

A Constructive Algorithm for Determining Array Shape with Application to FREDDEX

LAWRENCE J. ROSENBLUM

Systems Research and Technology Directorate

DONALD R. DEL BALZO

*Large Aperture Acoustics Branch
Acoustics Division*

September 30, 1981



NAVAL RESEARCH LABORATORY
Washington, D.C.

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NRL Report 8531	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A CONSTRUCTIVE ALGORITHM FOR DETERMINING ARRAY SHAPE WITH APPLICATION TO FREDDEX	5. TYPE OF REPORT & PERIOD COVERED Final report on one phase of a continuing NRL problem.	
	6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) Lawrence J. Rosenblum and Donald R. Del Balzo	8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, DC 20375	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 51-0367-0-1 62759N XF59552100	
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Electronics Systems Command Washington, DC 20360	12. REPORT DATE September 30, 1981	
	13. NUMBER OF PAGES 35	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Underwater acoustics Array shape Towed arrays Array performance		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Long, towed hydrophone arrays are useful in a sonar context as high-gain, high-resolution receivers. However, conventional beamforming assumes a straight, horizontal aperture. Towed arrays undergo dynamic shape changes due to ship track instabilities and to subsurface currents. This report documents a constructive algorithm for determining array shape as well as the results of using this algorithm to determine array shapes from the Spring 1979 FREDDEX experiment. It is shown that the algorithm works successfully on this data set and indicates that array deforma- <div style="text-align: right;">(Continued)</div>		

20. ABSTRACT (Continued)

tion for FREDDEX was minimal. A typical value for the standard deviation of the array was 0.2 m. The lowest spatial mode (the bow shape) predominated and cycled in about 16 min. Acoustic degradation was evaluated for several frequencies and was always well below one dB. Increased sidelobes were apparent, but they were still 35 dB below the maximum response.

CONTENTS

INTRODUCTION	1
Background on the Array Shape Problem	1
Background on FREDDEX	1
Description of the FREDDEX Array Shape Experiment	2
THE ARRAY SHAPE ALGORITHM	2
Description of Algorithm	2
Additional Remarks on the Algorithm	4
DATA PROCESSING	5
Digitizing and Time Delays	5
Array Shapes	7
Statistics	11
Beamforming	12
CONCLUSIONS	12
AKNOWLEDGMENTS	14
REFERENCE	14
Appendix	15

A CONSTRUCTIVE ALGORITHM FOR DETERMINING ARRAY SHAPE WITH APPLICATION TO FREDDX

INTRODUCTION

Background on the Array Shape Problem

Long, towed hydrophone arrays are useful in a sonar context as high-gain, high-resolution receivers. However, conventional beamforming assumes the array is straight and horizontal, which is generally not the case for the flexible, oil-filled hoses presently envisioned for surveillance applications. Future work will emphasize computer compensation of the array shapes prior to final beamforming. Recent work in this area has concentrated on measurement techniques, full-scale and model experimentation, and analysis of shapes. A general conclusion is that only a few of the lowest order spatial modes have a significant effect on array performance between 30 and 300 Hz.

An excellent summary of measurement techniques was presented by Martin [1]. He covered passive and active acoustic methods as well as non-acoustic sensor methods. He mentioned analysis techniques and their associated claims of statistical significance. In general, most of the combined measurement/analysis procedures claim a shape accuracy of 0.1 to 0.5 m. The actual results on a variety of arrays are contained in the classified literature.

Background on FREDDX

FREDDX, an integrated environmental-acoustic experiment conducted in the Atlantic Ocean near Bermuda in the spring of 1979, involved several ships, aircraft, and the use of satellites on a variety of underwater acoustic and environmental missions. The main objectives were to locate and describe a large oceanographic thermal anomaly and to conduct a series of acoustic measurements. The underlying purpose of the experiment was to assess the impact of ocean eddies on the acoustic performance of present and future undersea surveillance systems.

In recent years features with strong thermal gradients, such as the Gulf Stream and warm eddies, have been reported as affecting subsurface acoustic signal transmission. These features have been shown by experiment to reduce signal levels by as much as 15 dB, depending on geometry, source and receiver depths, and "strength" of thermal gradients. FREDDX was designed to extend previous knowledge to a cold eddy environment and to expand the analysis to include high-resolution measurements of the performance of towed arrays. Measurements such as array signal gain, resolving power, and target bearing accuracy were included. The initial acoustic results from a time-limited data set indicate good towed-array performance for the particular oceanographic feature studied and the specific array used.

To conduct a thorough, broadly supportive exercise, ten different organizations collaborated to provide expertise in the various source, receiver, and environmental areas of concern. Some experiments conducted in support of FREDDX were: satellite and over-the-horizon radar remote sensing of ocean surface characteristics, airborne sensing of shallow thermal structure, shipboard in-situ measurements of deep thermal structure, and towed and bottom-mounted receiver measurements of sound transmitted from moored and towed acoustic projectors.

Manuscript submitted August 13, 1981.

Description of the FREDDEX Array Shape Experiment

One particular receiver system used during the three weeks at sea phase of FREDDEX was a long, flexible hydrophone array towed by a surface ship at depths of several hundred meters. The array consisted of 60 hydrophones spaced 10 m apart. This corresponds to 32 wavelengths at 83 Hz or 88 wavelengths at 227 Hz. The array performance depends on many factors including the spatial coherence of the incoming acoustic wavefronts and the towed receiver shape and dynamics. The various factors influencing performance confound the results and thus are examined independently in order to determine their relative contributions.

A six hour time period at sea was devoted exclusively to measurements of array shape and array dynamics. The array deformation experiment consisted of the receiver ship towing the hydrophone array at a depth of 180 m for three hours and 260 m for an additional three hours at a position essentially broadside to the array. For these six hours the acoustic source towing ship moved along a track parallel to the receiver towing ship at a range of about 2.8 km. The acoustic projector was towed at a depth of 200 m. A 7 ms pulse (2 cycles at 300 Hz) with 150 Hz bandwidth was transmitted from the acoustic projector every 15 s. The pulse received by the array was on the order of 3 cycles due to source/transducer ringing. Absolute arrival times to each hydrophone were obtainable by providing each ship with a crystal controlled clock (synchronized within 1 ms to a common radio source, (WWV)) thus allowing accurate determination of the lowest order spatial mode, i.e. a bow shape, as well as the finer scale shape.

This report documents an algorithm developed for determining the shape of the towed array as well as the data processing techniques used on the collected acoustic FREDDEX data set to determine the array shape and its effect on acoustic performance.

THE ARRAY SHAPE ALGORITHM

Description of the Algorithm

At any fixed time t , let S denote the source and P_1, P_2, \dots, P_N denote the N hydrophone positions (the geometry is shown in Fig. 1). Put $\theta_i = \sphericalangle (P_i SP_{i+1})$, $\psi_i = \sphericalangle (P_i P_1 S)$ and $\phi_i = \sphericalangle (P_i P_1 P_{i-1})$. Let $a_i = |P_1 P_i|$ and b denote the fixed uniform distance between array elements. Define $\theta^{(i)} = \sum_{j=1}^i \theta_j$.

Assume that the N time delays $\Delta t_1, \Delta t_2, \dots, \Delta t_N$ between the source and the N hydrophones are known. Let c denote a representative sound speed in the ocean area of operation. Then the distances d_1, d_2, \dots, d_N between the hydrophones and the source are:

$$d_i = c \Delta t_i, \quad i = 1, \dots, N.$$

For fixed i , let X denote the intersection of $P_1 P_i$ with SP_{i-1} (extended if necessary).

Assuming that a_{i-1} is known, an iterative procedure for determining the i th hydrophone position will be derived. Since a_2 is the distance between consecutive hydrophones, this procedure can then be iteratively applied to determine all hydrophone positions. By the Law of Cosines:

$$\theta_i = \arccos \frac{d_i^2 + d_{i+1}^2 - b^2}{2d_i d_{i+1}}, \quad i = 1 \dots, N-1.$$

Hence $\theta^{(i-2)}$ and θ_{i-1} are known. Again by the Law of Cosines:

$$a_i^2 = d_1^2 + d_i^2 - 2d_1 d_i \cos \theta^{(i-1)}$$

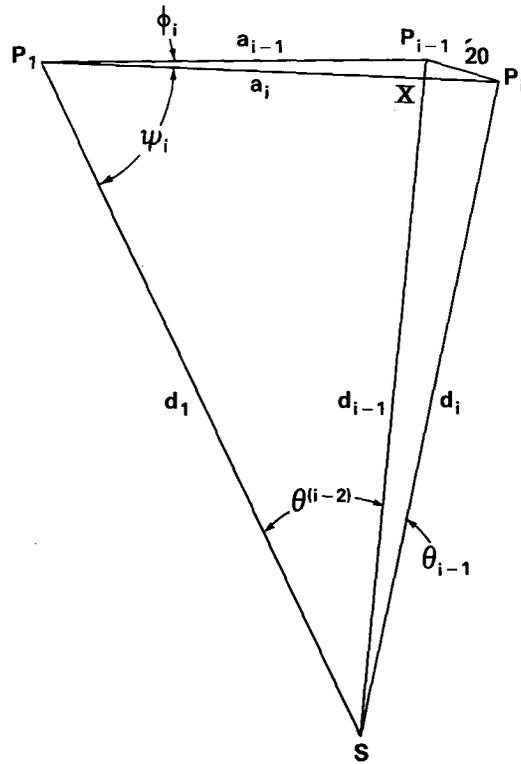


Fig. 1 — Geometry for the array shape algorithm

and

$$\phi_i = \arccos \frac{a_{i-1}^2 + a_i^2 - b^2}{2a_{i-1} a_i}.$$

Thus a_i and $|\phi_i|$ are known. Define y by $|SX| = d_{i-1} + y$. Associate a sign with ϕ_i by

$$\text{sgn}(\phi_i) = \begin{cases} +1 & \text{if } y \geq 0 \\ -1 & \text{if } y < 0 \end{cases}$$

By the Law of Sines:

$$\psi_i = \arcsin \frac{d_i \sin \theta^{(i-1)}}{a_i}$$

and

$$\frac{d_1}{\sin(\pi - \psi_i - \theta^{(i-2)})} = \frac{d_{i-1} + y}{\sin(\psi_i)}$$

i.e.,

$$y = \frac{d_1 \sin \psi_i}{\sin(\pi - \psi_i - \theta^{(i-2)})} - d_{i-1}.$$

Hence, ϕ_i is completely known and the i th phone position is thus determined. Since this procedure can be iteratively applied, the entire array shape can thus be determined.

Additional Remarks on the Algorithm

Since it is desired to use the array shape for additional processing, it is useful to associate a coordinate system with the solution. Define $\phi^{(k)} = \sum_{j=3}^k \phi_j$ for $3 \leq k \leq N$. Let $P_1 = (0, 0)$ and $P_1\bar{P}_2$ lie along the positive x axis. Then:

$$P_1 = (0, 0),$$

$$P_2 = (a_2, 0),$$

$$P_3 = (a_3 \cos \phi^{(3)}, a_3 \sin \phi^{(3)}),$$

$$P_4 = (a_4 \cos \phi^{(4)}, a_4 \sin \phi^{(4)}),$$

$$\vdots$$

$$P_N = (a_N \cos \phi^{(N)}, a_N \sin \phi^{(N)}), \quad \text{and } S = (d_1 \cos (\phi_3 - \psi_3), d_1 \sin (\phi_3 - \psi_3))$$

where ψ_3 is positive and ϕ_3 is taken with its sign.

Synthetic data were used to test the algorithm. The array was assumed to be piecewise linear and distances from an acoustic source to each hydrophone were computed assuming the perpendicular distance from the acoustic source to the array was 1.5 miles (as in the experiment). These distances were then input into a computer program designed to implement the algorithm. The original linear shape was successfully reproduced. To test the algorithm for statistical robustness, individual hydrophones within the simulated array were varied by amounts up to several meters. The shapes were then recomputed and it was found that the altered point(s) had almost no effect on the computed positions for the remaining hydrophones. Finally, several piecewise linear shapes were input into the algorithm, and it was found that the algorithm reproduced them exactly. Figure 2 shows the results from a test case in which values for a three segment array producing a deep bow effect were computed and input into the algorithm. The shape was correctly reproduced.

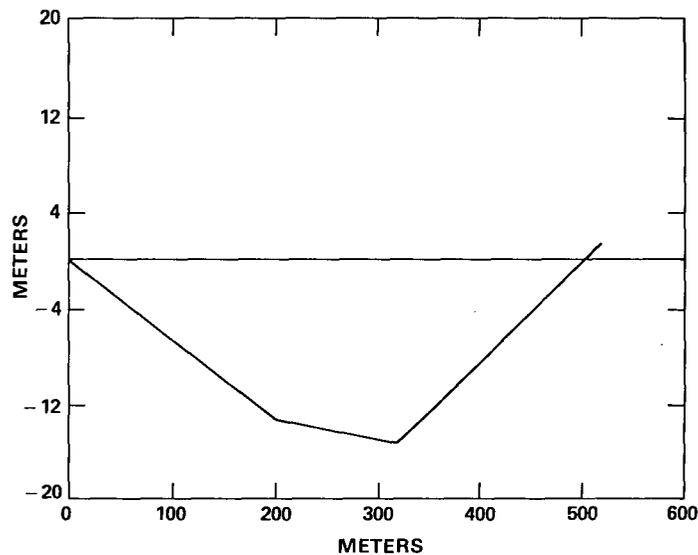


Fig. 2 — A test data case

For FREDDEX, depth sensors were placed at the front and the tail of the array, and it was found that depth variations were insignificant, i.e. the vertical tilt was less than 0.5°. Accordingly, in developing an algorithm to process FREDDEX data, depth could be ignored. In addition it was assumed that the array spacing was known and fixed and that the array was piecewise linear.

DATA PROCESSING

Digitizing and Time Delays

The next several sections discuss the implementation of the algorithm to the FREDDEX acoustic data. Figure 3 traces the processing and display steps from the analog acoustic tapes through the computation of array shape coordinates for the experiment. Figure 4 shows the flow of computations made from this file of array shape coordinates. In both cases, the program names are given along the solid lines. FORTRAN source listings for the major data processing programs are given in the appendix of this report.

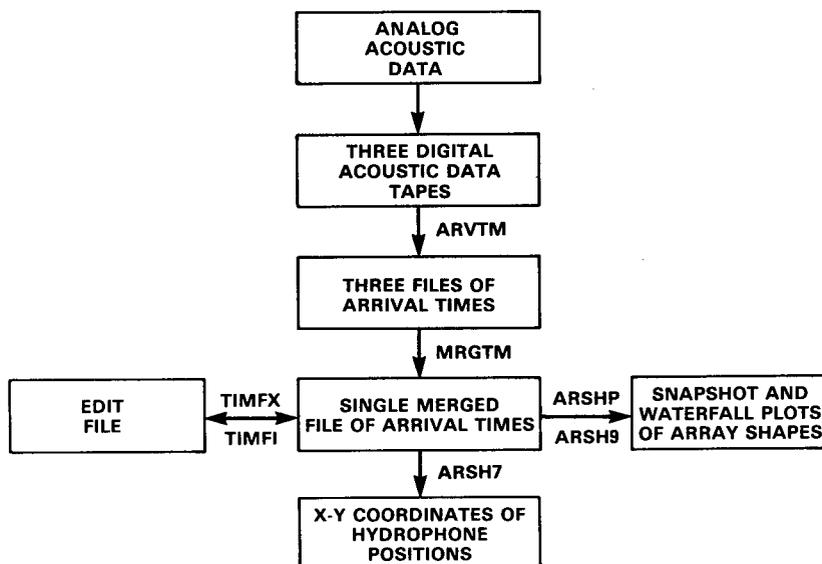


Fig. 3 — Flow chart of data processing to obtain array shape positions

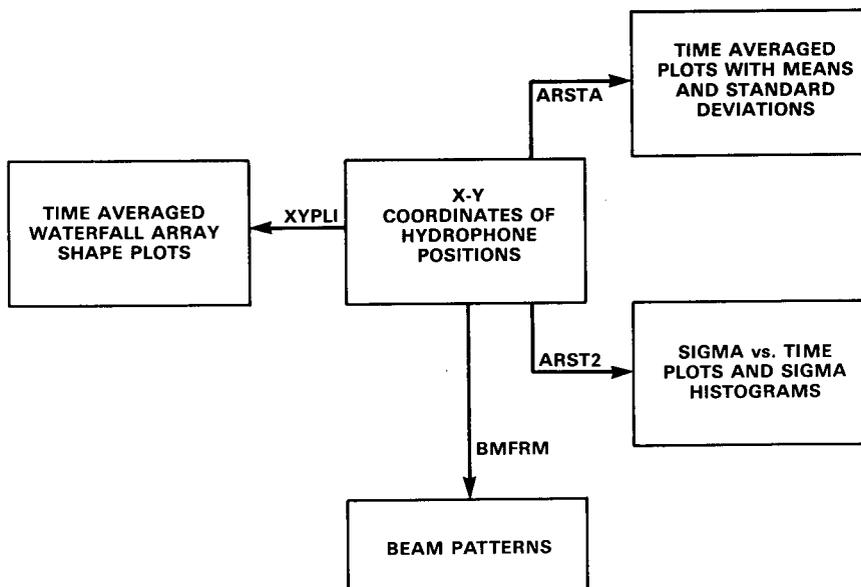


Fig. 4 — Flow chart of processing using file of array shape positions

Existing computer programs were used for the digitization and real-time display of collected analog acoustic data. Twenty-seven alternate channels of data spaced at 20 m were digitized at a sampling rate of 4096 samples per second. This produced eleven samples per cycle at the highest frequency of the pulse (375 Hz). Three digital data tapes were produced from the 6 hours of data for further processing. Figure 5 is an example of the real-time display used to monitor the digitizing process. Hydrophones 2, 15, and 27 are monitored and, from the time of arrival, one can see that the closest hydrophone to the source was between hydrophones 2 and 15 for this time slice. A digitizing window length of 125 ms per channel was chosen to capture the largest expected bow shape of the array. Hydrophone 15 was placed between 31.25 and 62.5 ms since the experiment attempted to align the two ships so that hydrophone 15 would be closest to the acoustic source.

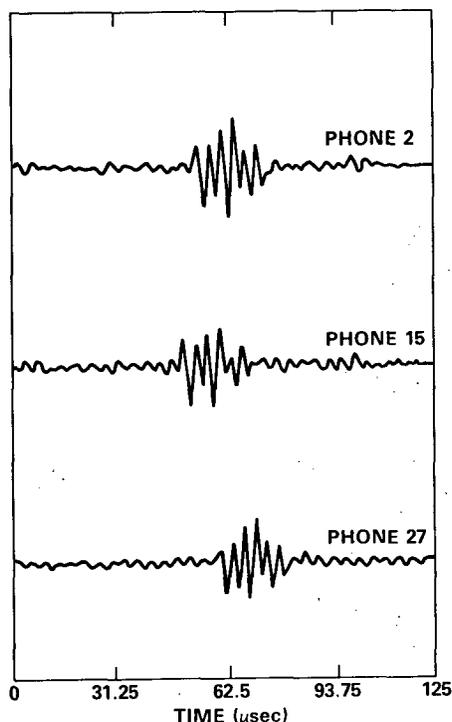


Fig.5 — Real-time digitization display

The next step was to produce time delays using program ARVTM. ARVTM read the digitized tape and computed time delays which were stored on a disc file for further processing. In order to eliminate transients which occurred during the digitizing process, any detection which occurred outside a two second window for the shot was not processed. For valid times the largest negative peak for each phone was found and a threshold was set at 1/2 the absolute value of this peak. Next, the index of the first negative value which (negatively) exceeded this threshold was determined, and the first positive value of greater index was located. The ping arrival time was then computed from the zero crossing of a linear interpolation between the two points. Once this had been done for all 27 hydrophones, the sequence of indices corresponding to each negative peak position was examined to assure that it had only one turning point. If an erroneous peak was located, the average of the indices of the peaks of the two surrounding hydrophones was used as an estimate of the peak and the zero crossing was recomputed. Finally, time delays were obtained for each hydrophone by using: 1) the header record to obtain the time delay to the start of the digitized data, 2) the index of the last negative point before the zero-crossing to obtain the time to this point from the start of the data, and 3) a linear interpolation between the last negative and first positive value to obtain the time of the zero crossing. This detection process proved quite successful allowing only a few instances of detection errors, almost all of which occurred in the first 90 minutes of data prior to the source level being increased.

Table 1 — Typical Time Delay Calculation

START TIME:	171	6	6	1			
PHONE	MAX	NEGPK	ZERO+	+ VAL	- VAL	TM DELAY	
1	8864	211	217	1408	-816	1.834874	
2	9856	207	212	3104	-112	1.832772	
3	9928	203	208	1568	-1216	1.831894	
4	8544	199	203	944	-2256	1.830739	
5	9008	196	201	128	-2752	1.830311	
6	10352	193	198	2528	-224	1.829366	
7	9608	190	196	2096	-112	1.828878	
8	9056	188	193	512	-2368	1.828326	
9	10800	187	191	480	-2480	1.827841	
10	7952	185	190	1840	-1456	1.827535	
11	10896	184	189	1984	-1568	1.827259	
12	8608	184	188	160	-3056	1.827136	
13	9440	183	188	640	-2432	1.827098	
14	10432	183	188	2368	-1152	1.826984	
15	8992	185	190	1552	-992	1.827488	
16	9648	185	191	2432	-256	1.827668	
17	7632	188	193	1792	-384	1.828168	
18	10272	190	194	464	-3088	1.828581	
19	7152	191	197	1792	-80	1.829112	
20	9952	195	201	1928	-728	1.830145	
21	9344	198	203	560	-2480	1.830766	
22	6592	202	209	1360	-128	1.832052	
23	8480	206	211	1488	-1152	1.832626	
24	10592	211	216	160	-2592	1.833970	
25	9488	216	221	1312	-1520	1.835092	
26	7808	223	228	1488	-832	1.836757	
27	8912	228	233	64	-2496	1.838129	

Table 1 illustrates the process of producing time delays for a fixed time slice (time = 171 6 6). For each hydrophone the largest negative peak was found. The first negative peak exceeding the threshold for each hydrophone was then computed and listed in the column labeled NEGPK in terms of its index within the time segment, i.e. from 1 to 512. The column labeled ZERO+ contains the index of the first positive value following the first negative peak and +VAL contains that value. The column labeled -VAL contains the value of the previous index. Interpolation between +VAL and -VAL then allowed accurate computation of the time delay as seen in column labeled TM DELAY. The time delays are then stored in a data file on disc for further processing.

Program MRGTM then merged the three time delay files into a single file for further processing. At the same time a further check was made to assure that, for a given time, the ping appeared within the proper time interval. Using estimated arrival times derived from the range and sound speed pings which were not detected between $1\frac{22}{32}$ s and $1\frac{27}{32}$ s after the quarter minute marks were eliminated.

Array Shapes

Next, time delays were used to produce array shapes using program ARSHP, which implements the algorithm previously described and produces plots of single scan array shapes as well as water fall plots of array shapes over user defined time periods. Using the measured sound speed of 1522 m/s, time delays were converted into distances and the array shape algorithm was applied, resulting in a set of x-y coordinates for each time slice. In order that successive shapes could easily be prepared and beamforming could be performed, a best-fit least-squares line was calculated for the shape at each fixed time and the x-y coordinates were translated and rotated by

$$\bar{x}_i = x_i \cos(\alpha) + (y_i - \beta) \sin(\alpha),$$

$$\bar{y}_i = -x_i \sin(\alpha) + (y_i - \beta) \cos(\alpha),$$

where α is the angle that the least squares fit line makes with the x-axis and β is the intercept with the y-axis. Plotting routines were then used to get individual snapshots of the array and to produce water-fall plots of array shape as a function of time. Figure 6 is an example of a single snapshot of the array.

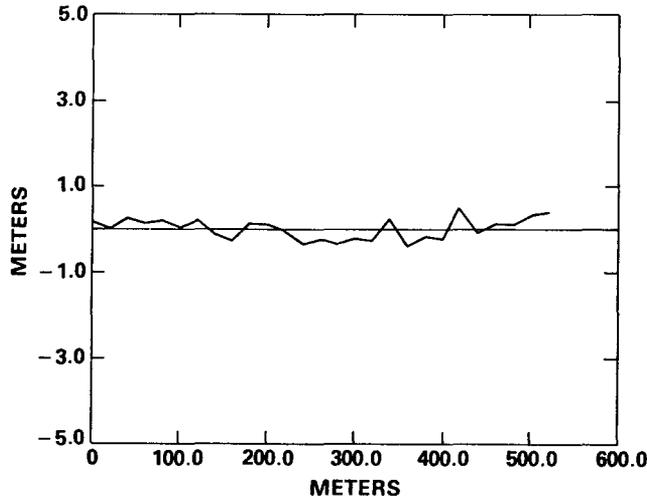


Fig. 6 — Typical array shape

This snapshot was taken about five hours into the experiment (at time: 171 6 0) and illustrates the hydrophone-to-hydrophone variability found by applying the algorithm without averaging. Larger variations, as seen at hydrophone 22, have often been found to be consistent over a small time interval, i.e. a variation of about 0.4 m from an expected position would occur for a 15- to 30-minute time period and the hydrophone would fall back into position. This deviation corresponds to an error of approximately one sample and is too small to be a peak detection problem. One possible explanation would be a jump of one position in the digitizer.

The next step in the processing was to correct for hydrophone-to-hydrophone phase delays. To do this, the section of data between 5:00 and 6:30 was selected because the averaged shape for this time period exhibited only linear characteristics. Time delays which would linearize the shape were computed for each hydrophone. Table 2 shows the absolute time delays and Table 3 shows the phase differences (relative to hydrophone 14). The average array for the entire seven hours of data is shown in Fig. 7. The smaller deviations noted around the 1/4 and 3/4 points on the array are accounted for by the predominance of the first order mode in the array. The slight bow in the array is due primarily to the initial 12 minutes of data when the ships were still maneuvering to correctly align themselves for the experiment. Next, program ARSH7 was used to process the time delay data file as previously discussed with the phone-to-phone phase corrections and to store the results in a new data file containing x - y coordinates of the array shapes over the entire data set. The waterfall plot program was then modified to work on the x - y coordinates. Figure 8 shows a waterfall plot of array shape. The shapes have been averaged in two minute intervals and the display contains one hour of data. The sinusoidal oscillation of the array as a function of time is easily seen with a period of about 16 minutes.

A very small number of bad data points still existed in the data set at this point for a variety of reasons. For example, when the sequence of indexes of negative peaks was searched for nonmonotonic values corresponding to false peak detection, no check was made to see if the minimum value was correctly detected. To eliminate these problems, hydrophones which differed by more than one meter from both surrounding hydrophones were replaced by the average y -coordinate value of the two surrounding hydrophones. This resulted in editing about 60 points out of a total of over 32,000. In addition, it was noted that on hydrophone 1, the same values would reoccur on occasion, once for 23 consecutive times. This was clearly due to some error in the digitization, most likely a problem with reading the first channel on the tape. In those instances where the values were identical for successive times, the first time delay was replaced by the extrapolated value obtained from hydrophone 2 and hydrophone 3. A check was made of all channels to see if this problem had occurred elsewhere, but it had not.

Table 2 — Phase Delays Relative to Hydrophone 14

FREQ PHONE	48.5	83.3	111.6	162.0	224.0	227.0	230.0	235.0	350.0
1	-1.1	-1.8	-2.5	-3.6	-4.9	-5.0	-5.1	-5.2	-7.7
2	-2.6	-4.4	-5.9	-8.6	-11.9	-12.1	-12.3	-12.5	-18.6
3	-2.7	-4.7	-6.3	-9.1	-12.6	-12.7	-12.9	-13.2	-19.7
4	-1.9	-3.2	-4.3	-6.2	-8.6	-8.7	-8.9	-9.1	-13.5
5	-5.0	-8.6	-11.5	-16.7	-23.1	-23.5	-23.8	-24.3	-36.2
6	-0.9	-1.6	-2.1	-3.1	-4.3	-4.3	-4.4	-4.5	-6.7
7	-3.0	-5.2	-6.9	-10.0	-13.9	-14.1	-14.2	-14.6	-21.7
8	-2.1	-3.6	-4.9	-7.1	-9.8	-9.9	-10.0	-10.2	-15.2
9	-2.5	-4.3	-5.7	-8.3	-11.5	-11.7	-11.8	-12.1	-10.0
10	-4.3	-7.4	-10.0	-14.5	-20.0	-20.3	-20.5	-21.0	-31.2
11	-3.9	-6.6	-8.9	-12.9	-17.8	-10.1	-10.3	-10.7	-27.0
12	-3.2	-5.5	-7.4	-10.7	-14.8	-15.0	-15.2	-15.5	-23.1
13	-2.7	-4.6	-6.1	-8.9	-12.3	-12.5	-12.7	-12.9	-19.3
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	-2.3	-3.9	-5.2	-7.6	-10.5	-10.6	-10.8	-11.0	-16.4
16	-0.8	-1.4	-1.9	-2.7	-3.8	-3.8	-3.9	-4.0	-5.9
17	-1.8	-3.1	-4.1	-5.9	-8.2	-8.3	-8.4	-8.6	-12.9
18	-4.3	-7.5	-10.0	-14.5	-20.1	-20.3	-20.6	-21.1	-31.4
19	-0.9	-1.5	-2.0	-3.0	-4.1	-4.2	-4.2	-4.3	-6.4
20	-3.0	-5.2	-7.0	-10.1	-14.0	-14.1	-14.3	-14.6	-21.8
21	-0.3	-0.5	-0.7	-1.0	-1.5	-1.5	-1.5	-1.5	-2.3
22	-4.3	-7.4	-9.9	-14.4	-19.9	-20.2	-20.5	-20.9	-31.1
23	-1.8	-3.1	-4.2	-6.1	-8.5	-8.6	-8.7	-8.9	-13.2
24	-3.0	-5.2	-7.0	-10.1	-14.0	-14.2	-14.4	-14.7	-21.9
25	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1
26	-4.8	-8.2	-11.0	-16.0	-22.1	-22.4	-22.7	-23.2	-34.5
27	-3.0	-5.1	-6.8	-9.9	-13.7	-13.9	-14.1	-14.4	-21.4

Table 3 — Absolute Time Delays (ms) for each Hydrophone

1	2	3	4	5	6	7	8	9
0.000	-0.007	-0.015	0.034	-0.146	0.031	-0.031	0.020	-0.002
10	11	12	13	14	15	16	17	18
-0.107	-0.080	-0.042	-0.012	0.141	0.011	0.094	0.039	-0.100
19	20	21	22	23	24	25	26	27
0.090	-0.032	-0.123	-0.106	0.036	-0.033	0.148	-0.133	-0.029

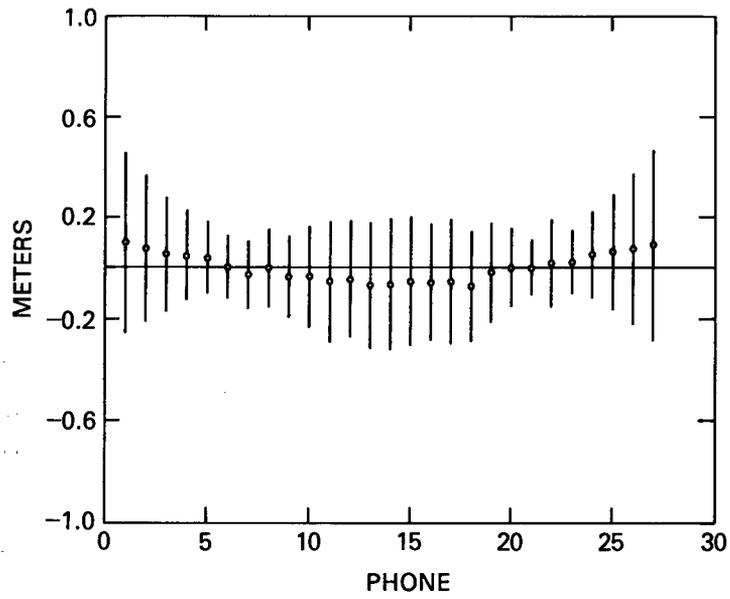


Fig. 7 — Array average for the entire event

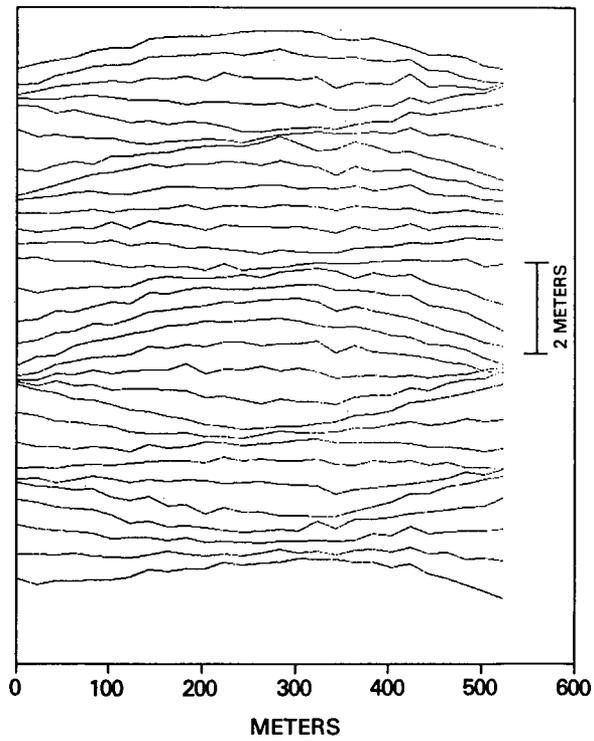


Fig. 8 — Waterfall display of two-minute averaged data

Statistics

Now that the array positions were determined, it was possible to look at array statistics and to do beamforming. First, throughout the processing it had been assumed that the array elements were 20 m apart. Program XCHK1 examined the computed x positions (after rotation) and verified that variations from multiples of 20 were indeed insignificant.

Next, program ARSTA computed the mean and standard deviation (sigma) for the array averaged over arbitrary (user defined) time intervals. Figures 9a and 9b are examples of this type of plot. Sixteen minutes of data centered about 171 3 0 are seen. The full oscillation of the bow shape is again observed. The maximal bow of approximately 1 m seen at both 171 2 58 and 171 3 4 is typical. To further examine the array variability, plots were made of sigma as a function of time for user input averaged intervals. Figure 10 shows this type of plot for a two-minute average. Sigma is seen to run at about 0.2 m except at times when the array has maximal sinusoidal peaks. The gaps in Figure 10 occur at times when the acoustic source was not operational. Figure 11 shows plots of sigma bins vs number of occurrences for bin size 0.02 m. This plot shows that 0.2 m is the typical sigma over the experiment. Since 0.2 m is small compared to the shortest wavelength in FREDDEX (approximately 4 m), it is clear that the array deformation caused by towing had an insignificant acoustic effect.

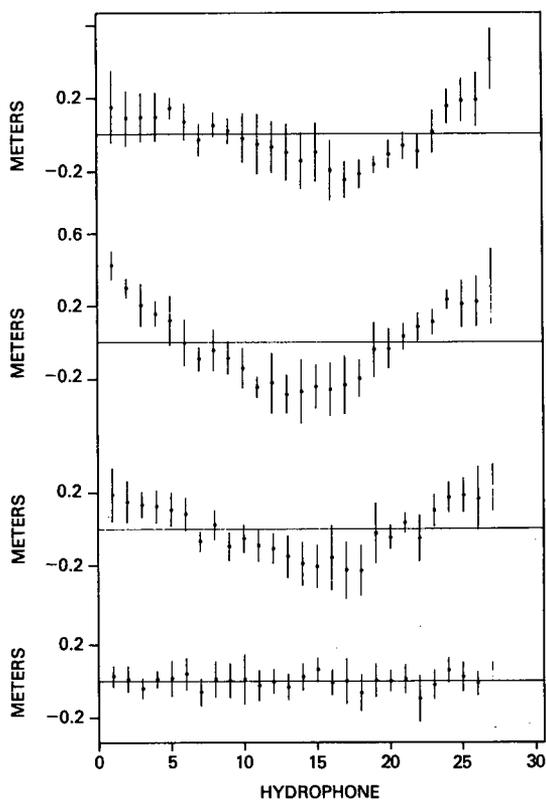


Fig. 9a — Two-minute averages of array shapes

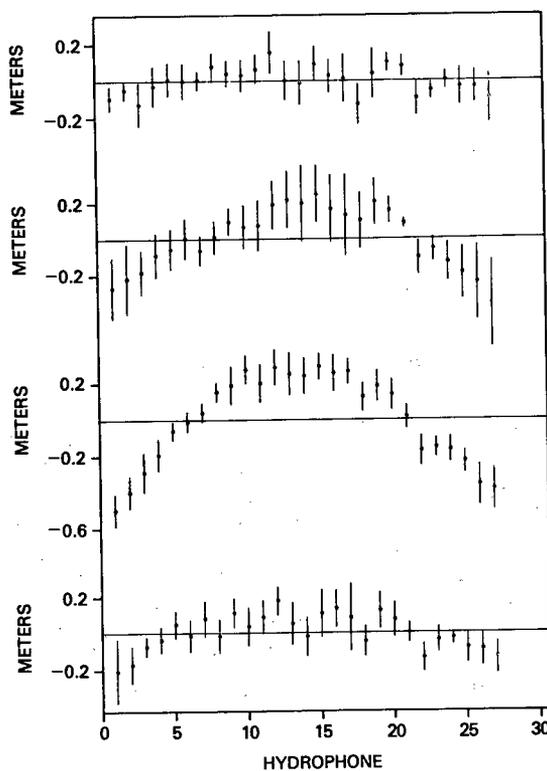


Fig. 9b — Two-minute averages of array shapes

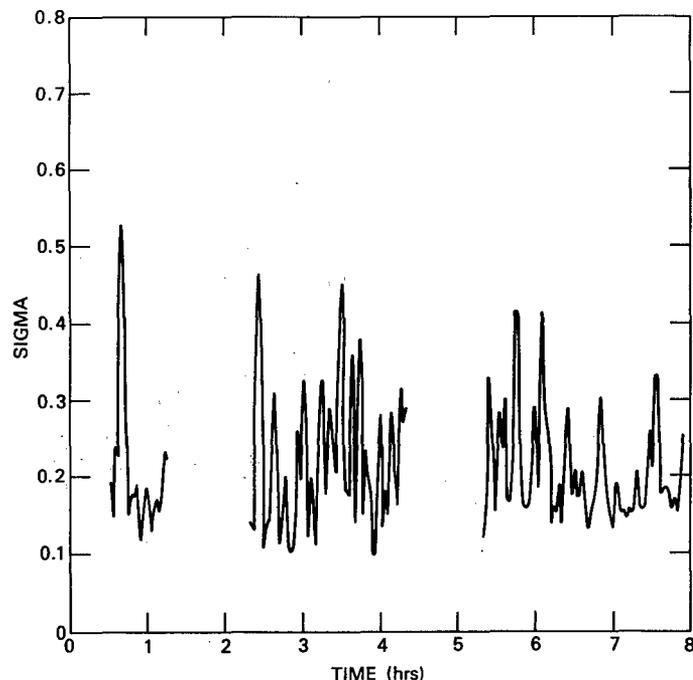


Fig. 10 — Sigma of two-minute averages over the entire event

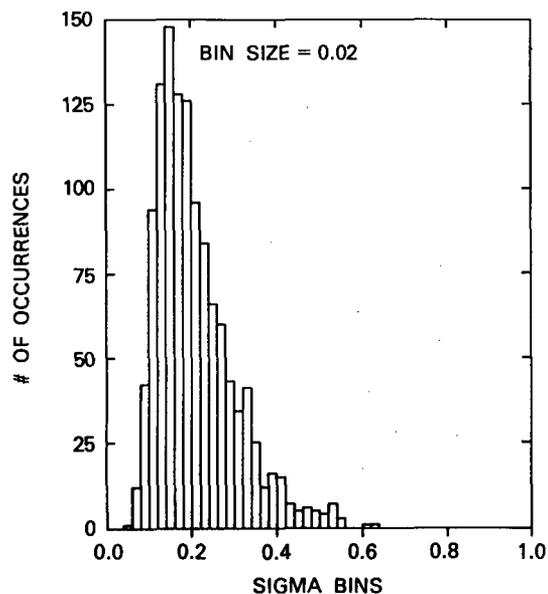


Fig. 11 — Distribution of sigma for a bin size of 0.02 m

Beamforming

Beam patterns were produced from the file of x-y coordinates by program BMFRM. Beam patterns, averaged over any user defined time intervals, were computed for arbitrary (user defined) angles of arrival (ϕ) and frequency. The 27-hydrophone subarray used to compute shapes was both interpolated and extrapolated to reproduce the original 60 hydrophone array spaced 10 m apart. A Hanning window was applied and a complex FFT was taken. The array was then reordered (since the FFT results are not ordered by angle) and normalized and the power was taken to compute the beam pattern. The 3-dB down points and width were then computed and a plot of computed and nominal beam patterns produced. Figure 12 shows the average beam pattern for the entire test computed for a perpendicular arrival path ($\phi = 0^\circ$) and a frequency of 227 Hz. Figure 13 shows a typical beam pattern over a 15-minute average (start time: 171 3 0) for $\phi = 0^\circ$ and a frequency of 83 Hz. Figure 14 shows a one-hour averaged beam pattern for 362 Hz (start time: 171 3 0). In these figures, the actual array represents the beam pattern computed for the previously obtained array shapes and then averaged over the stated time interval. The linear array beam pattern denotes the beam pattern that would be obtained if there were no spatial distortion of the array, i.e. the array is linear. As would be expected from the minimal spatial distortion which the array was found to have during towing, no significant degradation of array performance is noted. In Fig. 12 the array has a peak degradation of 0.11 dB and the 3-dB down beamwidth is 0.92 degrees. In Fig. 13 the peak degradation is 0.013 dB and the 3-dB down beamwidth is 2.51 degrees. Figure 14 has a peak degradation of 0.29 dB and 3-dB beamwidth of 0.59 degrees. Large increases in sidelobe levels are seen in these figures; however, these sidelobes are more than 35 dB down and thus will have no effect upon the acoustic analysis problem.

CONCLUSIONS

1. The algorithm worked very successfully on the FREDDEX data set. Small perturbations in the array were successfully reconstructed by the algorithm. The high-source levels used during the

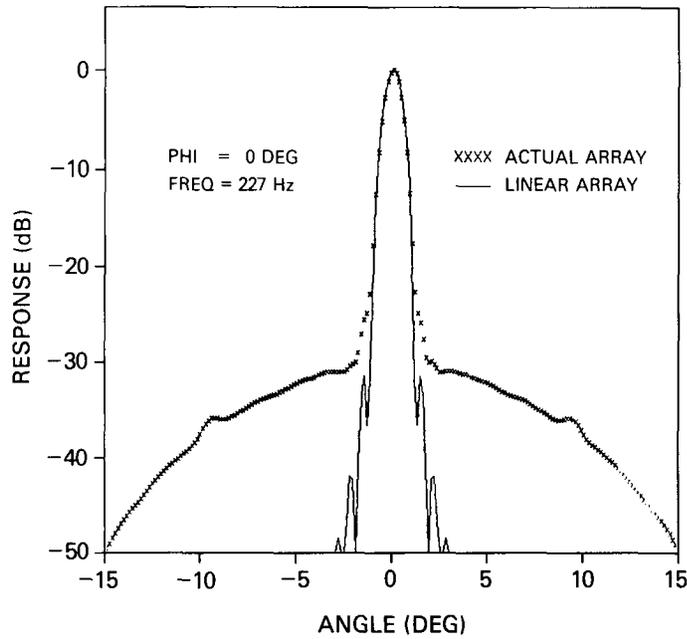


Fig. 12 — Averaged beam pattern at 227 Hz for the entire event

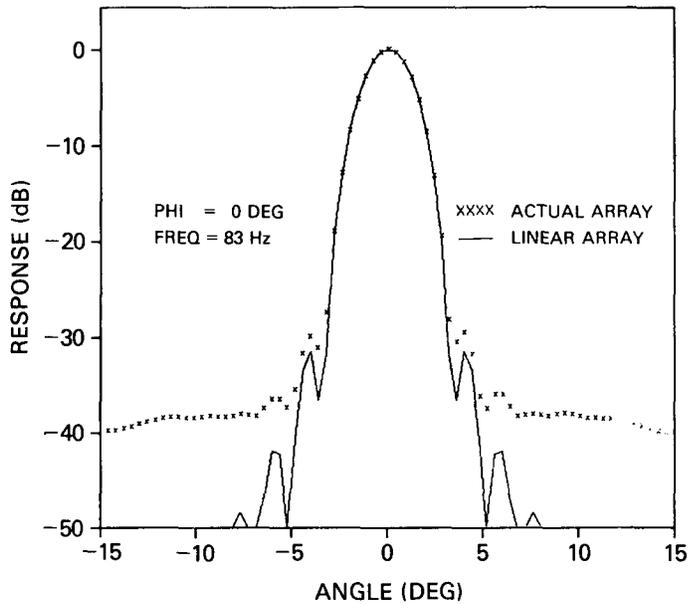


Fig. 13 — Averaged beam pattern at 83 Hz (15-minute average)

experiment were essential to allow the accurate computation of arrival times required by the algorithm. Interpolation of the zero crossing was required to compute the arrival time with sufficient accuracy for the algorithm. Also the range was close enough and the source and receiver depths deep enough to resolve direct and surface reflected paths.

2. For FREDDEX, there was little array deformation. The computed typical deviation of 0.2 m is small compared to the shortest wavelength in FREDDEX of approximately 4 m. These results are not typical since the array was being towed at the rather high speed of 5 knots. However, the algorithm would be expected to work well for data with greater array deformations.

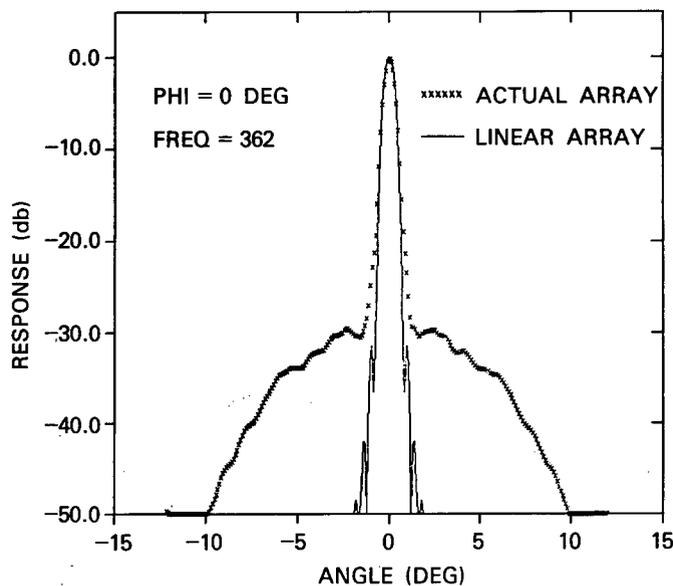


Fig. 14 — Averaged beam pattern at 362 Hz (one-hour average)

3. The lowest spatial mode (the bow shape) cycles is about 16 minutes. No significant effect from higher order modes was seen.

4. Since two ships are needed, current techniques for acquiring this type of data are very expensive. Other potential techniques such as a distribution of compasses along the array or using a far-field beacon source for bearing and shape computation merit investigation.

ACKNOWLEDGMENTS

The FREDDEX experiment was an outgrowth of the Mesoscale Environmental Effects on ASW Working Group, which was established by the Chief of Naval Material in October, 1978. The cooperative spirit of the participants and the diverse talents brought together are a direct result of NAVMAT stimulating and initiating this study. In particular, Glenn Spalding is acknowledged for his support of the exercise and the follow-on data analysis.

The NRL Ship Facility Group and the scientific and operational staffs aboard the USNS HAYES, USNS LYNCH, R/V ENDEAVOR, and the NRL and NAVOCEANO aircraft are acknowledged for their highly professional and competent activities throughout the FREDDEX experiment. CINCLANTFLT, COMSUBLANT, COSL, and NRL's Operational Services Group are commended for their efforts in securing the many area clearances.

Use of the array was provided by ARPA and this contribution is greatly appreciated. Mr. Gordon Cooke of NOSC is especially recognized for his competence in pretest preparation activities and his expert advice in deploying, handling, and retrieving the array.

Special appreciation is extended to Dr. B. Adams and Mr. D. Steiger for their valuable suggestions during the preparation of this manuscript and to Mr. Mark Feeney for his assistance in processing the FREDDEX data. The work was sponsored by the Naval Electronics Systems Command.

REFERENCE

1. R.L. Martin, "Techniques for Measuring the Ship of Long Arrays— A Summary," IEEE Regional Meeting in Washington, D.C., Oct. 79.

APPENDIX

This appendix contains FORTRAN source listings of the major programs used in the data processing. The reader should refer to Figs. 3 and 4 where flow charts of the programs and their functions are given. The programs were written to operate on a Digital Equipment Corporation (DEC) 11/45 computer system operating under RSX-11M.

```

..
C      PROGRAM ARVYM
C
C      THIS PROGRAM WILL PROCESS THE DIGITIZED DATA TAPES AND
C      PRODUCE A FILE OF TIME DELAYS FOR EACH PHONE.  A SECOND FILE
C      WITH MAX AND MIN FOR EACH PHONE WILL ALSO BE PRODUCED.
C      THE ARRIVAL TIME IS DEFINED AS THE 1ST ZERO CROSSING
C      AFTER THE 1ST LARGE NEGATIVE PEAK.  THIS IS DETERMINED
C      BY FINDING THE MAXIMUM NEGATIVE VALUE AND THEN THRESH-
C      HOLDING AT 1/2 THIS VALUE.  INTERPOLATION IS PERFORMED
C      TO LOCATE THE ZERO CROSSING AND SOME CHECKS ARE MADE
C      TO ALLEVIATE PROBLEMS DUE TO TRANSIENTS AND BAD MEASURE-
C      MENTS.
C
C      PROGRAMMER: L. ROSENBLUM    VERSION: 4-11-80
C
C      RECORD STRUCTURE FOR OUTPUT FILE TIMDL
C      WORD 1          JULIAN DAY
C      WORD 2          HOUR
C      WORD 3          MINUTE
C      WORD 4          SECOND
C      WORD 5/WORD 6  FRACTION OF SECOND
C      WORDS 7-60     TIME DELAYS FOR PHONES 1-27
C
C      RECORD STRUCTURE FOR OUTPUT FILE MAXVL
C      WORDS 1-6      TIME AS ABOVE
C      WORDS 7-36     MAX VALUES FOR PHONES 1-30
C      WORDS 34-66    MIN VALUES FOR PHONES 1-30
C
C      DIMENSION IBUF(512), ISTOP(3), IMAX(30), IPOS(30), ITIM(6)
C      DIMENSION ICHK(30), IPOS1(30), IZERP(30), IZERN(30), TDEL(30)
C      DIMENSION IARR1(60), IARR0(66), IMAX1(30), ITIM2(6)
C      EQUIVALENCE (IARR1(1), ITIM(1)), (IARR1(7), TDEL),
C      I (IARR0(1), ITIM2(1)), (IARR0(7), IMAX1(1)), (IARR0(37), IMAX(1))
C      DATA ICONS, LUOUT/5.5/
C      GET INPUT PARAMETERS
C      TYPE *, ' ENTER STOP TIME: HOUR, MIN, SEC '
C      ACCEPT *, ISTOP(1), ISTOP(2), ISTOP(3)
C      TYPE *, ' ENTER MAG TAPE UNIT #'
C      ACCEPT *, IUNIT
C      SET UP OUTPUT FILES
C      CALL ASSIGN(1, 'SY:TIMDL.TP3')

```

ROSENBLUM AND DEL BALZO

```

DEFINE FILE 1 (500,60,U,IPNT1)
CALL ASSIGN(3,'SY:MAXVL,TP3')
DEFINE FILE 3 (500,66,U,IPNT0)
IPNT1=1
IPNT0=1
CALL ASSIGN(2,'SY:LR0UT')
C   READ THE HEADER FILE
10 CALL MTREAD(IBUF,512,IER,0,IUNIT,NUMB)
DO 5 I=1,6
ITIM(I)=IBUF(I+6)
5 CONTINUE
TYPE *, ' TIME: ',(ITIM(J),J=1,4)
C   SKIP TIMES WHEN ARRAY WAS OUT OF WATER OR ON WAY IN OR OUT.
IF(IBUF(8).EQ.1.AND.IBUF(9).GT.16)GO TO 22
IF(IBUF(8).EQ.2.AND.IBUF(9).LT.19)GO TO 22
IF(IBUF(8).EQ.4.AND.IBUF(9).GT.20) GO TO 22
IF(IBUF(8).EQ.5.AND.IBUF(9).LT.6) GO TO 22
C   CHECK FOR STOP TIME
IF(IBUF(8).LT.ISTOP(1))GO TO 15
IF(IBUF(8).EQ.ISTOP(1).AND.IBUF(9).LT.ISTOP(2))GO TO 15
IF(IBUF(8).EQ.ISTOP(1).AND.IBUF(9).EQ.ISTOP(2).AND.
1  IBUF(10).LT.ISTOP(3))GO TO 15
IF(IBUF(7).EQ.171)GO TO 99
CALL SKIPF(IUNIT,1)
GO TO 10
C   CHECK THAT PING FALLS IN PROPER SECOND - IF NOT REJECT FILE
15 DO 20 I=1,60,15
J=I+2
IF(IBUF(10).GE.I.AND.IBUF(10).LE.J)GO TO 30
20 CONTINUE
22 WRITE(2,25)(IBUF(J),J=7,10)
25 FORMAT(1X,' TIME = ',4I4,2X,' NOT IN BOUNDS')
C   GO TO NEXT TIME HEADER
CALL SKIPF(IUNIT,1)
CALL MTWAIT(IER)
GO TO 10
C   READ IN RECORD FROM TAPE
30 DO 100 I=1,30
IMAX(I)=0
CALL MTREAD(IBUF,512,IER,0,IUNIT,NUMB)
C   FIND MINIMUM VALUE FOR EACH PHONE
DO 40 J=1,512
IF(IBUF(J).GE.0)GO TO 40
ITEST=IABS(IBUF(J))
IF(ITEST.GT.IMAX(I))IPEAK=J
IF(ITEST.GT.IMAX(I))IMAX(I)=ITEST
40 CONTINUE
C   FIND POS MAX FOR FILE MAXVL
IMAX1(I)=0
DO 45 J=1,512
IF(IBUF(J).GT.IMAX1(I))IMAX1(I)=IBUF(J)
45 CONTINUE
C   SET DETECTION LEVEL AND FIND 1ST NEGATIVE BELOW LEVEL
IDTCT= -ICONS*(IMAX(I)/10)
IF(IPEAK.LT.101)WRITE(LUOUT,48)I
48 FORMAT(1X,'PEAK VALUE FOUND WITHIN 1ST 100 SAMPLES PHONE:',I2)
IF(IPEAK.LT.101)IPEAK=101
DO 50 J=IPEAK-100,IPEAK
IF(IBUF(J).LT.IDTCT)GO TO 60

```

```

50 CONTINUE
   IF(J.EQ.512)WRITE(LUOUT,55)I
55 FORMAT(1X,'NO NEGATIVE VALUE BELOW THRESHHOLD - PHONE ',I2)
   GO TO 100
C     FIND 1ST NEGATIVE PEAK
60 DO 70 K=1,512
   IF(J+K.GT.512)GO TO 80
   IF(IBUF(J+K-1).LT.IBUF(J+K))GO TO 80
70 CONTINUE
80 IPOS(I)=J+K-1
C     FIND ZERO CROSSING
   L=IPOS(I)
90 L=L+1
   IF(L.GT.500)GO TO 100
   IF(IBUF(L).LT.0)GO TO 90
   IPOS1(I)=L
   IZERP(I)=IBUF(L)
   IZERN(I)=IBUF(L-1)
100 CONTINUE
C     CHECK SEQUENCE FOR BAD MAX OR MIN
C     IF FIND ONE GO BACK OR FORWARD 11*N SAMPLES
   DO 140 I=1,27
   ICHK(I)=0
140 CONTINUE
   DO 150 I=2,26
   ITRY1=IPOS(I-1)-IPOS(I)
   ITRY2=IPOS(I)-IPOS(I+1)
   IF(ITRY1.GE.0.AND.ITRY2.GE.0)GO TO 150
   IF(ITRY1.LE.0.AND.ITRY2.LE.0)GO TO 150
   IF(IABS(ITRY1).LE.10.OR.IABS(ITRY2).LE.10)GO TO 150
   IPOS(I)=(IPOS(I-1)+IPOS(I+1))/2
   ICHK(I)=1
150 CONTINUE
C     SEE IF ANY LOCAL MINIMA CHANGED
   DO 152 I=1,27
   IF(ICHK(I).NE.0)GO TO 154
152 CONTINUE
   GO TO 165
C     REPOSITION TAPE TO HEADER RECORD
154 CALL SKIPF(IUNIT,0)
   CALL MTWAIT(IER)
C     REFINDE LOCAL MINIMA FOR THOSE CHANGED ABOVE
   DO 160 I=1,27
   CALL SKIPB(IUNIT,1)
   CALL MTWAIT(IER)
   IF(ICHK(I).EQ.0)GO TO 160
   JTEMP=1
   TYPE *, ' REFINDE MINIMA PHONE # ',I
   CALL MTREAD(IBUF,512,IER,0,IUNIT,NUMB)
155 IX=IBUF(IPOS(I))
   IXM1=IBUF(IPOS(I)-1)
   IXP1=IBUF(IPOS(I)+1)
   IF(IX.LE.IXM1.AND.IX.LE.IXP1)GO TO 159
   IF(IX.GT.IXM1)IPOS(I)=IPOS(I)-1
   IF(IX.LT.IXM1.AND.IX.GT.IXP1)IPOS(I)=IPOS(I)+1
   IF(IPOS(I).LE.50.OR.IPOS(I).GE.450)GO TO 159
   TYPE *, ' PHONE # ',I, ' ITERATION # ',JTEMP

```

```

JTEMP=JTEMP+1
IF(JTEMP.GT.100)GO TO 159
GO TO 155
C   REFIND ZERO CROSSING
.. 159 L=IPOS(I)
158 L=L+1
    IF(L.GT.500)GO TO 160
    IF(IBUF(L).LT.0)GO TO 158
    IPOS1(I)=L
    IZERP(I)=IBUF(L)
    IZERN(I)=IBUF(L-1)
160 CONTINUE
C   CONVERT SEQUENCE FROM POSITIONS TO ARRIVAL TIMES
165 ISEC1=ITIM(4)-15*(ITIM(4)/15)
    SEC=FLOAT(ISEC1)+FLOAT(ITIM(5))/FLOAT(ITIM(6))
    DO 170 I=1,29
        TDEL1=FLOAT(IPOS1(I)-1)/4096.
        IF(IZERN(I).EQ.0.AND.IZERP(I).EQ.0)GO TO 170
        TDEL2=FLOAT(-IZERN(I))/(FLOAT(IZERP(I))-FLOAT(IZERN(I)))
        TDEL2=TDEL2/4096.
        TDEL(I)=SEC+TDEL1+TDEL2
170 CONTINUE
C   DEFINE ITIM2
    DO 175 J=1,6
        ITIM2(J)=ITIM(J)
175 CONTINUE
C   WRITE ONE OUTPUT RECORD TO LP EVERY TWO MINUTES
    ITST=ITIM(3)-2*(ITIM(3)/2)
    IF(ITST.NE.0.OR.ITIM(4).GE.15)GO TO 985
    WRITE(2,994)(ITIM(J),J=1,4)
994 FORMAT(1X,'START TIME: ',4I4)
    WRITE(2,991)
991 FORMAT(1X,2('PHONE',4X,'MAX',2X,'NEGPK',2X,'ZERO+',2X,'+ VAL',
1      2X,'- VAL',2X,'TM DELAY',6X))
    DO 996 I=1,15
        WRITE(2,995)I,IMAX(I),IPOS(I),IPOS1(I),IZERP(I),IZERN(I),TDEL(I)
1, I+15,IMAX(I+15),IPOS(I+15),IPOS1(I+15),IZERP(I+15),IZERN(I+15),
2 TDEL(I+15)
995 FORMAT(1X,2(6(I5,2X),F8.6,6X))
996 CONTINUE
    WRITE(2,990)
990 FORMAT(1X,' ')
985 WRITE(1' IPNT1)(IARR1(J),J=1,60)
    WRITE(3' IPNT0)(IARR0(J),J=1,60)
C   SKIP OVER EOF
    CALL SKIPF(IUNIT,1)
    CALL MTWAIT(IER)
C   GO TO START OF NEXT RECORD
    GO TO 10
C   TYPE READ ERROR MESSAGE ON TERMINAL
110 TYPE *,' READ ERROR '
C   CLOSE FILES AND SPOOL OUTPUT
99 CALL CLOSE(1)
    CALL CLOSE(3)
    CALL SPOOL(2)
.. END

```

```

C PROGRAM MRGTM
C
C THIS FILE WILL EDIT THE THREE FILES TIMDL.TPN
C N=1,2,3 FOR BAD POINTS AND WILL MERGE THE RESULT
C INTO FILE TIMDL.ALL. FORMAT FOR TIMDL.ALL:
C WORD # 1 RECORD #
C 2-7 TIME (DAY,HR,MIN,SEC,NUMER,DENOM)
C 8-61 TIME DELAYS
C
C PROGRAMMER: L. ROSENBLUM VERSION: APRIL 23, 1980

```

```

C DIMENSION IBUF(60),IBUF1(61)
C EQUIVALENCE (IBUF1(1),IREC),(IBUF1(2),IBUF(1))
C IREC=1
C CALL ASSIGN(2,'SY:TIMDL.ALL')
C DEFINE FILE 2 (1289,61,U,IPNT1)
C IPNT1=1
C DO 100 ITP=1,3
C IF(ITP.EQ.1)CALL ASSIGN(1,'SY:TIMDL.TP1')
C IF(ITP.EQ.2)CALL ASSIGN(1,'SY:TIMDL.TP2')
C IF(ITP.EQ.3)CALL ASSIGN(1,'SY:TIMDL.TP3')
C DEFINE FILE 1(500,60,U,IPNTR)
C IPNTR=1
C IF(ITP.EQ.1)NUMB=396
C IF(ITP.EQ.2)NUMB=444
C IF(ITP.EQ.3)NUMB=449
C DO 50 I=1,NUMB
C READ(1'IPNTR,ERR=99)(IBUF(J),J=1,60)
C IF(IBUF(5).LT.22.OR.IBUF(5).GT.27)TYPE *,(IBUF(J),J=1,6)
C IF(IBUF(5).LT.22.OR.IBUF(5).GT.27)GO TO 50
C IPNT1=IREC
C WRITE(2'IPNT1)(IBUF1(J),J=1,61)
C IREC=IREC+1
C 50 CONTINUE
C 99 CALL CLOSE(1)
C 100 CONTINUE
C CALL CLOSE(2)
C ..
C END

```

```

C PROGRAM ARSH7
C
C THIS PROGRAM WILL COMPUTE X,Y COORDINATES FOR THE ARRAY
C POSITIONS AT GIVEN TIMES AND STORE THE RESULTING
C POSITIONS IN THE FILE XYPOS.ALL. TIME DELAYS ARE READ
C FROM THE FILE TIMDL.ALL. THIS PROGRAM IS A MODIFICATION
C OF THE WATERFALL PLOT PROGRAM ARSH9.
C
C FILE TIMDL.ALL HAS THE FOLLOWING RECORD STRUCTURE
C WORD 1 RECORD NUMBER
C WORDS 2-7 TIME(DAY,HR,MIN,SEC,NUMER,DENOM)
C WORDS 8-61 27 TIME DELAYS
C
C FILE XYPOS.ALL HAS THE FOLLOWING RECORD FORMAT
C WORD 1 RECORD #
C WORDS 2-7 TIME(DAY,HR,MIN,SEC,NUMER,DENOM)
C WORDS 8-61 27 X-POSITIONS
C WORDS 62-115 27 Y-POSITIONS

```

ROSENBLUM AND DEL BALZO

```

C
C   PROGRAMMER: L.ROSENBLUM  VERSION: 5-5-80
C
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  REAL TDEL,X2,Y2
  DIMENSION D(27),A(27),THETA(26),THET1(26),PHI(27)
  DIMENSION PSI(27),PHI1(27),X(27),Y(27),X1(27),Y1(27)
  DIMENSION ITIM(6),TDEL(27),IBUF(61),PHNDL(30),IBUF1(115),X2(27)
  DIMENSION Y2(27)
  EQUIVALENCE (IBUF(2),ITIM(1)),(IBUF(8),TDEL(1)),(IBUF1(8),X2(1)),
1      (IBUF1(62),Y2(1))
  DATA SNDSP,DBET,A(1),PHI(1),PHI(2)/1522.,20.,0.,0.,0./
  DATA PI,YMAX1,SYSDL,PLTY/3.1415926,5.0,-17.0,1.0/
  DATA PHNDL/.080,-.007,-.015,.034,-.146,.088,-.031,.020,-.002,
1      -.107,-.088,-.042,-.012,.141,.011,.094,.039,-.108,
2      .090,-.032,.123,-.106,.036,-.033,.140,-.133,-.029,
3      .0,0.0,0.0/
  A(2)=DBET
  TYPE*, ' ENTER START RECORD #'
  ACCEPT *, INUM0
  TYPE *, ' ENTER # OF RECORDS FOR PROCESSING '
  ACCEPT *, INUM1
C   SET UP FILES
  CALL ASSIGN(10,'SY:TIMDL.ALL')
  DEFINE FILE 10 (1289,61,U,IPNTR)
  CALL ASSIGN(11,'SY:XYPOS.ALL')
  DEFINE FILE 11 (1275,115,U,IPNT1)
  IPNTR=INUM0
  IPNT1=1
  DO 500 II=1,INUM1
C   READ A RECORD
  READ(10,IPNTR,ERR=99)(IBUF(J),J=1,61)
C   DO NOT PROCESS BAD RECORDS
  IF (II.EQ.68.OR.II.EQ.178.OR.II.EQ.191.OR.II.EQ.676.OR.
1  II.EQ.831.OR.II.EQ.1206)GO TO 500
C   ADJUST FOR SYSTEM DELAYS AND SINGLE PHONE DELAYS
  ITST1=II-(50*(II/50))
  IF (ITST1.EQ.0)TYPE *, ' REC # ',II
  DO 27 I=1,27
C   ADD PHONE TO PHONE CORRECTIONS
  TDEL(I)=TDEL(I)+(SYSDL/1000.0)+(PHNDL(I)/1000.0)
27 CONTINUE
C   CONVERT FROM TIME DELAYS TO DISTANCES
  DO 25 I=1,27
  D(I)=1522.*TDEL(I)
25 CONTINUE
C   GET THETA'S
  DO 20 I=1,26
  TEMP=D(I)*D(I)+D(I+1)*D(I+1)-DBET*DBET
  TEMP=TEMP/(2.0*D(I)*D(I+1))
  THETA(I)=DACOS(TEMP)
20 CONTINUE
C   GET SUM OF THETAS
  THET1(1)=THETA(1)
  DO 30 I=2,26
  THET1(I)=THETA(I)+THET1(I-1)
30 CONTINUE

```

```

C      GET A'S
      DO 40 I=3,27
      A(I)=D(I)*D(I)+D(I)*D(I)-2.0*D(I)*D(I)*COS(THET1(I-1))
      A(I)=SQRT(A(I))
40 CONTINUE
C      GET PHI
      DO 50 I=3,27
      TEMP=A(I-1)*A(I-1)+A(I)*A(I)-DBET*DBET
      TEMP=TEMP/(2.0*A(I-1)*A(I))
      PHI(I)=DACOS(TEMP)
50 CONTINUE
C      GET SIGN PHI
      DO 60 I=3,27
      PSI(I)=DASIN((D(I)*SIN(THET1(I-1)))/A(I))
      TEMP=SIN(PI-PSI(I)-THET1(I-2))
      YTST=((D(I)*SIN(PSI(I)))/TEMP)-D(I-1)
      IF(YTST.LT.0.0)PHI(I)=-PHI(I)
60 CONTINUE
C      GET SUM OF PHI'S
      PHI1(I)=0.0
      DO 90 I=2,27
      PHI1(I)=PHI1(I-1)+PHI(I)
90 CONTINUE
C      PRINT OUTPUT
      DO 70 I=1,27
      OUT1=(180.0*PHI(I))/PI
      OUT2=(180.0*PHI1(I))/PI
70 CONTINUE
      DO 105 I=1,27
      X(I)=A(I)*COS(PHI1(I))
      Y(I)=A(I)*SIN(PHI1(I))
105 CONTINUE
C      DO LEAST SQUARES FIT FOR Y AND GET SLOPE
      XAVG=0.0
      YAVG=0.0
      XIXI=0.0
      XIYI=0.0
      DO 150 I=1,27
      XAVG=XAVG+X(I)
      YAVG=YAVG+Y(I)
      XIXI=XIXI+X(I)*X(I)
      XIYI=XIYI+X(I)*Y(I)
150 CONTINUE
      XAVG=XAVG/27.0
      YAVG=YAVG/27.0
      XIXI=XIXI/27.0
      XIYI=XIYI/27.0
      SLOPE=(XIYI-XAVG*YAVG)/(XIXI-XAVG*XAVG)
      ANGLE=DASIN(SLOPE/(SLOPE*SLOPE+1))
      YINT=YAVG-SLOPE*XAVG
C      ROTATE THRU ANGLE 'ANGLE'; X1(I),Y1(I) = ROTATED PTS.
C      TRANSLATE (0.,YINT) TO ORIGIN.
      DO 180 I=1,27
      X1(I)=X(I)*COS(ANGLE)+(Y(I)-YINT)*SIN(ANGLE)
      Y1(I)=-X(I)*SIN(ANGLE)+(Y(I)-YINT)*COS(ANGLE)
180 CONTINUE

```

ROSENBLUM AND DEL BALZO

```

C     STORE AWAY XY COORDS IN XYPOS.ALL
      DO 185 I=1,7
        IBUF1(I)=IBUF(I)
185  CONTINUE
      DO 188 I=1,27
        X2(I)=X1(I)
        Y2(I)=Y1(I)
188  CONTINUE
      WRITE(11,IPNTR,ERR=99)(IBUF1(J),J=1,115)
500  CONTINUE
C     CLOSE FILES
99   CALL CLOSE(11)
      CALL CLOSE(10)
      END

```

```

C     PROGRAM XYPL1
C
C     THIS PROGRAM WILL PRODUCE WATERFALL PLOTS USING THE XY POSITIONS
C     PRODUCED BY ARSH7 AND STORED IN THE FILE XYPOS.ALL.
C     THIS VERSION OF XYPLT WILL PRODUCE AVERAGED WATERFALL PLOTS.
C
C     PROGRAMMER: L. ROSENBLUM   VERSION 6/4/80
C
      DIMENSION IBUF(115),ITIM(6),X1(27),Y1(27),YMEAN(27)
      EQUIVALENCE (IBUF(2),ITIM(1)),(IBUF(8),X1(1)),(IBUF(62),Y1(1))
      DATA YMAX1,PLTY/5.0,1.0/
C     SET UP FILE
      CALL ASSIGN(10,'SY:XYPOS.ALL')
      DEFINE FILE 10(1275,115,U,IPNTR)
C     GET LIMITS FOR DATA
      TYPE *, ' ENTER START RECORD NUMBER'
      ACCEPT *, INUM0
      TYPE *, ' ENTER # OF RECORDS TO BE AVERAGED'
      ACCEPT *, INUM3
      TYPE *, ' ENTER NUMBER OF AVERAGES ON ONE PLOT'
      ACCEPT *, INUM1
      TYPE *, ' ENTER # OF WATERFALL PLOTS'
      ACCEPT *, INUM2
      IPNTR=INUM0
C     DO NUM2 WATERFALL PLOTS
      DO 999 JJJ=1,INUM2
C     DO INUM1 AVERAGES ON ONE PLOT
      DO 500 II=1,INUM1
C     INITIALIZE MEAN
      DO 200 I=1,27
        YMEAN(I)=0.0
200  CONTINUE
C     GET INUM3 RECORDS FOR AVERAGE
      DO 250 KK=1,INUM3
C     READ A RECORD
      READ (10,IPNTR,ERR=99)(IBUF(J),J=1,115)
C     ADD NEW VALUES TO MEAN
      DO 210 I=1,27
        YMEAN(I)=YMEAN(I)+Y1(I)
210  CONTINUE

```

```

250 CONTINUE
DO 260 I=1,27
YMEAN(I)=YMEAN(I)/FLOAT(INUM3)
260 CONTINUE
C SCALE DATA
DO 130 I=1,27
X1(I)=X1(I)*6.0/600.0
YMEAN(I)=YMEAN(I)*0.5
130 CONTINUE
C INITIALIZE AND DRAW AXES
C SKIP INIT IF NOT 1ST TIME THRU
IF(PLTY.GT.9.0)GO TO 99
IF(PLTY.NE.1.0)GO TO 138
CALL AXIS(0.5,1.0,0.,600.,100.,0.,'METERS',6.1)
CALL SYMBOL(2.0,0.2,0.1,'TIME: ',0.0,6)
CALL SYMBOL(6.5,4.0,0.08,'1 INCH = 2 METERS ',90.0,18)
DO 135 I=2,5
RNUM=FLOAT(IBUF(I))
IF(I.EQ.5)RNUM=RNUM-1
X11=2.0+0.4*(FLOAT(I)-1.0)
CALL NUMBER(X11,0.2,0.1,RNUM,0.0,-0)
135 CONTINUE
CALL PLOT(0.5,1.0,-3)
C DRAW PLOT
138 DO 140 I=1,27
IF(YMEAN(I).GT.YMAX1)YMEAN(I)=YMAX1
IF(YMEAN(I).LT.-YMAX1)YMEAN(I)=-YMAX1
YMEAN(I)=YMEAN(I)+PLTY
IF(I.EQ.1)CALL PLOT(0.0,YMEAN(I),3)
CALL PLOT(X1(I),YMEAN(I),2)
140 CONTINUE
PLTY=PLTY+0.2
500 CONTINUE
PLTY=1.0
TYPE *, ' PLOT # ',JJJ,' COMPLETED'
99 CALL PLOT(0.,0.,999)
999 CONTINUE
CALL ENDPLT
CALL CLOSE(10)
END

```

```

C PROGRAM ARSTA
C
C THIS PROGRAM READS THE FILE XYPOS.ALL AND COMPUTES
C THE MEAN AND STANDARD DEVIATION. SEE PROGRAM
C ARSH7 FOR THE FORMAT OF XYPOS.
C
C

```

PROGRAMMER: LARRY ROSENBLUM VERSION 5-5-80

```

DIMENSION IBUF(115),X(27),Y(27),ITIME(6),YMEAN(27),INO(27)
DIMENSION YSD(27)
EQUIVALENCE (IBUF(8),X(1)),(IBUF(62),Y(1))
TYPE *, 'ENTER START RECORD #'
ACCEPT *, INUM0
TYPE *, 'ENTER NUMBER OF RECORDS TO BE PROCESSED'

```



```

DO 94 J=1,27
YOUT=2.0*YMEAN(J)/3.0
IF(J.GT.1)YOUT1=YOUT-2.0*YMEAN(J-1)/30.0
WRITE(11,93)J,YOUT,YOUT1
93 FORMAT(1X,'PHONE: ',I2,4X,'CORR: ',F8.5,4X,'MOD. CORR:',F8.5)
94 CONTINUE
90 FORMAT(5X)
99 CALL CLOSE(10)
CALL SPOOL(11)
C THIS SECTION PLOTS OUT RESULTS
CALL PLOTS
CALL AXIS(0.5,0.5,0.,30.,5.,0.,'PHONE ',6,1)
CALL AXIS(0.5,0.5,-1.0,1.0,0.4,90.0,'METERS',-6,1)
CALL SYMBOL(1.8,6.0,0.1,'START TIME: ',0.0,12)
DO 205 I=1,4
RNUM=FLOAT(ITIME(I))
IF(I.EQ.4)RNUM=RNUM-1.0
X11=2.0+0.4*FLOAT(I)
CALL NUMBER(X11,6.0,0.1,RNUM,0.0,0)
205 CONTINUE
CALL SYMBOL(2.0,5.8,0.1,'# OF RECS ',0.0,10)
RNUM=FLOAT(INUM1)
CALL NUMBER(2.6,5.8,0.1,RNUM,0.0,0)
CALL PLOT(0.5,3.0,-3)
CALL PLOT(6.0,0.0,2)
C SCALE DATA
DO 210 I=1,27
XPLT=6.0*FLOAT(I)/30.0
YTEM1=2.5*(YMEAN(I)-YSD(I))
YTEM2=2.5*(YMEAN(I)+YSD(I))
YTEM0=2.5*YMEAN(I)-0.02
XTEM0=XPLT-0.01
CALL SYMBOL(XTEM0,YTEM0,0.04,'0 ',0.0,2)
CALL PLOT (XPLT,YTEM1,+3)
CALL PLOT (XPLT,YTEM2,+2)
210 CONTINUE
CALL PLOT(0.,0.,999)
CALL ENDPLT
END

```

C PROGRAM ARST2

C THIS PROGRAM WILL COMPUTE SIGMA ACROSS THE ARRAY FOR A FIXED
C TIME: $SIGMA = \sqrt{((1/27) * \text{SUMMATION}((Y(J) - YAVG) * (J-1)))}$.
C THE PROGRAM THEN PRODUCES PLOTS OF EITHER SIGMA VS. TIME FOR
C USER INPUT AVERAGE INTERVALS OR HISTOGRAM TYPE PLOTS OF BIN
C SIZE VS. NUMBER OF OCCURANCES OF SIGMA WITHIN BIN. WHICH PLOT
C IS CHOSEN IS USER DETERMINED, AS IS THE BIN SIZE FOR THE
C HISTOGRAM TYPE PLOT.

C PROGRAMMER: LARRY ROSENBLUM VERSION: JULY 1, 1980

C DIMENSION IBUF(115),Y(27),ITIME(6),SIGMA(1275,4),IBIN(100)
C EQUIVALENCE (IBUF(2),ITIME(1)),(IBUF(62),Y(1))

```

C
TYPE *, 'ENTER: 1=SIGMA VS. TIME 2=BIN SIZE VS. # OCCURANCES'
ACCEPT *, IANS
IF(IANS.EQ.2)GO TO 5
TYPE *, 'ENTER # OF MINUTES TO AVERAGE OVER'
ACCEPT *, INUM0
GO TO 6
5 TYPE *, 'ENTER HISTOGRAM BIN SIZE (0.01 TO 0.5)'
ACCEPT *, BIN
IF(BIN.LT.0.01.OR.BIN.GT.0.5)GO TO 5
C
SET UP FILES
6 CALL ASSIGN(11, 'SY:ARST2.OUT')
CALL ASSIGN(10, 'SY:XYPOS.ALL')
DEFINE FILE 10 (1275,115,U,IPNTR)
IPNTR=1
C
C THIS SECTION COMPUTES SIGMA FOR EACH SHAPE
C
ICNT1=0
C READ A RECORD
10 READ(10*IPNTR,ERR=99)(IBUF(J),J=1,115)
C GET MEAN - NOTE: MEAN SHOULD BE 0 SINCE DONE BY LS FIT
YMEAN=0.0
DO 20 I=1,27
YMEAN=YMEAN+Y(I)
20 CONTINUE
YMEAN=YMEAN/27.0
C COMPUTE SIGMA
SUM=0.0
DO 30 I=1,27
SUM=SUM+(Y(I)-YMEAN)*(Y(I)-YMEAN)
30 CONTINUE
IPNT1=IPNTR-ICNT1
SIGMA(IPNT1-1,1)=ITIME(2)
SIGMA(IPNT1-1,2)=ITIME(3)
SIGMA(IPNT1-1,3)=ITIME(4)
SIGMA(IPNT1-1,4)=SQRT((1.0/27.0)*SUM)
ITST=IPNTR-(100*(IPNTR/100))
IF(ITST.EQ.0)TYPE *, IPNTR
..
GO TO 10
C CLOSE FILES
99 CALL CLOSE(10)
INUM=IPNTR-2-ICNT1
IF(IANS.EQ.2)GO TO 155
C
C THIS SECTION DOES PLOTS OF SIGMA VS. TIME
C
C GET START TIME
TSTRT=SIGMA(1,1)+SIGMA(1,2)/60.0
C INITIALIZE AND DRAW AXES
CALL PLOTS
CALL AXIS(0.5,0.5,0.,.8,.1,.0., 'TIME (HRS)', 10,1)
CALL AXIS(0.5,0.5,0.,.8,.1,90., 'SIGMA ', -6,1)
CALL SYMBOL(2.7,7.8,0.2, 'MINUTE AVERAGES ', 0.0,16)
RNUM=FLOAT(INUM0)
CALL NUMBER(0.5,7.8,0.2,RNUM,0.0,-0)
CALL PLOT(0.5,0.5,-3)

```

```

C     SCALE AND PLOT
SIGAV=0.0
ICNT=0
DO 150 I=1, INUM
TMTST=SIGMA(I,1)+SIGMA(I,2)/60.0+SIGMA(I,3)/3600.0
IF((TMTST-TSTRT).GE.(FLOAT(INUM0)/60.0))GO TO 110
SIGAV=SIGAV+SIGMA(I,4)
ICNT=ICNT+1
GO TO 150
C     HAVE INUM0 MINUTES OF DATA IN AVERAGE
110 SIGAV=SIGAV/FLOAT(ICNT)
XPLT=TSTRT+FLOAT(INUM0)/120.0
YPLT=10.0*SIGAV
TEST=ABS(XPLT-XOLD)
IF(TEST.LE.0.33)CALL PLOT(XPLT,YPLT,2)
IF(TEST.GT.0.33)CALL PLOT(XPLT,YPLT,3)
XOLD=XPLT
SIGAV=SIGMA(I,4)
ICNT=1
TSTRT=SIGMA(I,1)+SIGMA(I,2)/60.0
150 CONTINUE
CALL PLOT(0.,0.,999)
CALL ENDPLT
GO TO 199
C
C     THIS SECTION DOES HISTOGRAM TYPE PLOT OF BIN VS # OCCURANCES
C
C     INITIALIZE IBIN
155 DO 160 I=1,100
IBIN(I)=0
160 CONTINUE
C     COMPUTE # OF POINTS IN EACH BIN
NBIN=INT(1.0/BIN)
IF(FLOAT(NBIN).NE.(1.0/BIN))NBIN=NBIN+1
DO 170 I=1, INUM
INDEX=1+INT(SIGMA(I,4)/BIN)
IBIN(INDEX)=IBIN(INDEX)+1
170 CONTINUE
C     FIND MAX VALUE FOR ALL BINS
IMAX=0
DO 175 I=1,NBIN
IF(IMAX.LT.IBIN(I))IMAX=IBIN(I)
175 CONTINUE
C     FIND SCALE FOR Y AXIS AND DRAW AXES
CALL PLOTS
YMAX=FLOAT(6*(1+IMAX/6))
YSTEP=YMAX/6.0
CALL AXIS(0.5,0.5,0.,1.,0.2,0.0,'SIGMA BINS',10,1)
CALL AXIS(0.5,0.5,0.,YMAX, YSTEP,90.0,'# OF OCCURANCES ',-16,1)
CALL SYMBOL(2.0,6.2,0.1,'BIN SIZE =',0.0,10)
CALL NUMBER(2.5,6.2,0.1,BIN,0.0,2)
CALL PLOT(0.5,0.5,-3)
C     DRAW PLOT
DO 180 I=1,NBIN
XPLTL=5.0*FLOAT(I-1)*BIN
XPLTR =5.0*FLOAT(I)*BIN
YPLT=6.0*FLOAT(IBIN(I))/YMAX
CALL PLOT(XPLTL,YPLT,2)

```

ROSENBLUM AND DEL BALZO

```

CALL PLOT(XPLTR,YPLT,2)
CALL PLOT(XPLTR,0,2)
180 CONTINUE
CALL PLOT(0.,0.,999)
CALL ENDPLT
C   SPOOL OUTPUT
.. 199 CALL SPOOL(11)
.. END

.

C   PROGRAM BMFRM
C
C   THIS PROGRAM WILL TAKE THE XY POSITION FILE XYPOS.ALL
C   AND TAKE A FOURIER TRANSFORM TO PRODUCE BEAM PATTERNS.
C   THE RESULT IS PLOTTED OUT IN DB'S. CAN PLOT EITHER SINGLE
C   SCAN OR TIME AVERAGED BEAMPATTERNS.
C
C   PROGRAMMER: LARRY ROSENBLUM   VERSION: 6-26-80
C
DIMENSION IBUF(115),ITIM(6),Y(27),Y1(60),H(60),A(256),B(256),IS(4)
DIMENSION C(256),CDB(256),IBUF1(512),IBUF2(2),CLIN(256),BAVG(256)
COMPLEX A,B,ATEMP
EQUIVALENCE (IBUF(62),Y(1)),(IBUF(2),ITIM(1))
EQUIVALENCE (IBUF1(1),CLIN(1)),(IBUF2(1),YNORM)
DATA PI,SSPD,FREQ,ARDEL/3.1415926,1522.0,227.0,10.0/
DATA PHI,HNPWR,XAXLN,IBEGN/0.0,2.666667,6.0,0/

C   TYPE *, 'ENTER PHI - ANGLE OF ARRIVAL IN DEGREES'
ACCEPT *,PHI
TYPE *, 'ENTER FREQ.'
ACCEPT *,FREQ
IF(FREQ.EQ.0.0)FREQ=227.0
PHI1=PHI*PI/360.0
8 TYPE *, 'ENTER AVERAGING INTERVAL'
TYPE *, '0 = 1 SCAN, 1 = N MIN., 2 = 7 HOURS(ALL DATA)'
ACCEPT *, IANS
IF(IANS.LT.0.OR.IANS.GT.2)GO TO 8
C   SET UP FILES
CALL ASSIGN(11,'SY:BMFRM.OUT')
CALL ASSIGN(10,'SY:XYPOS.ALL')
DEFINE FILE 10(1275,115.U,IPNTR)
IPNTR=1
IF(IANS.EQ.2)GO TO 14
IF(IANS.EQ.0)TYPE *, 'ENTER RECORD #'
IF(IANS.EQ.0)ACCEPT *, IPNTR
IF(IANS.NE.1)GO TO 14
TYPE *, 'ENTER START TIME (HR.,MIN.)'
ACCEPT *, Isth, Istm
ISTRT=60*ISTH+ISTM
TYPE *, 'ENTER INTERVAL LENGTH IN MINUTES'
ACCEPT *, Imin1
ISTP=ISTRT+IMIN1
C   WRITE OUT TYPE OF PLOT
WRITE(11,333)PHI,FREQ
333 FORMAT(1X,'PHI = ',F8.2,4X,'FREQ = ',F8.2)
14 IF(IANS.EQ.0)WRITE(11,340)IPNTR

```

```

340 FORMAT(1X,'SINGLE SCAN - REC # ',I4)
    IF(IANS.EQ.2)WRITE(11,345)
345 FORMAT(1X,'AVERAGE - ALL DATA')
    IF(IANS.EQ.1)WRITE(11,350)IMIN1,ISTH,ISTM
350 FORMAT(1X,I4,' MINUTE AVG.',4X,'START TIME(HR.,MIN.): ',2I4)
C   GET LINEAR ARRAY PATTERN FROM FILE ARLIN.DAT
    CALL ASSIGN(2,'SY:ARLIN.DAT')
    DEFINE FILE 2(2,512,U,IPNT1)
    IPNT1=1
    READ(2'IPNT1)(IBUF1(J),J=1,512)
    READ(2'IPNT1)(IBUF2(J),J=1,2)
    CALL CLOSE(2)
C   GET HANNING WINDOW COEFS H(J) AND COMPUTE LAMBDA
    DO 5 I=1,60
    TEMP=PI*(FLOAT(I)-0.5)/30.0
    H(I)=0.5*(1.0-COS(TEMP))
    5 CONTINUE
    XLAMB=SSPD/FREQ
C   READ FILE
    10 READ(10'IPNTR,ERR=99)(IBUF(J),J=1,115)
    ITST=IPNTR-20*(IPNTR/20)
C   IF(ITST.EQ.0.AND.IANS.NE.0)TYPE *,'REC # ',IPNTR
C   SAVE 1ST TIME
    IF(IBEGN.EQ.1)GO TO 11
    DO 17 K=1,4
    IS(K)=ITIM(K)
    17 CONTINUE
    IF(IANS.EQ.0)WRITE(11,355)(IS(J),J=1,4)
    11 IF(IANS.EQ.2)IREC1=2
    IF(IANS.NE.1)GO TO 12
    ITST=ITIM(2)*60+ITIM(3)
    IF(ITST.LT.ISTR)GO TO 10
    IF(ITST.EQ.ISTR.AND.IBEGN.EQ.0)IREC1=IPNTR
    IF(IBEGN.EQ.0)WRITE(11,355)(IS(J),J=1,4)
355 FORMAT(1X,'START TIME: ',4I4)
    IBEGN=1
    IF(ITST.GE.ISTP)GO TO 99
    12 DO 15 I=1,256
    A(I)=(0.0,0.0)
    B(I)=(0.0,0.0)
    15 CONTINUE
C   PUT Y'S INTO CORRECT PLACE IN Y1'S
    DO 20 I=1,27
    Y1(2*(I+1))=Y(I)
    20 CONTINUE
C   INTERPOLATE BETWEEN EVEN Y1'S
    DO 30 I=5,55,2
    Y1(I)=(Y1(I-1)+Y1(I+1))/2.0
    30 CONTINUE
C   EXTERPOLATE FIRST 3 Y1'S
    DO 40 I=1,3
    Y1(I)=Y1(4)+(FLOAT(4-I)*(Y1(4)-Y1(5)))
    40 CONTINUE
C   EXTERPOLATE LAST FOUR PHONES
    DO 50 I=1,4
    Y1(56+I)=Y1(56)+(FLOAT(I)*(Y1(56)-Y1(55)))
    50 CONTINUE

```

```

C     COMPUTE COMPLEX FFT
DO 120 J=1,60
C1=2.0*PI/XLAMB
T1=FLOAT(J)*ARDEL*SIN(PHI1)
T2=Y1(J)*COS(PHI1)
TEMP=C1*(T1+T2)
A(J)=CEXP(CMPLX(0.0,TEMP))
A(J)=H(J)*A(J)
120 CONTINUE
CALL CFT(A,256,B)
C     REORDER THE CFT OUTPUT ARRAY
DO 130 I=1,128
B(I)=A(I+128)
B(I+128)=A(I)
130 CONTINUE
DO 140 I=1,256
A(I)=B(I)
140 CONTINUE
C     NORMALIZE THE ARRAY AND TAKE POWER
DO 150 I=1,256
ATEMP=256.0*A(I)
B(I)=(ATEMP*CMPLX(REAL(ATEMP),-AIMAG(ATEMP)))
C(I)=HNPWR*REAL(B(I))
150 CONTINUE
IF(IANS.EQ.0)GO TO 152
C     ADD NEWEST FFT INTO AVERAGE
DO 300 I=1,256
IF(IPNTR.EQ.IREC1)BAVG(I)=BAVG(I)+C(I)
IF(IPNTR.NE.IREC1)BAVG(I)=(C(I)/FLOAT(IPNTR-IREC1+1))
1 +(FLOAT(IPNTR-IREC1)/FLOAT(IPNTR-IREC1+1))*BAVG(I)
300 CONTINUE
GO TO 10
C     COMPUTE DB'S
152 CDBMX=-999.9
INDEX=0
DO 160 I=1,256
CDB(I)=10.0*LOG10(C(I)/YNORM)
IF(CDB(I).GT.CDBMX)INDEX=I
IF(CDB(I).GT.CDBMX)CDBMX=CDB(I)
160 CONTINUE
WRITE(11,330)INDEX,CDBMX
99 CALL CLOSE(10)
IF(IANS.EQ.0)GO TO 400
C     COMPUTE DB'S FOR AVERAGE
CDBMX=-999.9
INDEX=0
DO 320 I=1,256
CDB(I)=10.0*LOG10(BAVG(I)/YNORM)
IF(CDB(I).GT.CDBMX)INDEX=I
IF(CDB(I).GT.CDBMX)CDBMX=CDB(I)
320 CONTINUE
WRITE(11,330)INDEX,CDBMX
330 FORMAT(1X,'INDEX = ',I4.4X,'MAX DB = ',F8.3)
C     FIND 3DB DOWN POINTS
ARDEL=ARDEL*COS(PHI)
400 DO 450 I=130,256
IF(CDB(129)-CDB(I).LT.3.0)GO TO 450
ANG1=(180.0/PI)*ASIN((FLOAT(I-130)*XLAMB)/(256.0*ARDEL))
ANG2=(180.0/PI)*ASIN((FLOAT(I-129)*XLAMB)/(256.0*ARDEL))

```

```

RAT1=CDB(129)-CDB(I-1)
RAT2=CDB(129)-CDB(I)
RATIO=(3.0-RAT1)/(RAT2-RAT1)
ANG3=ANG1+RATIO*(ANG2-ANG1)
GO TO 455
450 CONTINUE
455 DO 460 I=1,128
    IF(CDB(129)-CDB(129-I).LT.3.0)GO TO 460
    ANG1=(-180.0/PI)*ASIN((FLOAT(I-1)*XLAMB)/(256.0*ARDEL))
    ANG2=(-180.0/PI)*ASIN((FLOAT(I)*XLAMB)/(256.0*ARDEL))
    RAT1=CDB(129)-CDB(130-I)
    RAT2=CDB(129)-CDB(129-I)
    RATIO=(3.0-RAT1)/(RAT2-RAT1)
    ANG4=ANG1+RATIO*(ANG2-ANG1)
    GO TO 465
460 CONTINUE
465 ANG=(ANG3-ANG4)/2.0
    WIDTH=2.0*ANG
    WRITE(11,470)ANG3,ANG4,WIDTH
470 FORMAT(1X,'3DB PTS',F8.3,2X,F8.3,' WIDTH ',F8.3,' DEG.')
C    NOW PLOT OUT BEAM PATTERN
    CALL PLOTS
    ANGMX=(180.0/PI)*ASIN(XLAMB/(2.0*ARDEL))
    XAXIN=ANGMX/(XAXLN/2.0)
    ANGM1=15.0
    XAXI1= 5.0
    CALL AXIS(0.5,0.5,-ANGM1,ANGM1,XAXI1,0.0,'ANGLE (DEG.)',12,1)
    CALL AXIS(0.5,0.5,-50.,0.,10.,90.,'RESPONSE (DB)',-14,1)
    CALL SYMBOL(1.0,6.0,0.1,'PHI =          DEG.',0.0,16)
    CALL NUMBER(1.1,6.0,0.1,PHI,0.0,-0)
    CALL SYMBOL(1.0,5.7,0.1,'FREQ =          HZ.',0.0,16)
    CALL NUMBER(1.1,5.7,0.1,FREQ,0.0,-0)
    CALL SYMBOL(4.2,6.0,0.1,'XXXX ACTUAL ARRAY ',0.0,20)
    CALL SYMBOL(4.2,5.7,0.1,'          LINEAR ARRAY ',0.0,20)
    CALL PLOT(4.2,5.75,3)
    CALL PLOT(4.55,5.75,2)
    IF(IANS.EQ.0)CALL SYMBOL(1.5,6.6,0.2,'SINGLE SCAN BEAMPATTERN ',0.
1    ,23)
    IF(IANS.NE.2)CALL SYMBOL(2.6,6.2,0.1,'START TIME: ',0.0,12)
    XLOC=3.4
    XADD=0.35
    DO 185 I=1,4
    RNUM=FLOAT(IS(I))
    IF(IANS.EQ.1.AND.I.EQ.4)GO TO 185
    XPOS1=XLOC+(I-1)*XADD
    IF(IANS.EQ.1)CALL NUMBER(XPOS1,6.2,0.1,RNUM,0.0,-0)
185 CONTINUE
    IF(IANS.GT.0)CALL SYMBOL(1.6,6.6,.2,'AVERAGED BEAMPATTERN ',0.,20)
    IF(IANS.EQ.1)CALL SYMBOL(2.6,6.4,.1,'          MINUTE AVERAGE',0.,20)
    XMIN1=FLOAT(IMIN1)
    IF(IANS.EQ.1)CALL NUMBER(2.1,6.4,.1,XMIN1,0.0,-0)
    IF(IANS.EQ.2)CALL SYMBOL(2.7,6.4,.1,'ENTIRE EVENT',0.,12)
    CALL PLOT(0.5,0.5,-3)
C    PLOT LINEAR PATTERN

```

ROSENBLUM AND DEL BALZO

```

DO 190 I=1,256
II=IABS(129-I)
ANG=(180.0/PI)*ASIN((FLOAT(II)*XLAMB)/(256.0*ARDEL))
IF(I.LT.129)ANG=-ANG
XPLT=(XAXLN/2.0)+(XAXLN/2.0)*(ANG/ANGM1)
IF(ABS(ANG).GT.ANGM1)GO TO 190
YPLT=5.0+CLIN(I)/10.0
IF(YPLT.LT.0.0)YPLT=0.0
IF(I.EQ.1)CALL PLOT(XPLT,YPLT,3)
IF(I.NE.1)CALL PLOT(XPLT,YPLT,2)
190 CONTINUE
C   PLOT ACTUAL ARRAY
DO 200 I=1,256
II=IABS(129-I)
ANG=(180.0/PI)*ASIN((FLOAT(II)*XLAMB)/(256.0*ARDEL))
IF(I.LT.129)ANG=-ANG
XPLT=(XAXLN/2.0)+(XAXLN/2.0)*(ANG/ANGM1)
IF(ABS(ANG).GT.ANGM1)GO TO 200
YPLT=5.0+CDB(I)/10.0
IF(YPLT.LT.0.0)YPLT=0.0
YHT=0.04
XFACT=(6.0/7.0)*(YHT/2.0)
YFACT=YHT/2.0
CALL SYMBOL(XPLT-XFACT,YPLT-YFACT,YHT,'X',0.0,2)
200 CONTINUE
CALL PLOT(0.0,0.0,999)
CALL ENDPLT
CALL SPOOL(11)
END

```