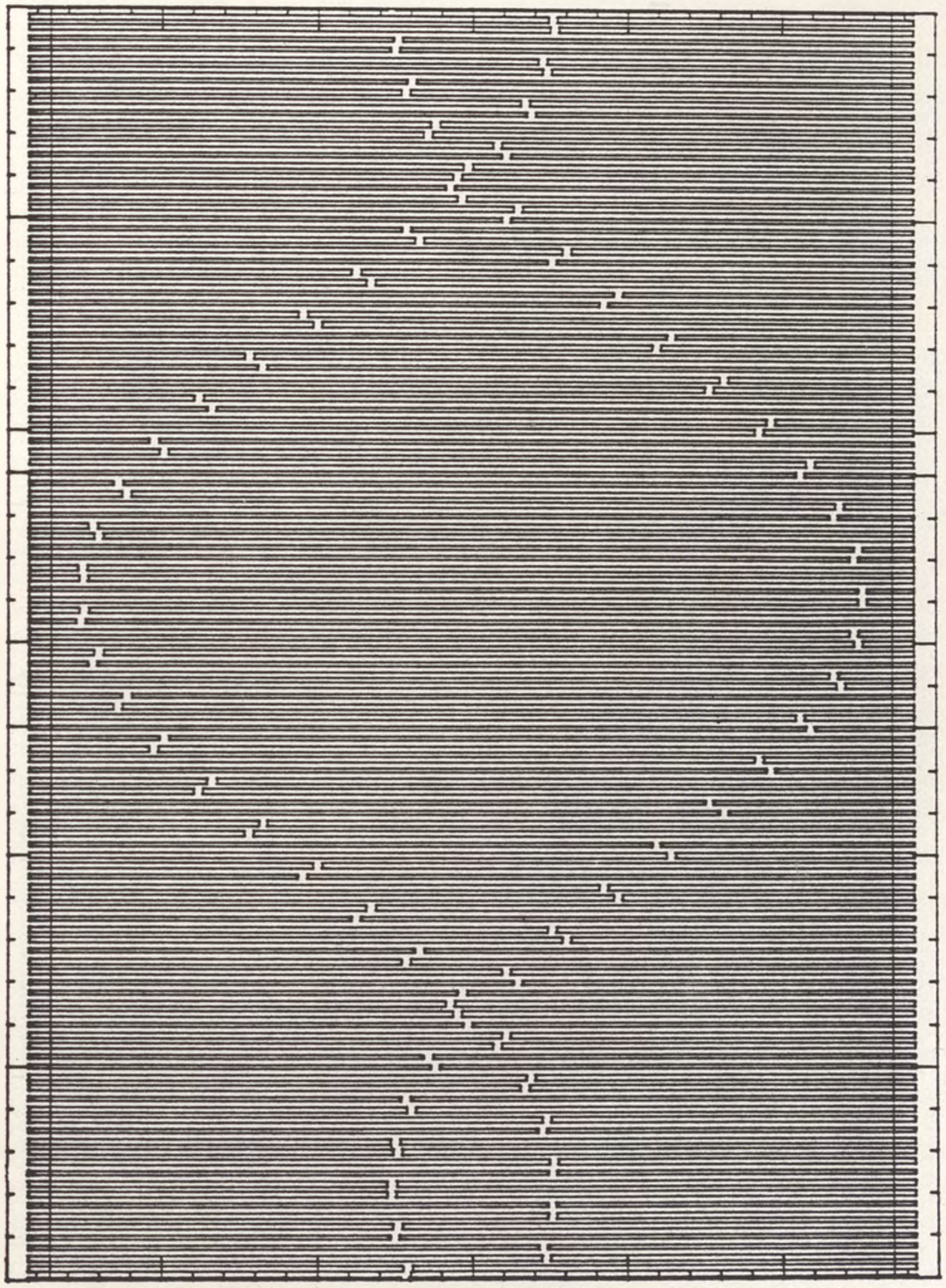


Figure 46. Low Shape Factor Filter Beam Centered Overlap Transducer Structure.

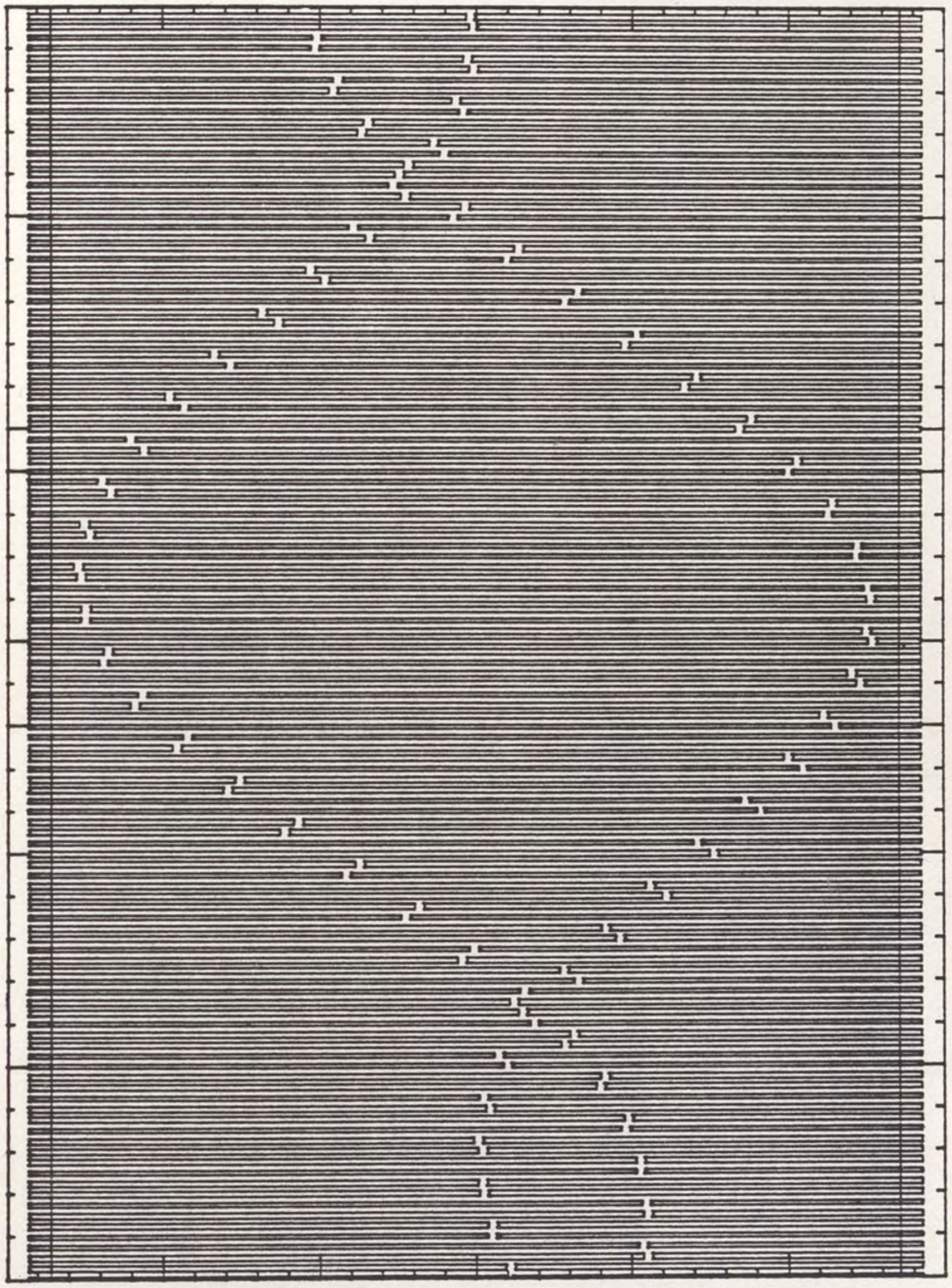


Figure 47. Low Shape Factor Filter Slant Centered Overlap Transducer Structure.

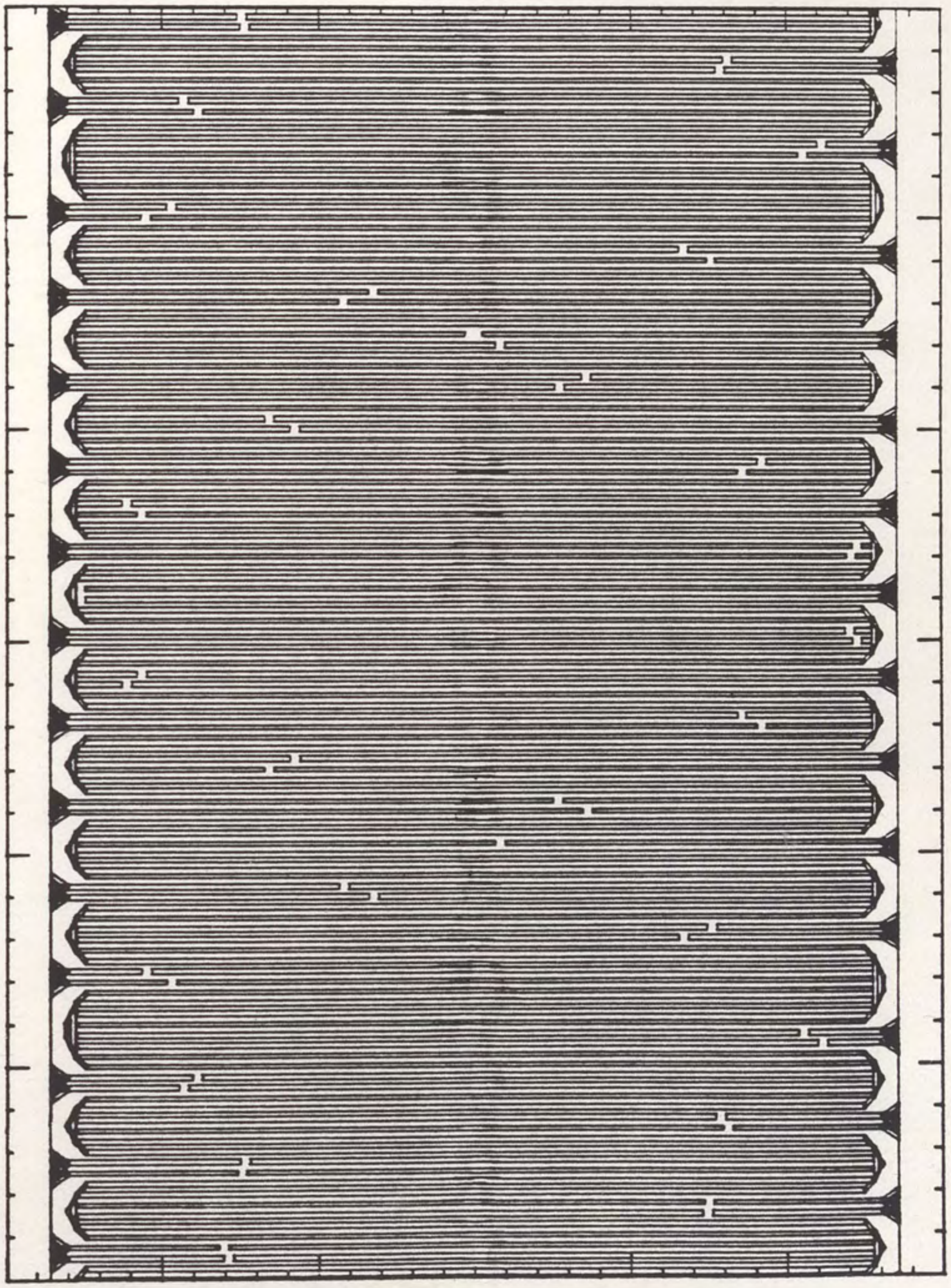


Figure 48. Low Shape Factor Filter Beam Spread Overlap Transducer Structure.

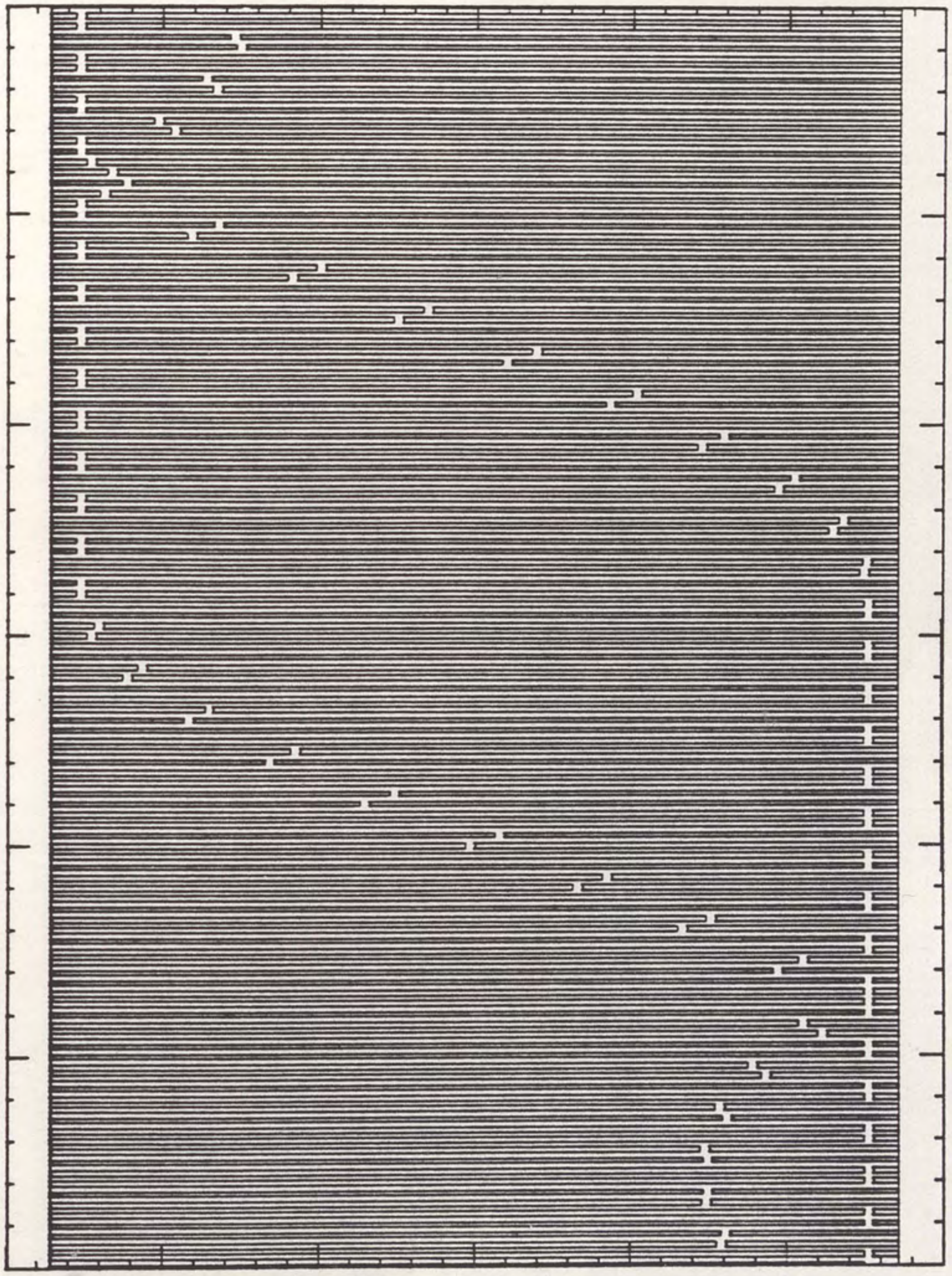
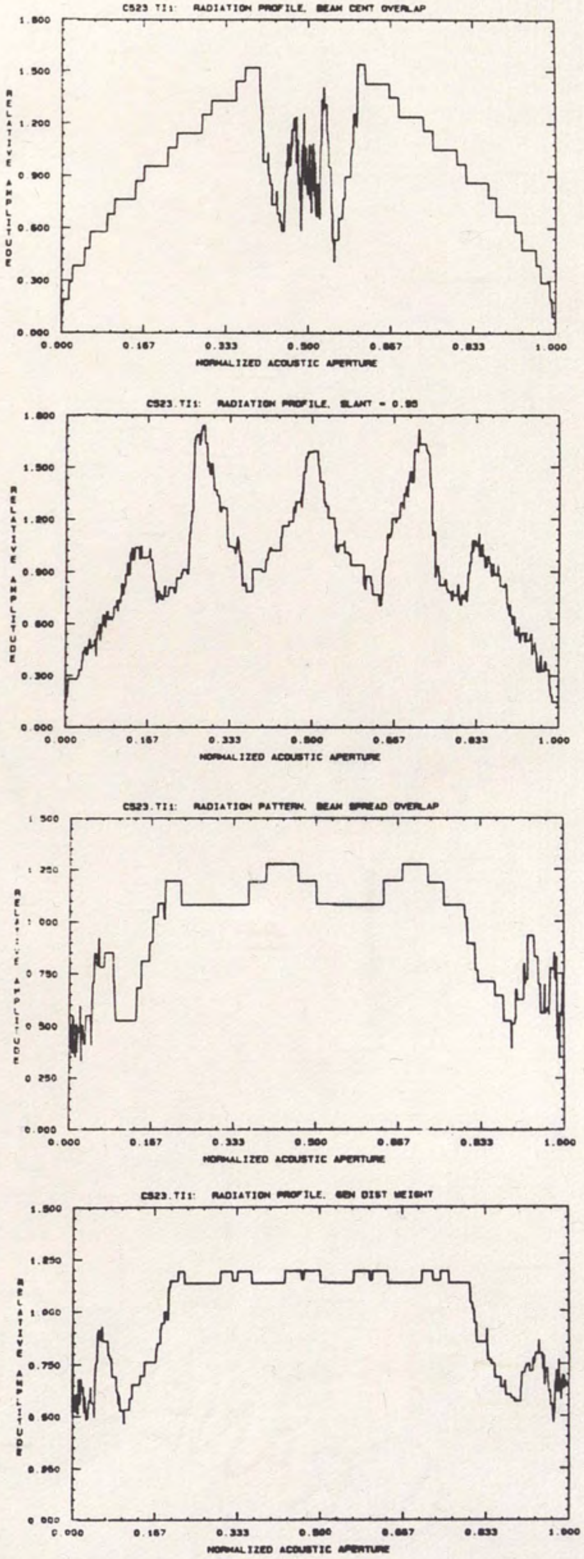


Figure 49. Low Shape Factor Filter Generalized Distance Weighted Transducer Structure.



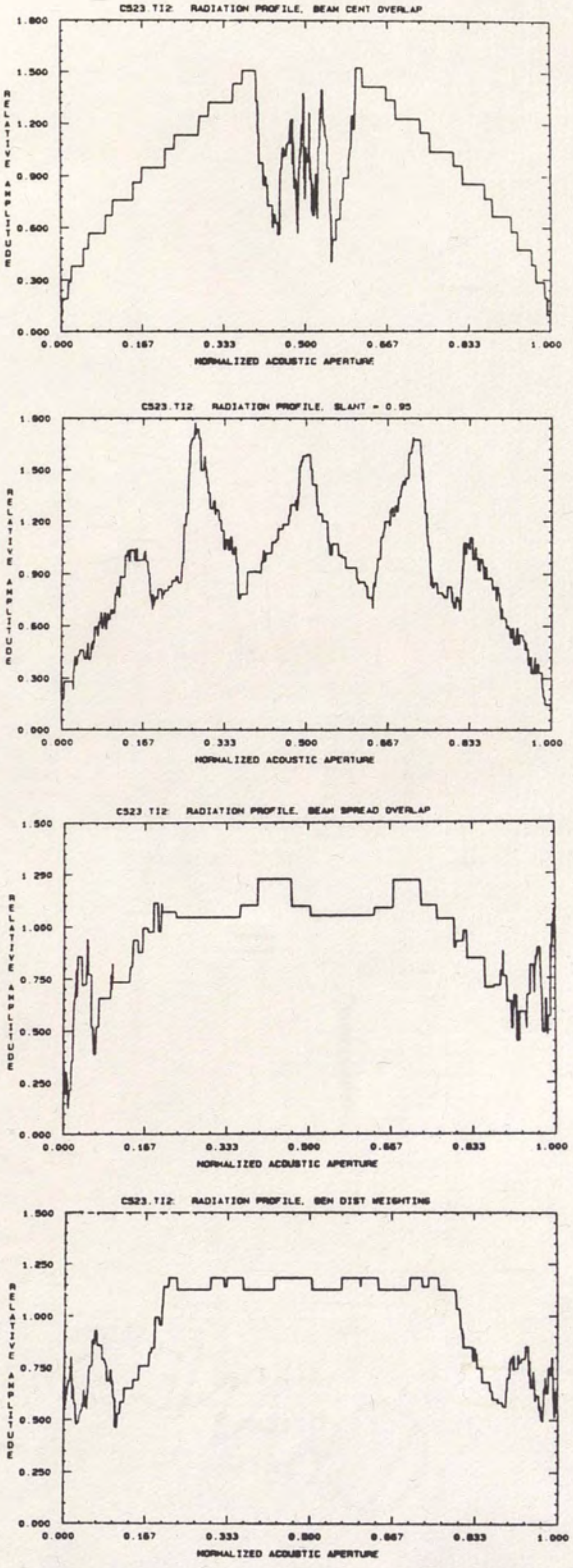
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 50. Low Shape Factor Filter Input Transducer Center Frequency Radiation Profile Data.



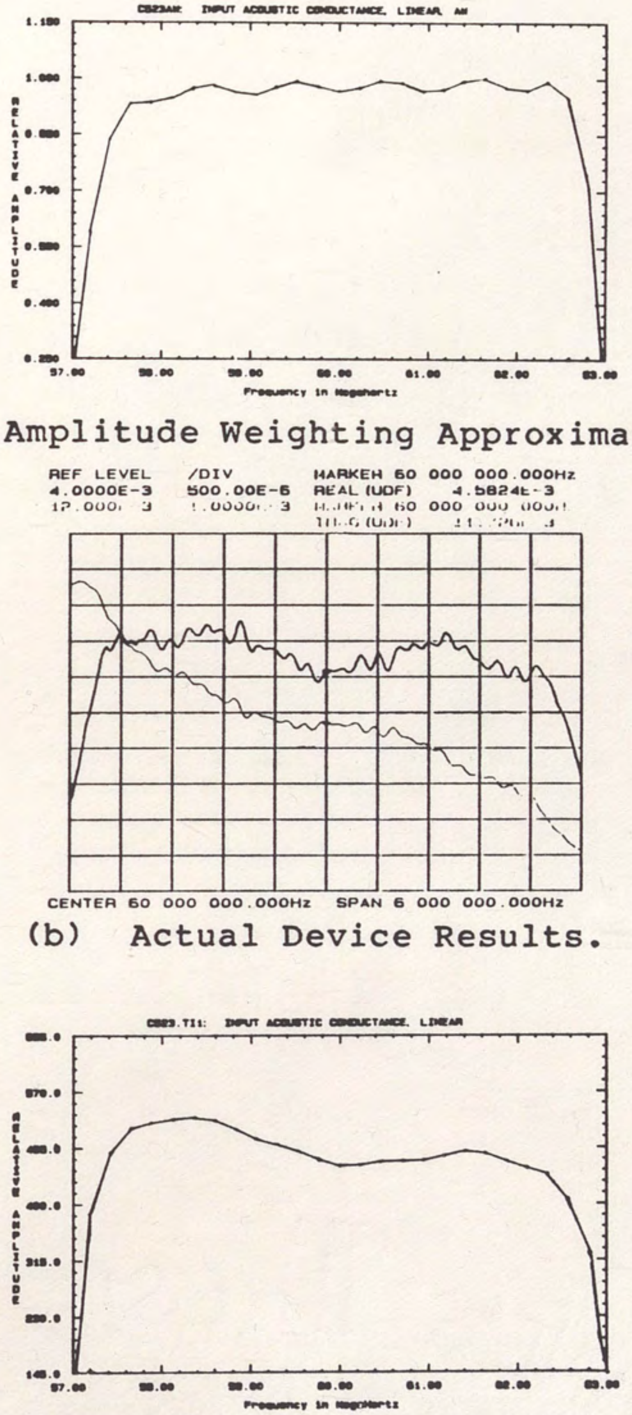
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 51. Low Shape Factor Filter Output Transducer Center Frequency Radiation Profile Data.

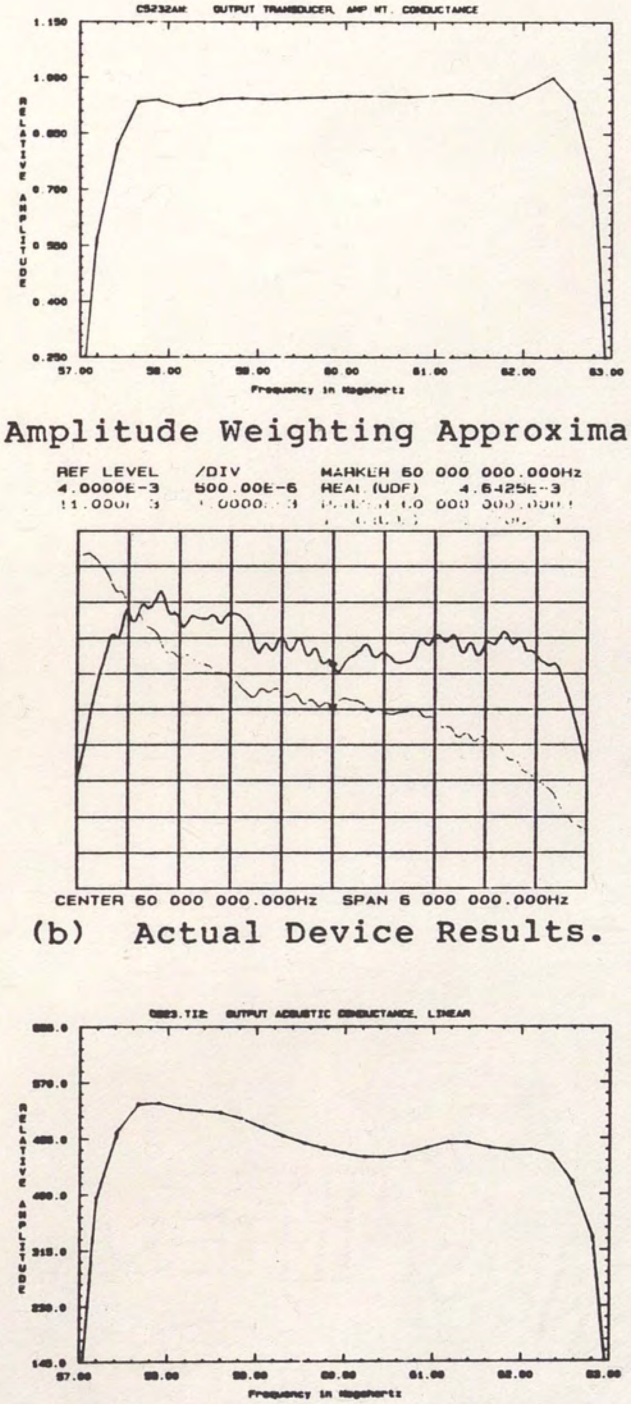


(a) Amplitude Weighting Approximation.

(b) Actual Device Results.

(c) Minimum Track Analysis Model.

Figure 52. Low Shape Factor Filter Input Transducer, Comparison of Simulations With Experiment For Acoustic Conductance of Slant Centered Overlap Apodized Transducer.

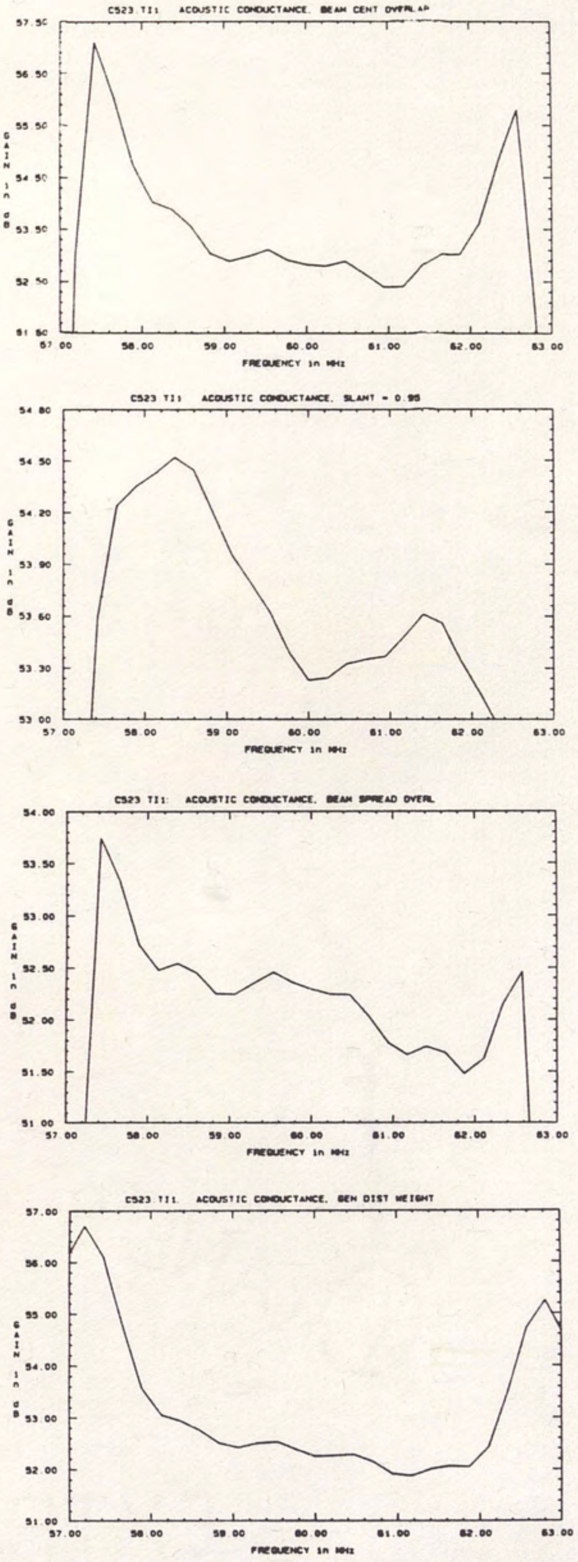


(a) Amplitude Weighting Approximation.

(b) Actual Device Results.

(c) Minimum Track Model.

Figure 53. Low Shape Factor Filter Output Transducer, Comparison of Simulations With Experiment For Acoustic Conductance of Slant Centered Overlap Apodized Transducer.



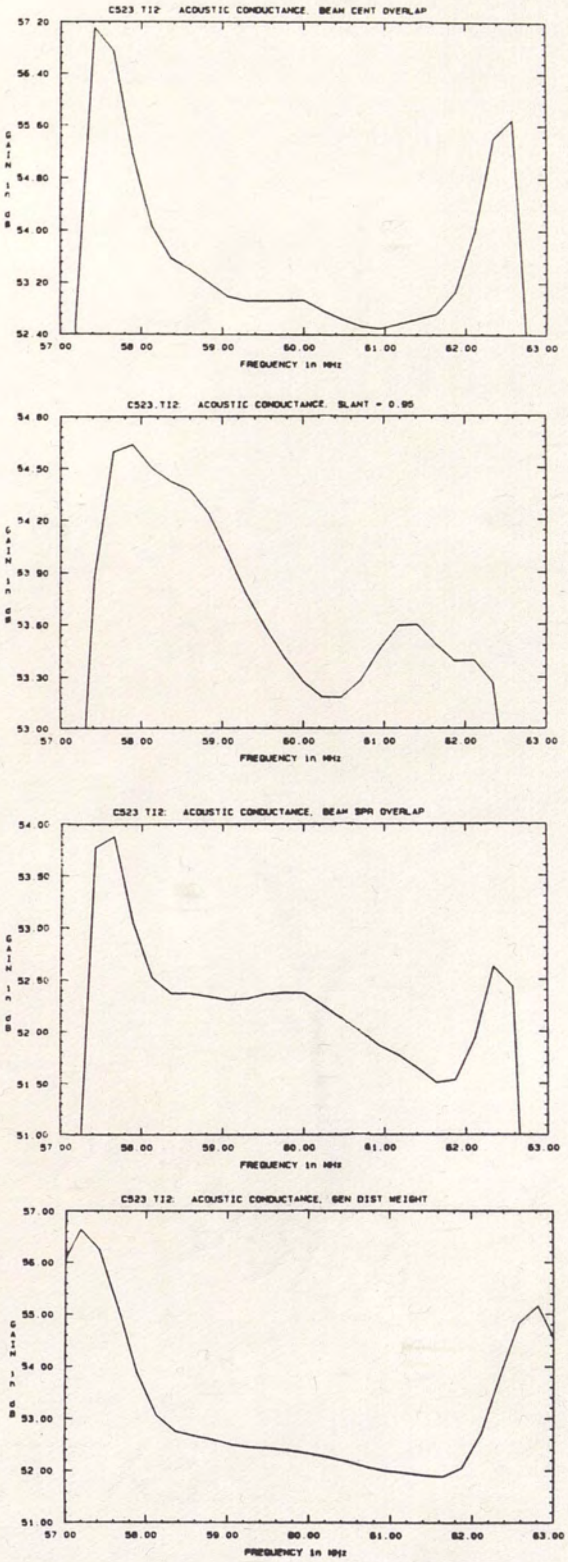
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 54. Low Shape Factor Filter Input Transducer Passband Acoustic Conductance.



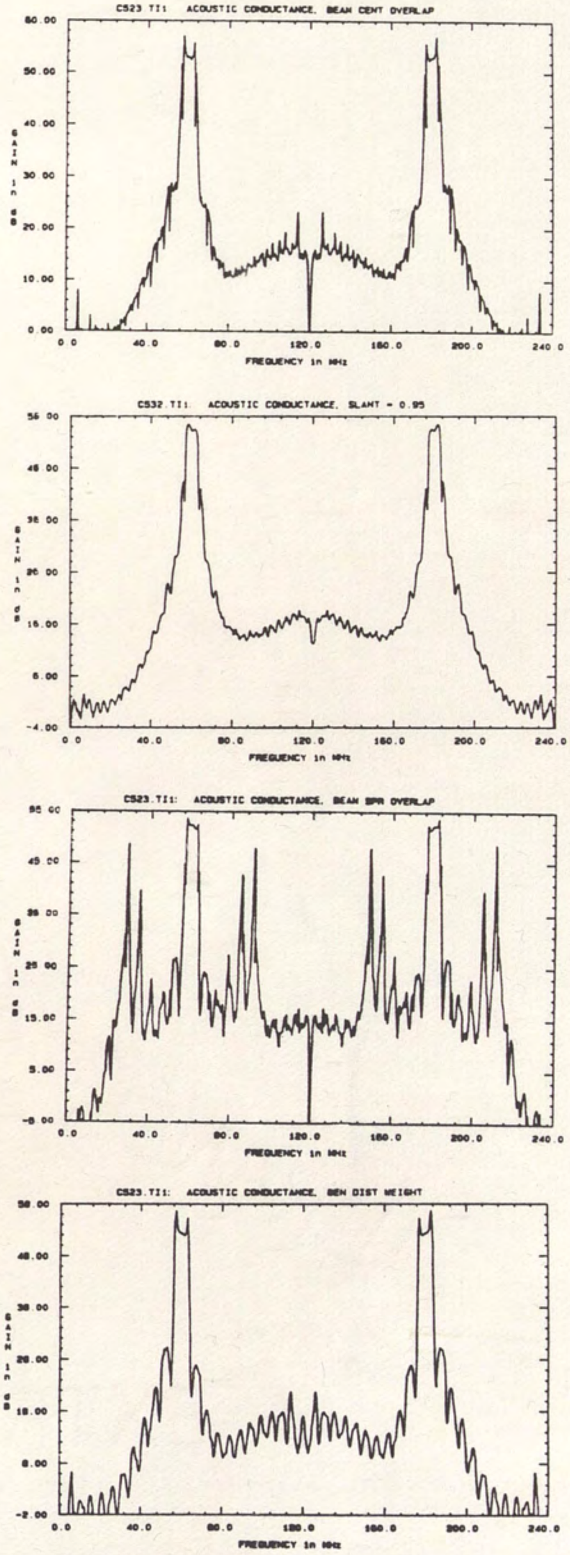
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 55. Low Shape Factor Filter Output Transducer Passband Acoustic Conductance.



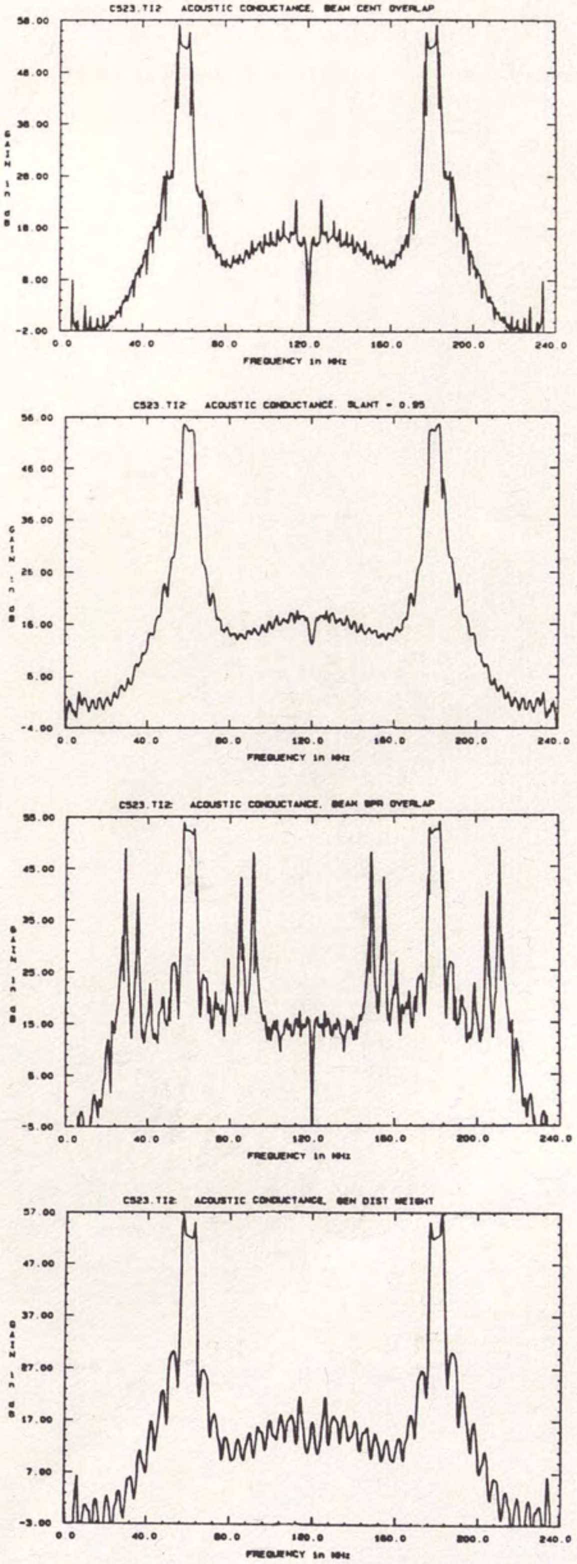
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 56. Low Shape Factor Filter Input Transducer Wideband Acoustic Conductance.



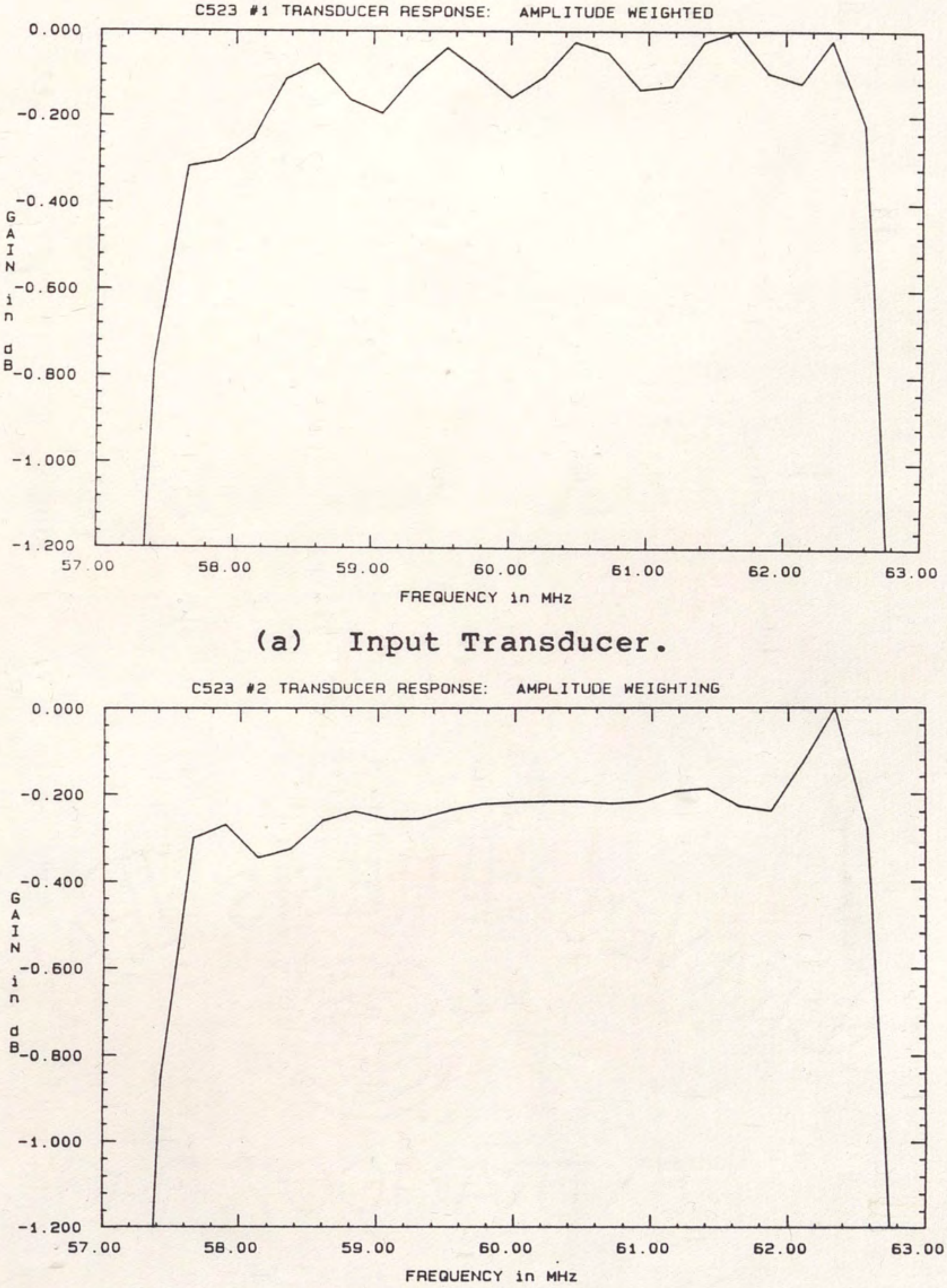
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

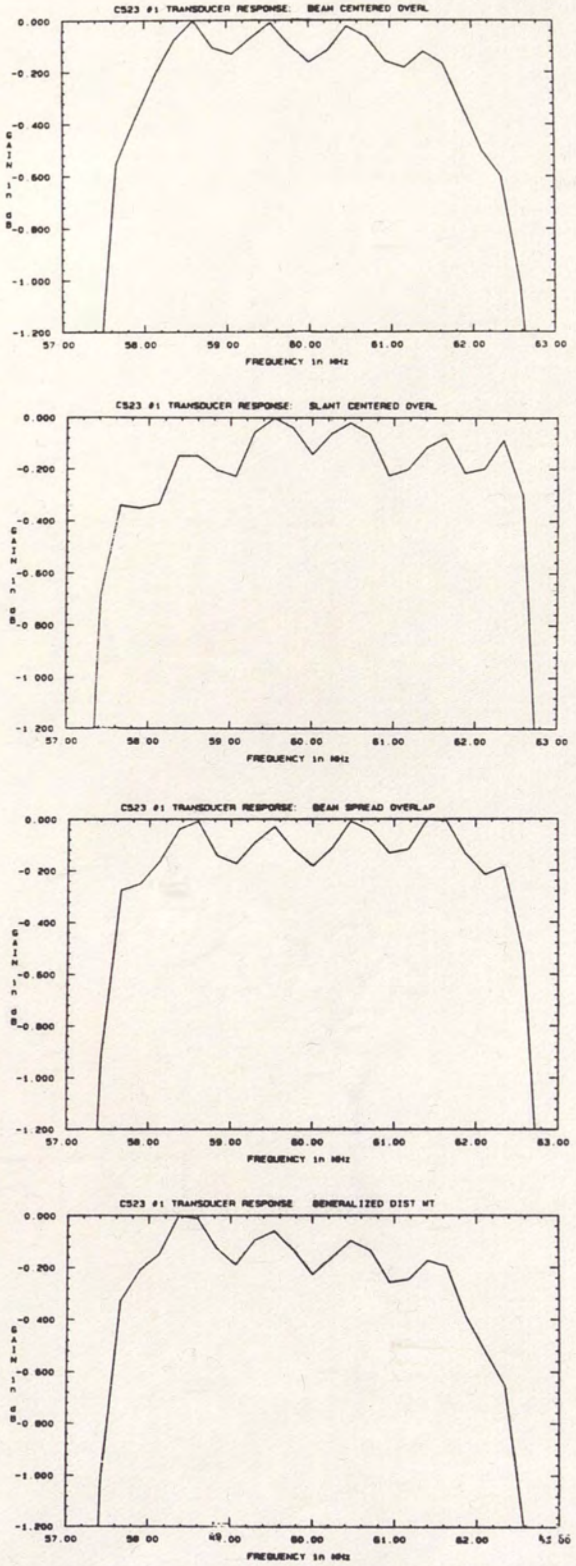
Figure 57. Low Shape Factor Filter Output Transducer Wideband Acoustic Conductance.



(a) Input Transducer.

(b) Output Transducer.

Figure 58. Low Shape Factor Filter Amplitude Weighted Transducer Responses.



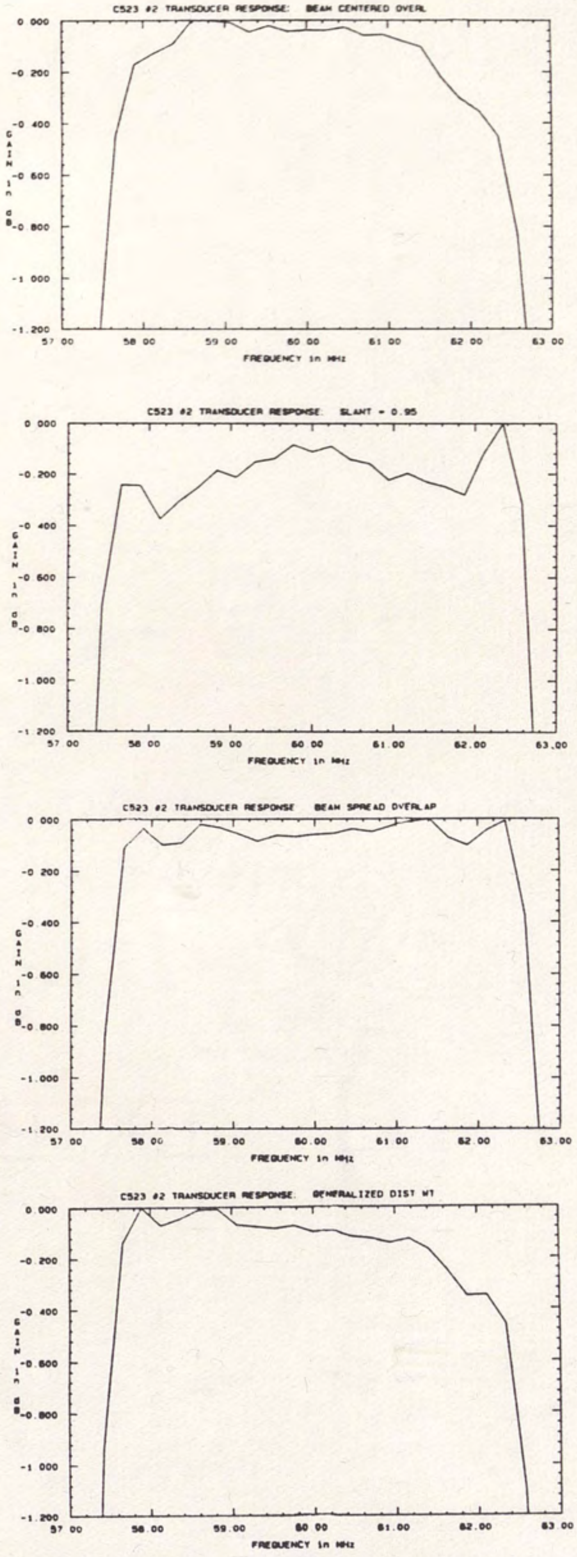
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 59. Low Shape Factor Filter Input Transducer Passband Responses.



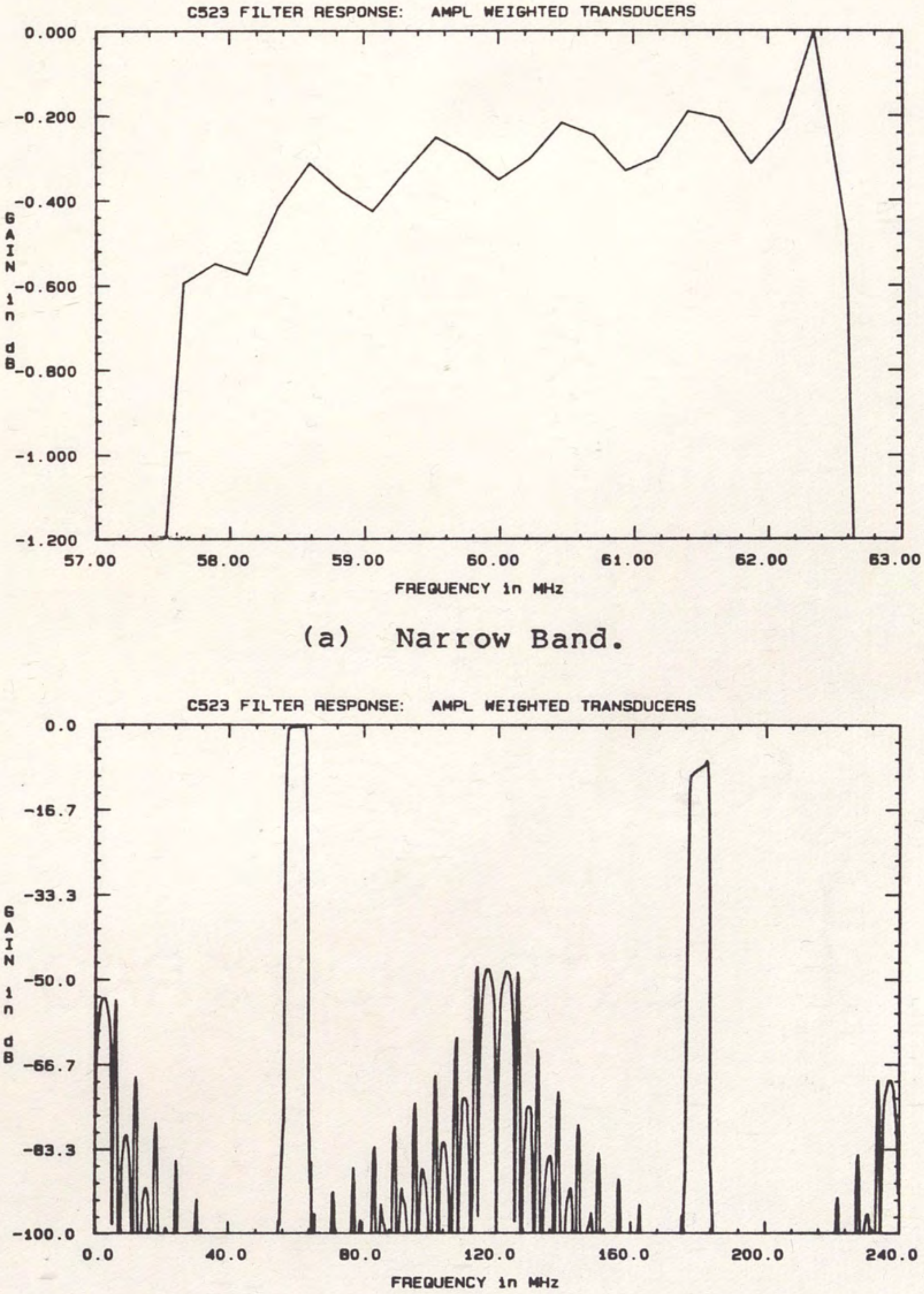
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

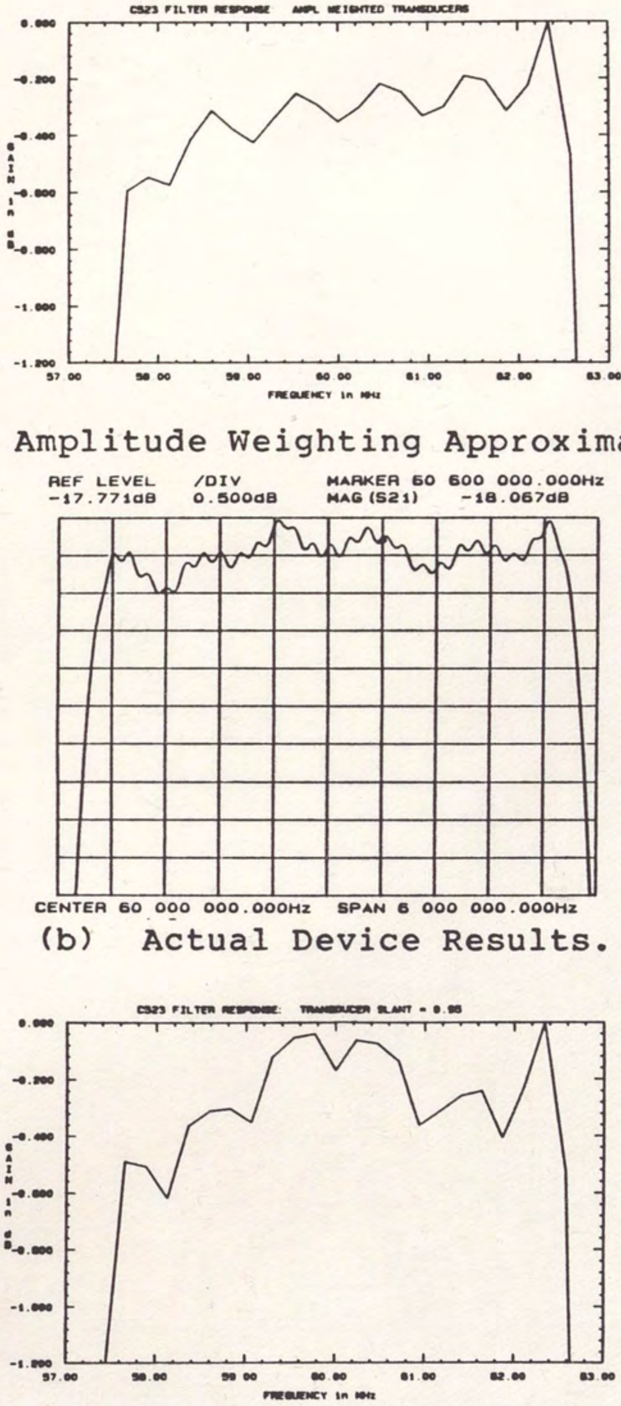
Figure 60. Low Shape Factor Filter Output Transducer Passband Responses.



(a) Narrow Band.

(b) Wide Band.

Figure 61. Low Shape Factor Filter Amplitude Weighted Filter Response.

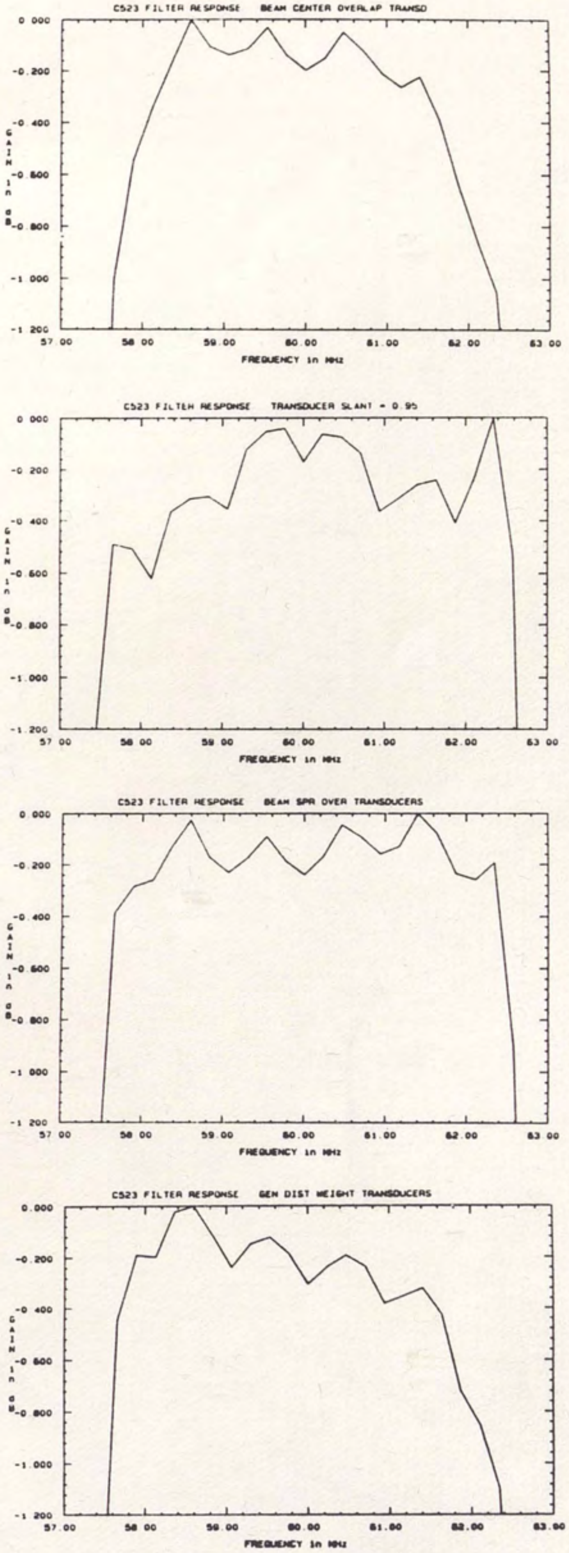


(a) Amplitude Weighting Approximation.

(b) Actual Device Results.

(c) Minimum Track Model.

Figure 62. Low Shape Factor Filter, Comparison of Response Simulations With Experiment.



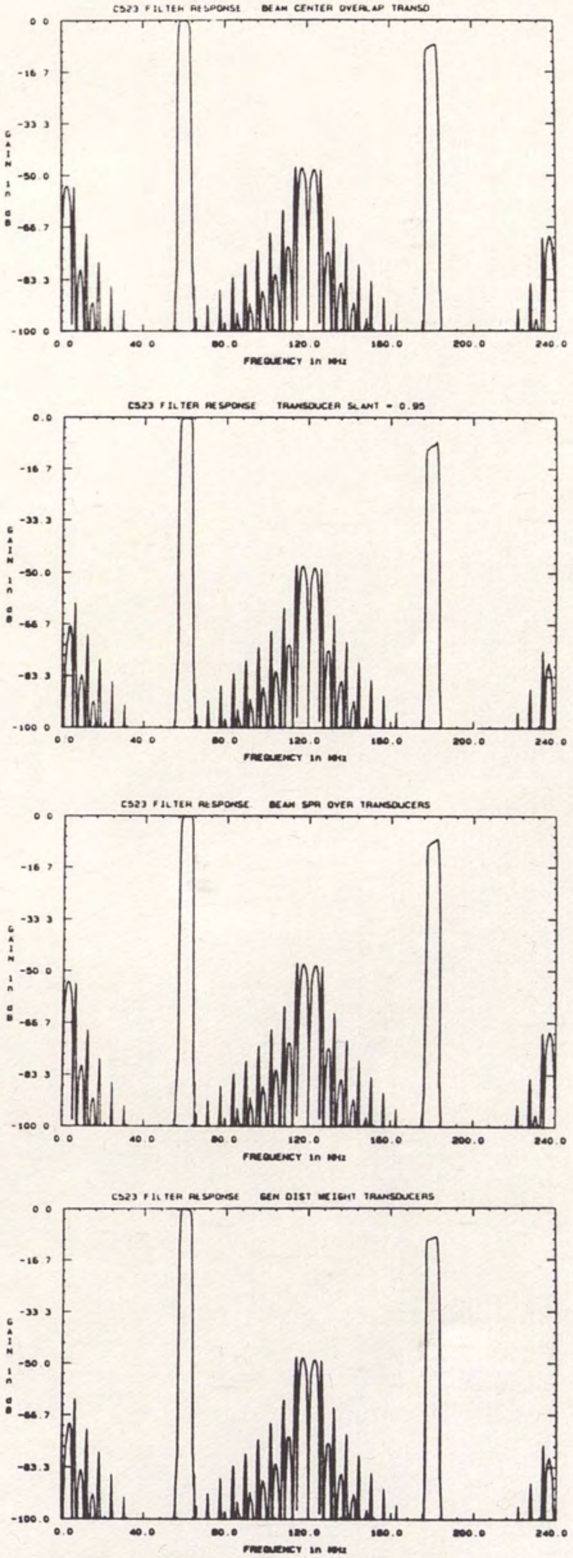
(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 63. Low Shape Factor Filter Passband Responses.



(a) Beam Centered Overlap.

(b) Slant Centered Overlap.

(c) Beam Spread Overlap.

(d) Generalized Distance Weighting.

Figure 64. Low Shape Factor Filter Wideband Responses.

LIST OF REFERENCES

1. Hartmann, C. S.; Bell, Jr., D. T.; and Rosenfeld, R. C. "Impulse Model Design of Acoustic Surface-Wave Filters." IEEE Transactions on Microwave Theory and Techniques 21 (April 1973): 162-175.

2. Ganguly, A. K. and Vassell, M. O. "Frequency Response of Acoustic Surface Wave Filters." Journal of Applied Physics 44 (March 1973): 1072-1085.

3. Defranould, J. D. "Low Loss SAW Bandpass Filters." IEEE Ultrasonics Symposium Proceedings (1981): 17-22.

4. Malocha, D. C. "Surface Wave Devices Using Low Loss Filter Technologies." IEEE Ultrasonics Symposium Proceedings (1981): 83-88.

5. Lewis, B.; Jordan, P. M.; Milsom, R. F.; and Morgan, D. P. "Charge and Field Superposition Methods for Analysis of Generalized SAW Interdigital Transducers." IEEE Ultrasonics Symposium Proceedings (1978): 709-714.

6. Datta, S.; Hunsinger, B. J.; and Malocha, D. C. "A Generalized Model for Periodic Transducers with Arbitrary Voltages." IEEE Transactions on Sonics and Ultrasonics 26 (May 1979): pp. 235-242.

7. Oates, D. E.; Smythe, D. L.; and Green, J. B. "SAW/FET Programmable Transversal Filter with 100 MHz. Bandwidth and Enhanced Programmability." IEEE Ultrasonics Symposium Proceedings (1985): To Be Published.

8. Panasik, C. M. and Zimmerman, D. E. "A 16 Tap Hybrid Programmable Transversal Filter Using Monolithic GaAs Dual-Gate FET Array." IEEE Ultrasonics Symposium Proceedings (1985): To Be Published.

9. Malocha, D. C. and B. J. Hunsinger, B. J. "Capacitive Tap Weighted SAW Transducers With Reduced Losses." IEEE Ultrasonics Symposium Proceedings (1977): 763-766.

10. Slobodnik, A. J.; Laker, K. R.; Szabo, T. L.; Kearns, W. J.; and Roberts, G. A. "Low Sidelobe SAW Filters Using Overlap and Withdrawal Weighted Transducers." IEEE Ultrasonics Symposium Proceedings (1977): 757-762.

11. Gafni, H.; Biran, A.; and Halzel, A. "On the Radiation Conductance of Apodized SAW Transducers." IEEE Ultrasonics Symposium Proceedings (1985): To Be Published.

12. Yamanouchi, K.; Meguro, T.; and Shibayama, K. "Acoustic Surface Wave Filters Using New Distance Weighting Techniques." IEEE Ultrasonics Symposium Proceedings (1980): 313-316.

13. Gerard, H. M. "Surface Wave Interdigital Electrode Chirp Filters." In Surface Wave Filters. Edited by H. Matthews. New York: John Wiley and Sons, 1977.

14. Naraine, P. M. and Campbell, C. K. "Wide Band Linear Phase SAW Filters Using Apodized Slanted Finger Transducers." IEEE Ultrasonics Symposium Proceedings (1983): 113-116.

15. Datta, S. and Hunsinger, B. J. "Element Factor for Periodic Transducers." IEEE Transactions on Sonics and Ultrasonics 27 (January 1980): 42-44.

16. Waldron, R. A. "Power Transfer Factors for Nonuniformly Irradiated Interdigital Piezoelectric Transducers." IEEE Transactions on Sonics and Ultrasonics 19 (October 1972): 448-453.

17. Gerard, M. M. "Principles of Surface Wave Filter Design." In Topics in Applied Physics: Acoustic Surface Waves. Edited by A. A. Oliner. New York: Springer-Verlag, 1978.

18. Ristic, V. M. Principles of Acoustic Devices. New York: John Wiley and Sons, 1983.

19. Richie, S. M. "Three Phase Unidirectional Surface Acoustic Wave Transducer Model and Computer Aided Design Implementation." Thesis, University of Central Florida, Spring 1983.

20. Yap, R. L. "Surface Acoustic Wave Bidirectional Filter Synthesis and Analysis." Thesis, University of Central Florida, Fall 1984.

21. Bishop, C. D. and Malocha, D. C. "Non-Iterative Design of SAW Bandpass Filters." IEEE Ultrasonics Symposium Proceedings (1984): 18-21.

22. Bishop, C. D. "Non-Iterative Design of SAW Bandpass Filters." Thesis, University of Central Florida, Spring 1985.

23. Burrus, C. S. and Parks, T. W. DFT/FFT and Convolution Algorithms. New York: John Wiley and Sons, 1985.

24. Streibl, I. "SAW Diffraction Compensation in LiNbO3." Thesis, Carleton University, January 1984.

25. Datta, S. Surface Acoustic Wave Devices. Englewood Cliffs: Prentice-Hall, 1986.