AFIT/GE/EE/77-20



# COMPARISON OF FAST FOURIER TRANSFORMS WITH OTHER TRANSFORMS IN SIGNAL PROCESSING FOR TACTICAL RADAR TARGET INDENTIFICATION

THESIS

AFIT/GE/EE/77-20 Robert L. Herron
Captain USAF



Approved for public release; distribution unlimited.

AFIT/GE/EE/77-20

# COMPARISON OF FAST FOURIER TRANSFORMS WITH OTHER TRANSFORMS IN SIGNAL PROCESSING FOR TACTICAL RADAR TARGET IDENTIFICATION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

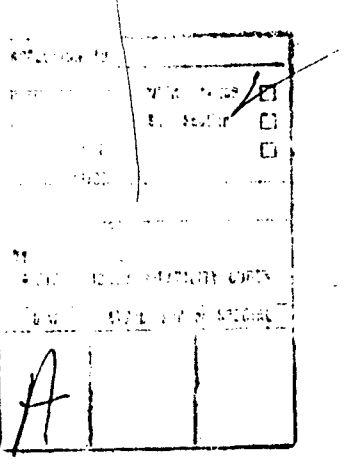
Requirements for the Degree of

Master of Science

Robert L. Herron, B.S.E.E.
Captain USAF

Graduate Electrical Engineering

December 1977



Approved for public release; distribution unlimited.

## Preface

This thesis was prepared at the request of the United States Air Force Systems Command's Rome Air Development Center (RADC/OCTM), Griffiss AFB, New York. Engineers at RADC are developing a tactical radar target identification system for use in the near future in the European theater. It has been designated a very high priority Air Force Technical Need.

The goal of this thesis is to contribute directly to meeting this need by investigating alternate signal processing techniques and attempting to apply these techniques to improve the performance of the TTI system.

I wish to acknowledge my indebtedness to my advisor, Capt. (Dr.) Gregg Vaughn whose suggestions and advice were greatly appreciated. I wish to thank Maj. Joe Carl for his valuable assistance and comments. I want to express my sincerest appreciatior to my laboratory sponsor, Richard J. Wood, RADC/OCTM, for his support and Dr. Robert Herman, Syracuse Research Corporation, for the use of the simulation program used in this thesis and for his help with understnading how it works.

I wish also to thank my wife, Renee, for her special understanding, encouragement, and patience throughout my AFIT program.

Robert L. Herron

This thesis was typed by Ms. Sharon Vogel

## Contents

## List of Figures

## List of Tables

## List of Symbols

| | |
|---|---|
| $X_r$ | Cross range |
| $\Delta X_r$ | Cross range resolution |
| $\lambda$ | Carrier wavelength |
| $\Delta\theta_m$ | Total integration angle - total aspect angle change |
| $\Omega$ | Angular velocity |
| $X_{ACR}$ | Unambiguous cross range window |
| R | Range |
| $\dot{R}, v_r$ | Range velocity, range rate |
| $f_D$ | Doppler frequency |
| c | Propagation velocity (speed of light) |
| $\sigma$ | Radar cross section |
| * | Arithmetic convolution |
| $\theta$ | Dyadic or logical convolution |
| $\bullet$ | Modulo-2 arithmetic with no carry |
| $\tau$ | Time shift |
| $R(\tau)$ | Arithmetic autocorrelation function |
| L(k) | Logical autocorrelation function |
| t | Time (continuous) |
| f | Frequency |
| s | Sequency |
| $E\{\cdot\}$ | Expected value |
| $F[\cdot]$ | Fourier transform operator |
| $W[\cdot]$ | Walsh transform operator |
| x(t) | Function of time |
| x(n) | Discrete function of time |

| | |
|---|---|
| $X(f)$ | Continuous Fourier transform |
| $X(k)$ | Discrete Fourier transform |
| $j$ | $\sqrt{-1}$ |
| $[\cdot]$ | Matrix |

## Abstract

The High Resolution Radar Branch of the Rome Air Development Center has developed a tactical target identification (TTI) pulsed-Doppler radar system which generates two-dimensional "images" of aircraft. The signal processing technique utilized the fast Fourier transform (FFT) to produce a slant-range versus cross-range display. If the TTI system is to be effectively employed in an aerial warfare environment then real-time processing is necessary. In an effort to speedup the signal processing several alternative transforms were studied as possible substitutes for the FFT. The Karhunen-Loeve, Cosine (Sine), Mellen, and Hankel transforms were investigated and found to be infeasible for use in TTI imaging. The Walsh (Hadamard) transform was studied in detail and tested in a simulation program and found that it could not be utilized in the TTI signal processing.

Two methods of converting from the Walsh sequency domain to the Fourier frequency domain were studied. The first scheme, a recursive relationship between the arithmetic and logical autocorrelation functions as presented by Robinson was discovered to be incorrect. The second, a method of computing the Fourier coefficients from the Walsh coefficients of a function was demonstrated to be too time consuming to be implemented in TTI signal processing.

Several floating-point FFT implementations were tested using the simulation program. Also, several fixed-point FFT algorithms were derived and tested. All of these were evaluated on the basis of speed and memory requirements and one fixed-point FFT algorithm was shown to be fast enough and accurate enough for implementation on the TTI Mini-computer.

# COMPARISON OF FAST FOURIER TRANSFORMS WITH OTHER TRANSFORMS IN SIGNAL PROCESSING FOR TACTICAL RADAR TARGET IDENTIFICATION

## I. Introduction

The High Resolution Radar Branch of the Rome Air Development Center, Griffiss AFB, New York is interested in developing a tactical target identification (TTI) radar system which can effectively respond to the expected WARSAW Pact threat against NATO in the 1980-1990 time frame. Radar target identification of aircraft facilitates the effective control of friendly airborne interceptor and close air support aircraft without active Identification Friend or Foe (IFF) devices. It also permits the positive identification and determination of uncooperative or enemy aircraft type and mission without endangering friendly aircraft and allows the discrimination between actual enemy aircraft and decoy vehicles (Ref 92).

The objective of the TTI development program is the generation of two-dimensional "images" of aircraft by processing in real-time the radar returns of the aircraft using a tactical pulsed-Doppler wideband radar system. Basic research is being conducted by the Syracuse Research Corporation (SRC), Syracuse, New York. The target identification simulation program used in this thesis was written by SRC and is based on parameters modelled the same as those of the ALCOR* system which has been used in actual radar imaging studies (Ref 31:1). In

---

*ARPA (Advance Research Projects Agency)/Lincoln Laboratory C-Band Observables Radar.

essence, the two-dimensional image can be formed if a target's aspect angle changes sufficiently relative to the radar over some time interval. The integration of the received radar waveform with respect to aspect angle will yield a slant range vs. Doppler array with entries of radar cross-section intensity. This array can be displayed giving an image of the target's highlights. The feasibility of such an approach using a pulsed-Doppler wideband radar system and associated signal processing has been established by Rafael (Ref 68) and Strattan (Ref 83).

The objective of this thesis is to investigate, evaluate, and compare other orthogonal transform (FFT) currently being used in the signal processing portion of the target identification system. The goal is to increase the speed of the signal processing to approach real-time. Image quality, processing time, and computer memory requirements must be considered when investigating and evaluating an alternate transform.

The remainder of this thesis is divided into five sections. Section II provides the theory, algorithm, and simulation of the radar target identification system. Section III investigates and considers alternate orthogonal transforms. Section IV proposes two methods for converting from the Walsh sequency domain to the Fourier frequency domain. Section V compares several Fast Fourier Transform algorithms. Conclusions and recommendations are made in Section VI. The Appendices contain listings of the computer programs, subprograms, and radar and target parameter data used in this thesis.

## II. Radar Target Identification: Theory, Algorithm, and Signal Processing

This section provides the background material on Radar Cross Section measurement and estimation. The basic assumptions are given and the salient features of the imaging algorithm are presented. Additionally, the simulation program used in this thesis is explained.

### Target Radar Cross Section

Radar cross section (RCS), $\sigma$, is a measure of the energy reflected from a target toward the receiving antenna (Ref 33:455). The RCS of a target is the area assumed to intercept the incident radiation, which, when isotropically reradiated, yields the actual power density at the receiving aperture (Ref 33:455). This returned energy varies with a multitude of parameters such as transmitted wavelength, polarization, target geometry, orientation, and reflectivity (Ref 58:141).

More precisely, the radar cross section of an object is proportional to the far-field ratio of reflected to incident power density, that is

$$\sigma = \frac{\text{Power reflected back to receiver/unit solid angle}}{\text{Incident power density}/4\pi} \tag{1}$$

(Ref 58:142). For an example, consider the RCS of a perfectly conducting isotropic scatterer. The power intercepted by the radiator is the product of the incident power density, $P_I$, and its geometric projected area, $A_I$. By the definition of isotropic scattering, this power is uniformly distributed over $4\pi$ steradians (Ref 58:142). For this isotropic scatterer then

$$\sigma_i = 4\pi \left[ \frac{P_I A_I / 4\pi}{P_I} \right] = A_I \tag{2}$$

Thus, the RCS of such an isotropic reflector is the geometric projected area (Ref 58:141).

For a complex target, such as an aircraft or missle, the RCS can be approximated by breaking the body into individual reflectors (scatterers) and assuming that the parts do not interact. In this case it can be shown that the total RCS is the vector sum of the individual cross sections

$$\sigma = \left| \sum_{k=1}^{N} \sqrt{\sigma k} \exp\left(\frac{j\Delta\pi d_k}{\lambda}\right) \right|^2 \tag{3}$$

where $\sigma_k$ is the RCS of the k th scatterer, $d_k$ is the distance between the k th scatterer and the receiver, and N the total number of scatterers (Ref 58:144).

Another approach considers the relative phase angles between the returns from these N scatterers. This approach leads to the following expression for the RCS of the entire body

$$\sigma = \left| \sum_{k=1}^{N} \sqrt{\sigma k} \exp(j\phi_k) \right|^2 \tag{4}$$

where $\phi_k$ is the relative phase angle associated with the k th component (Ref 26:974). It has been shown that the RCS of a target is related to the frequency response of the object, $G(j\omega)$, by the following relationship,

$$\sigma_s = |G(j\omega)|^2 \tag{5}$$

(Refs 51:1651; 83:5). The RCS, $\sigma_s$, can be thought of as the spectrum of the complex target. In general, $G(j\omega)$ will be aspect angle dependent except for a spherically symmetric object (Ref 51:1651).

Now, if a CW or pulses radar signal is reflected by a target moving

at a velocity, $v_r$, relative to the radar receiver, the whole spectrum would be translated in frequency by the Doppler shift, $f_D$, where

$$f_D = \frac{2\dot{R}f_o}{c}$$

$$= \frac{2v_r}{\lambda} \qquad (6)$$

where c is the propagation velocity (speed of light), $f_o$ is the transmitted carrier frequency, $\dot{R}$ or $v_r$ is the range rate or radial velocity, and $f_D$ is the Doppler shift (Refs 58:5; 47:357).

In a wideband pulsed-Doppler radar system, the received Doppler frequency spectrum is considered the target's radar cross section which is aspect angle dependent (Ref 58:173). Figure 1 shows the effect of return power from an aircraft where the surface scatterers making up the composite target echo can create a transition from phase addition to phase cancellation and change the cross section drastically (Ref 33:26). The spectra of RCS flucuations can be described in terms of several effects with the airframe the most important contributer (Ref 58:173). The airframe spectrum is due to the relative motion between the various scattering points on the fuselage and wings. This relative motion occurs as the aircraft aspect changes (Refs 58:173; 51:1651; 26:973).

The resulting spectral width is proportional to the transmitted frequency (Refs 58:73; 83:3). The frequency domain will give a bandwidth, therefore, of

$$B = 1/T \qquad (7)$$

where T is the pulse duration length (Ref 83:4).

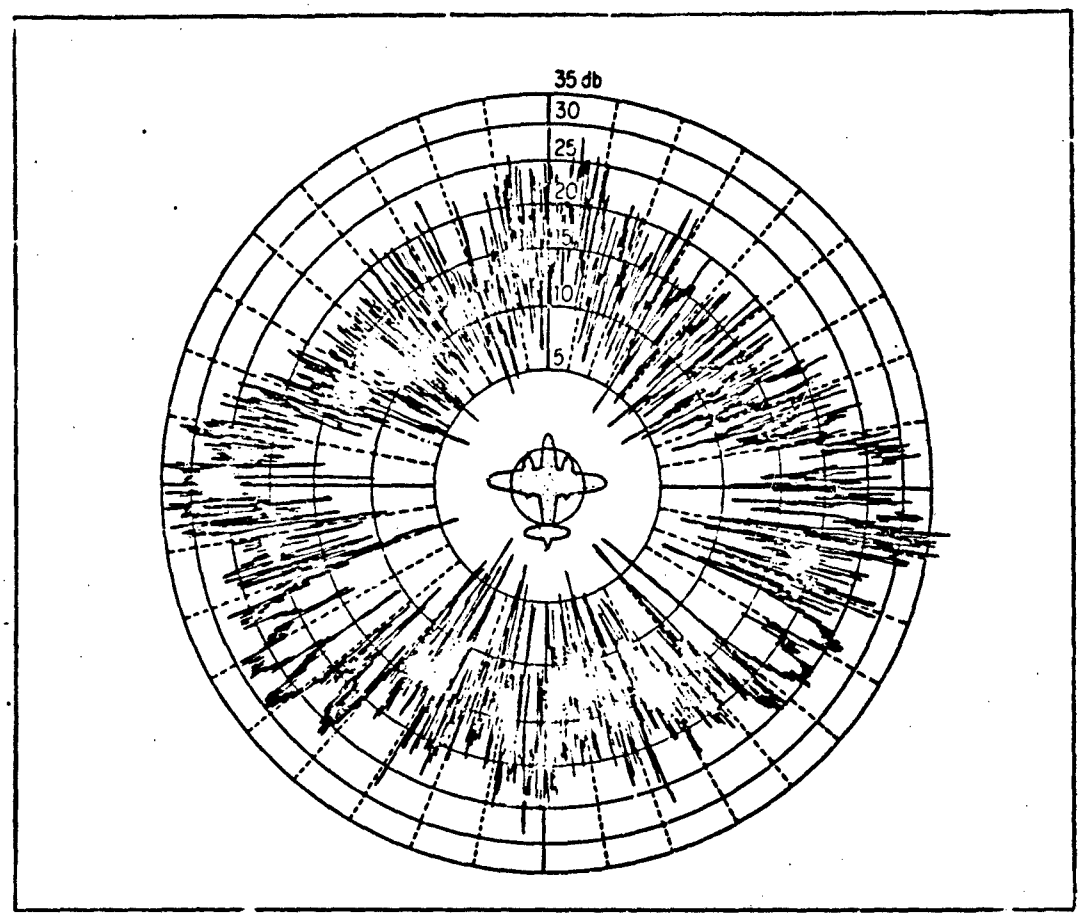



Figure 1. Return Power from an Aircraft as a Function of Azimuth Angle

RCS Measurement

The measurement of the radar cross section of a target makes use of Fourier transform theory and it will be shown that $|G(j\omega)|^2$ is in reality the Power Spectral Density (PSD) or Power Spectrum of the impulse response, g(t), of the target.

Fourier Integral. The Fourier Integral, defining a Fourier transform pair, is given for f(t) specified on the interval $(-\infty,\infty)$ as

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)\exp(-j\omega t)dt \qquad (8)$$

and

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)\exp(j\omega t)d\omega \qquad (9)$$

Power Spectral Density. If f(t) is specified on the interval $(-\infty,\infty)$ and if f(t) and F(jω) are a Fourier transform pair, then, the Power Spectral Density of the function, f(t) is defined as the absolute value squared of the Fourier transform of f(t). If P(ω) is the Power Spectrum, then

$$P(\omega) = |F[f(t)]|^2$$

$$= |F(J\omega)|^2 \qquad (10)$$

where F[·] is the Fourier transform operator. Also, if the autocorrelation function of f(t), R(τ), is Fourier transformed, then the following relationships will be found to be

$$R(\tau) = \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} f(t)f(t+\tau)dt$$

$$= f(t) * f(-t) \qquad (11)$$

where * denotes convolution, but

$$F[f(t)*f(t)] = F(j\omega)\cdot F(-j\omega)$$

$$= |F(j\omega)|^2 \qquad (12)$$

so that

$$P(\omega) = F[R(\tau)] \tag{13}$$

Equation (10) is known as the direct method for obtaining the power spectrum of a time series and is equal to the Fourier transform squared of the function (Ref 76:14,15). Equation (13) is known as the direct method or the Wiener-Khintchine theorem. It states that the power spectrum is the Fourier transform of the autocorrelation function (Refs 76:3; 28:128).

Therefore, it can be seen that Equation 5 is in fact the Power Spectral Density of g(t), the impulse response of the target. (That which would be measured at the Doppler filter.)

In this section the power spectrum is defined for an infinitely long, continuous time function, f(t) or g(t). However, in practical situations, only a finite amount of time is even available to observe the time function, particularly if a monostatic radar system is used.

Since the signal is in effect truncated, the effects on the Power Spectral Density resulting from the truncation of the data set must be considered.

RCS Estimation

Since the RCS of a target cannot be completely determined, it must be estimated. The Discrete Fourier transform (DFT) can be used to compute an estimate of the Radar Cross Section of a complex target.

Discrete Fourier Transform. Let f(n) be defined by N samples. The Discrete Fourier transform pair are defined as

$$F(k) = \sum_{n=0}^{N-1} f(n)\exp(-j2\pi(kn)/N) \tag{14}$$

and

$$f(n) = \sum_{k=0}^{N-1} F(k)\exp(j2\pi(kn)/N) \tag{15}$$

where the exponential function is periodic of period N (Ref 60:100).

Discrete Power Spectral Density. The sample power spectral density function or what is known as the raw periodogram is the DFT of $R(\tau)$, the autocorrelation function of f(n) (Ref 88:13). Define

$$P(\omega) = \frac{1}{2\pi} \sum_{\tau=-N}^{N} R(\tau)\exp(-j\tau\omega) \tag{16}$$

where $\tau$ is an integer.

Or conversely, the periodogram can be calculated as the modulus of the DFT of f(n).

$$P(\omega) = \left|\frac{1}{\sqrt{2\pi N}} \sum_{n=1}^{N-1} f(n)\exp(-jn\omega/N)\right|^2 \tag{17}$$

Equation (16) is analogous to the continuous Wiener-Khintchine theorem only in discrete form.

As it turns out, whichever way is used, the raw periodogram is an unsatisfactory estimate of the power spectrum unless the signal is perfectly periodic and noiseless. Therefore, the periodogram of a stochastic process will be an unstable estimate, erratic in appearance and behavior (Ref 88:14). The variance of the fluctuation

of the periodogram about the true power spectrum does not decrease to zero as N approaches infinity, as it should for a well behaved estimator (Refs 88:14; 30:109). Its variance is independent of N, and the probability distribution of the sample periodogram is a Chi-squared distribution with two degrees of freedom (Ref 88:80). To bring the variance down and stabilize the estimate, it is necessary to do some form of spectral averaging (Refs 76:13; 88:82; 62:548).

Currently, the "practice" of power spectral estimation has a strong empirical basis because most optimum techniques, such as maximum likelihood estimation, require more information about the signal than is usually available (Ref 62:532). As a result trade-offs are involved between different techniques such that there is no general agreement on the best method. The reader is directed to the literature for a more detailed look at power spectral estimation. Davenport and Root (Ref 28), Welch (Ref 90), Webb (Ref 88), Sentman (Ref 76) and others are excellent references. Oppenheim and Shafer (Ref 62) provide an excellent overview of the power spectral estimation problem. Deutsch (Ref 30) is a recommended reference for estimation theory. Blacksmith, et al, (Ref 17) contains an extensive bibliography on work done on radar cross section measurement.

Estimating Discrete Power Spectrums

Let f(t) be defined on the interval $(-\infty,\infty)$. If f(t) is truncated by multiplying by a data or observation window, w(t), (t can be considered either continuous or discrete) such that

$$w(t) = \begin{cases} 1, & |t| \leq T/2 \\ 0, & |t| > T/2 \end{cases} \tag{18}$$

then the truncated data set becomes

$$h(t) = f(t) \cdot w(t) \tag{19}$$

as shown in Figure 2 on the next page.

The function h(t) now represents the truncated data set available from which the power spectrum is to be calculated. The Power Spectral Density, $P_{ap}(\omega)$ of h(t), representing the apparent power spectrum of f(t), is then

$$P_{ap}(\omega) = |F[f(t) \cdot w(t)]|^2 \tag{20}$$

$$= |F(j\omega) * W(j\omega)|^2 \tag{21}$$

$$= |F(j\omega)|^2 * |W(j\omega)|^2 \tag{22}$$

Thus, the PSD, $|F(j\omega)|^2$, is modified by a convolution with $|W(j\omega)|^2$, the Fourier transform of the data window. $W(j\omega)$ is called the frequency window and is a sin x/x (sinc x) function as shown in Figure 3.

Convolution by the frequency window causes a certain degree of smoothing in the calculated PSD, but a small amount of leakage via the sidelobes from nearby frequency bands into the frequency band of

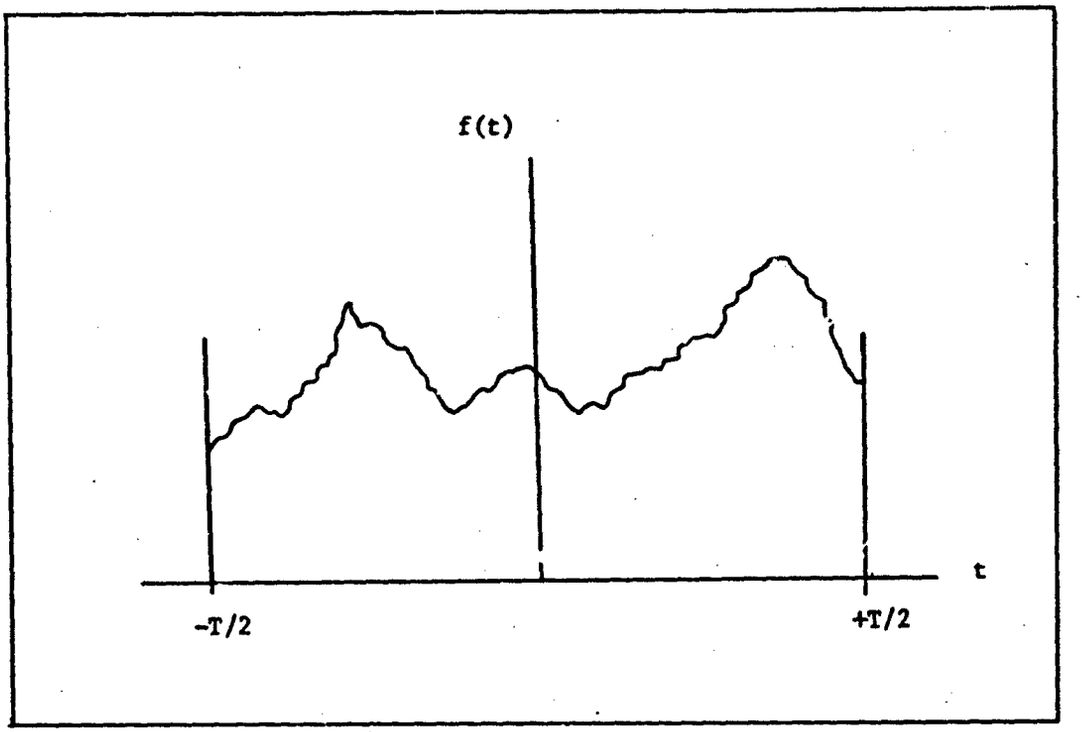



Figure 2. Truncated Data Set

interest (Ref 76:12). Normally, this is not serious, but if there are large well defined peaks in the power spectrum and when convolved with the frequency window, they act to reproduce the window, producing spurious peaks in the PSD corresponding to the sidelobes of the frequency window. These spurious peaks may be mistakenly identified as structural details of the true power spectrum when in fact they are merely artifacts created by the truncation of the original data set (Ref 76:13).

Several methods exist for sidelobe suppression. Three common methods include data tapering, sectioning and averaging, and data weighting (Refs 88:84; 2:548; 90:56). The simplest and most straight forward method is the use of data weighting. Several types exist and the most widely used is the Hamming weight function

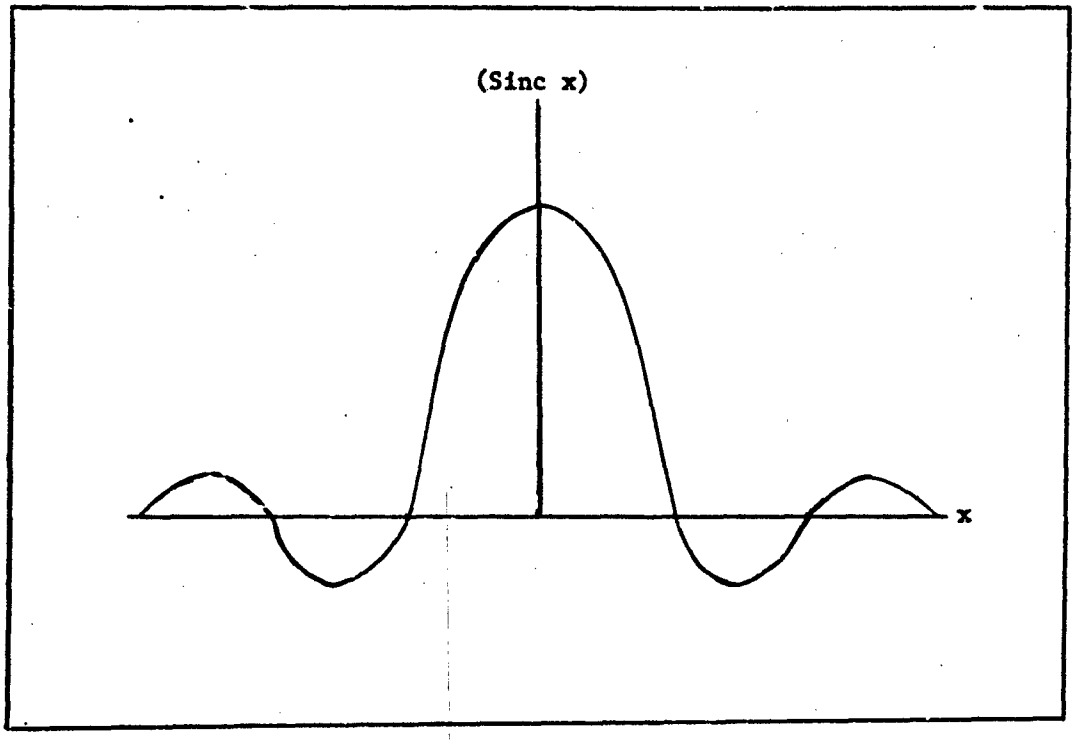



Figure 3. Sin x/x (Sinc x) Function

$$w_H(n) \leq 0.54 - 0.46\cos(2\pi n/(N-1)), \tag{23}$$

for

$$0 \leq n \leq N-1$$

where N is the total number of samples (Ref (62:241-242).

The statistical reliability of PSD estimates is discussed by Webb (Ref 88), Sentman (Ref 76), and Davenport and Root (Ref 28).

Assumptions

The basis of the imaging technique used in this thesis is based

on the assumption that the body being observed is rigid, and consists of sections that behave as point scatterers (Ref 56:1-4). It is also assumed that, relative to the observer, the object has some motion about a fixed set of axes (Ref 56:1-4). Figure 4. shows a typical body with the axes centered at the center of rotation of the body and with the rotation vector normal to the x-y plane (Refs 56:1-4,4; 68:5). The body is assumed to remain in one range bin. The separation between transmitted pulses is sufficient to prevent overlapping, which is met by the following condition

$$4T_d = N_r^{5/2} T_b \tag{24}$$

where $T_d$ is the duration of the total waveform, $T_b$ is pulse duration within the waveform, and $N_r$ is codelength required for PRF staggering (Ref 82:4). The maximum unambiguous range of the radar is determined by the burst length, $T_d$, so that

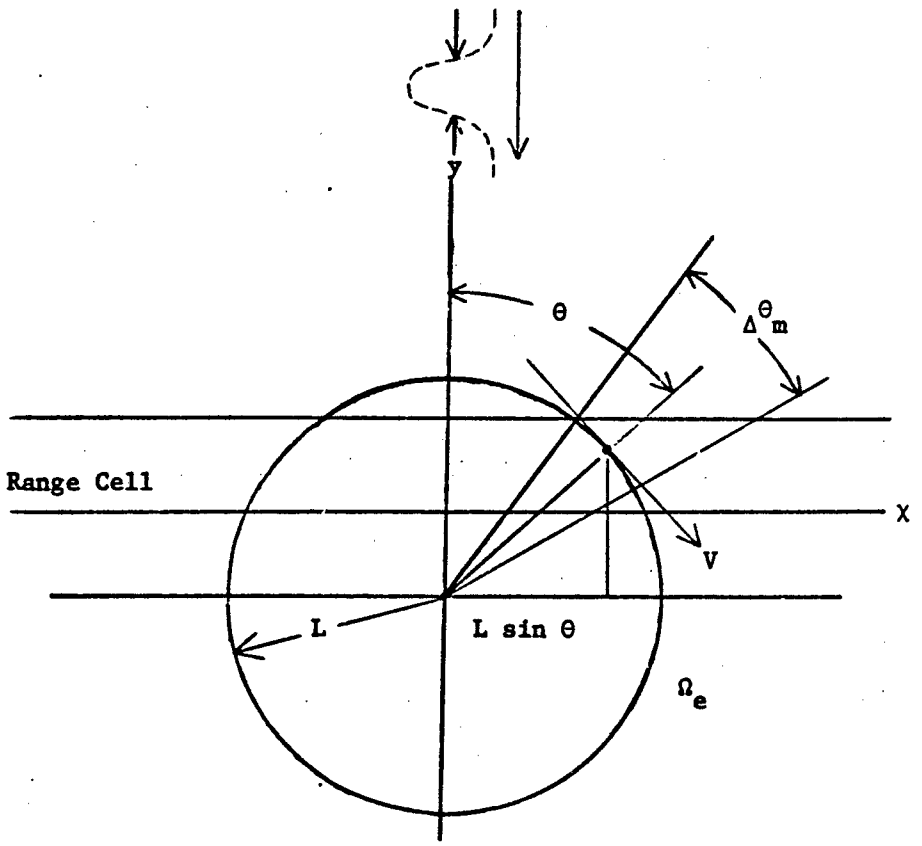
$$R_{max} = cT_d/2 \tag{25}$$

and the first Doppler ambiguity is sufficiently removed to correspond to double velocity of the fastest target to eliminate foldover, or

$$\frac{1}{T_c} = 4\dot{f}y_{max}/c \tag{26}$$

where $T_c$ is the average interval between subpulses (Ref 83:4).

Additionally, it is assumed the signal processing is of first-order, which means that integration intervals are short enough for certain linearities to pervail, that is, the body has linear motion (no acceleration) during the processing interval in which the target is rotating about some effective rotation center (Ref 68:3). In other words, the Doppler shift produced by the rotating scattering points is




Arbitrary location but point rotater about fixed x-7 center with radius L

Two-way Doppler shift $f_o \simeq V_r/(\lambda/2)$

$V_r$ = Radial Velocity

$V = \Omega_e L$ = Total velocity of point

$\Omega_e$ = rotation rate

$V_r = V \sin \theta = \Omega_e L \sin \theta$

$L \sin \theta = X$ = position along $\chi$-axis

$f_D = \Omega_e X/(\lambda|2)$

$X = f_d(\lambda/2)\ /\Omega_e$

Figure 4. Range-Doppler Image Relationships

considered approximately constant during a signal processing interval. The rotation can be viewed as a changing aspect between the radar line of sight (LOS) and the target.

Imaging Algorithm

The radar imaging simulation program models the target as a collection of scattering points $\vec{p}_i$, with each point assigned an RCS $\sigma_i$ which is assumed to be aspect angle independent (Refs 31:1; 57:2). At equal time increments the RCS is modelled as a function of frequency according to

$$\sqrt{\sigma(f)} = \sum_{i=1}^{N} \sqrt{\sigma_i} \exp\left(-j4\pi f \vec{k}\cdot\vec{r}_i / c\right) \tag{27}$$

where N represents the number of scatterers, $\vec{k}$ is the RLOS direction vector, and $\vec{r}_i$ is the range from an origin on the body to the point $\vec{p}_i$ (Ref 31:1). For one image, "flight" continues until the body has undergone sufficient aspect angle change to give a cross-range resolution of 1.5 feet or approximately 2.9 degrees of change, as determined by

$$\Delta X_r = \frac{f_D(\lambda/2)}{\Omega_e} = \frac{(\lambda/2)}{\Delta\Theta m} \tag{28}$$

where $\Delta\Theta m$ is the aspect angle change of the body relative to the RLOS and $\Omega_e$ is the effective vehicle rotation rate (angular velocity) (Ref 68:5).

The range resolution is shown in Figure 5 is determined by the radar bandwidth. The dwell time on target needed to produce an image with a given $\Delta X_r$ is

$$T_{dwell} = 0.65 / (\Delta X_r \Omega_e) \tag{29}$$

where it can be shown that

$$\vec{\Omega}_e = \vec{\Omega}_L - \vec{\Omega}_v \sin \Theta \tag{30}$$

where $\vec{\Omega}_L$ is the instantaneous RLSO angular velocity due to the varying aspect between the target track and the RLOS and $\vec{\Omega}_v \sin \Theta$ is the instantaneous velocity of the vehicle perpendicular to the RLOS (Refs 68:6; 56:1-5).

Since a moving target has many degrees of freedom of motion, the effective vehicular rotation may be brought about by changing aspect between the RLOS and the target track, and by changes in aspect due to target motion abouts its center of mass (Ref 68:6).

The magnitude of $\vec{\Omega}_e$ is used to scale the image from Doppler to cross-range units, and the image projection plane is the plane perpendicular to $\vec{\Omega}_e$ (Ref 68:6). This assumes that $\vec{\Omega}_e$ is essentially constant in both magnitude and direction over the processing interval (Ref 68:6).

Signal Processing

Data from a total time span T, where total angular change is given by

$$\Delta\Theta_M = \vec{\Omega}_e T \tag{31}$$

and a range window R centered on the target are sampled and collected and stored in an array. Both the amplitude and phase of the returns are stored. The sampling increment in range is R and in time, $\Delta t$, where

$$\Delta t = 1/\text{radar pulse repetition frequency} \tag{32}$$

To put the data into the assumed form of a collection of points rotating about a rotation center, it is necessary to compensate for the motion of the rotation of the rotation center with respect to the radar since if the target is rotated slightly, the phase of the center will change (Ref 68:7). This can be done by obtaining the distance to the rotation center as a function of time with radar tracking data, and then by subtracting the calculated signal phase at each time from all of the phase values in a pulse return recorded at that time. Alternately, the phase recorded for an actual discrete target point in the signal may be used as a reference for compensating all the phases in the return (Refs 56:1-7; 68:7). This process is known as aligning or "cohering" the data. Next, the data are Hamming weighted, as discussed earlier, for sidelobe control and Fourier transformed along constant slant range lines to produce the image output (Ref 68:7).

The unambiguous cross-range window $X_{ACR}$ is given by

$$X_{ACR} = \frac{\lambda/2}{\Delta\Theta_M} = \frac{\lambda/2}{\Omega_e \Delta t} \tag{33}$$

The corresponding unambiguous range window, $\dot{R}_{ACR}$, and the Doppler window, $D_{ACR}$, are simply

$$\dot{R}_{ACR} = \frac{\lambda/2}{\Delta t} \quad \text{ft/s} \tag{34}$$

$$D_{ACR} = \frac{\Delta}{\Delta t} \tag{35}$$

with $\lambda$ in feet and $\Delta t$ in seconds (Ref 68:8).

The cross-range grid increment, $\Delta X$ is given by

$$\Delta X = X_{ACR}/(\text{number of output points in the transform}) \tag{36}$$

In general, $\Delta X < \Delta X_r$, the cross-range resolution, in order that the sampling not be too coarse to miss significant output features (Ref 68:9). Strattan (Ref 83) points out that the number of profiles (RCS estimates) integrated should be at least equal to the ratio of the maximum cross-range dimension of the target to $\Delta X_r$ and that the transforms should be performed at slant-range intervals no larger than the range resolution

$$\Delta R \triangleq c/2B \tag{37}$$

where $B \triangleq 1/T$, the Doppler bandwidth of the system. The angular interval needed for the cross-range scale factor may be determined approximately from flight path tracking data. Figure 5 shows the resolution relationship necessary for good imaging (Ref 68:5).

Detailed mathemetical first-order range-Doppler processing is given by Rafael (Ref 68).

Simulation Program TGTID

The Simulation Program TGTID (Appendix A) used in this thesis was written by researchers at the Syracuse Research Corporation, Syracuse, New York. The listings of the program and its supporting subroutines are located in the Appendices.

The "input data" is coherent radar cross-section data as a function of frequency. It is inversed Fourier transformed to radar cross-section as a function of range. This data is first aligned (cohered) so that the first peak of each range sample occupies the same range bin and then a phase adjustment is made giving the first peak zero phase.

For fixed range, the adjusted data (now radar cross-section as a function of pulse repetition interval) is Fourier transformed along

Assume $f_D$ is constant over rotation interval T.

Target rotates $\Delta\theta m = \Omega_e T$

Stays within range cell.
Spectral analysis locates $f_D$ therefore X.

Doppler resolution $f_{Dr} = \frac{1}{T} = \frac{e}{\Delta\theta m}$

Cross range resolution

$$X_r = \frac{f_{Dr}(\lambda/2)}{\Omega e} = \frac{(\lambda/2)}{\Delta\theta m}$$

Y location given by slant range resolution.

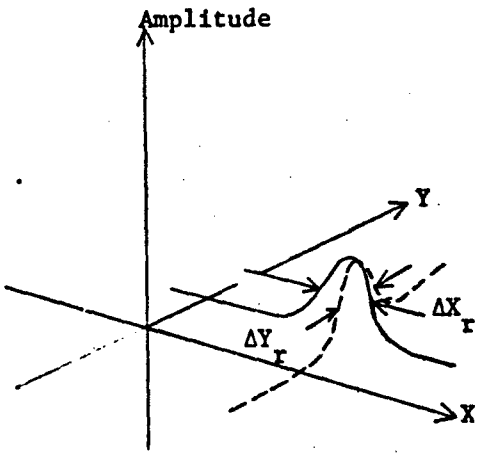



Figure 5. Range-Doppler Resoletion Relationships

constant range cells to give the RCS in terms of Doppler frequency which is then scaled to cross-range. The result is a cross-range versus range "radar image" whose entries are the associated RCS's (Ref 56:2). This array is then displayed and "squared up" so that an undistorted "picture" of the aircraft RCS response is realized. A block diagram of the processing is shown in Figure 6 (Ref 68:8).

Program TGTID Description

The main program TGTID (Appendix A) reads in the radar (location and frequency) and target (scatterer) parameters. Subroutine CINIT (Appendix B) computes the number of samples and order of these samples as a power of two based on the given parameters. Subroutines SLANTV, (Appendix B),DOTP (Appendix B), CTRAN (Appendix B), FFT (Appendix C),

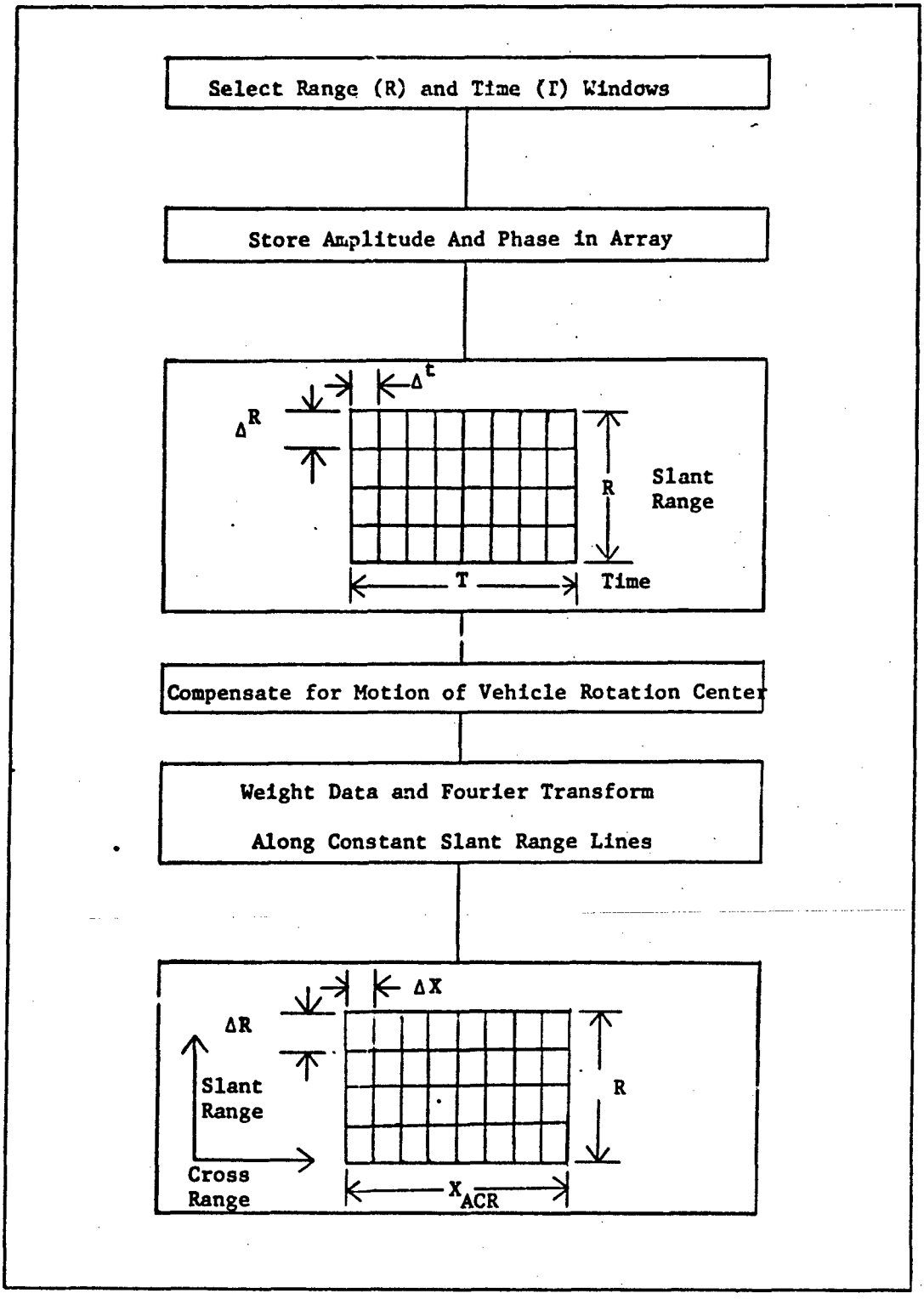



Figure 6. Block Diagram of Basic Image Processing

and ROLL (Appendix B) are used to "generate" the RCS data as a function of range. Subroutines CIMAGE, (Appendix B), HAMWGT (Appendix B), FFT, and ROLL are used to process the data and Subroutines PLOTID and BUFOUT (Appendix G) are used to display the synthetic image created from the RCS data. A complete description of each subroutine is given with their listing in the Appendices. Radar and Scatterer parameters used in this thesis are in Appendix H.

For this thesis, four scatterers were simulated and their returns processed. A representation of the target (four scatterers) and the radar set is shown in Figure 7. A sample "image" of the four scatterers as "processed" by the Syracuse Research Corporation's computing system using this simulation program with FFT6 is shown in Figure 8 and was used as a standard of comparison for the alternative forms of processing used in this thesis.

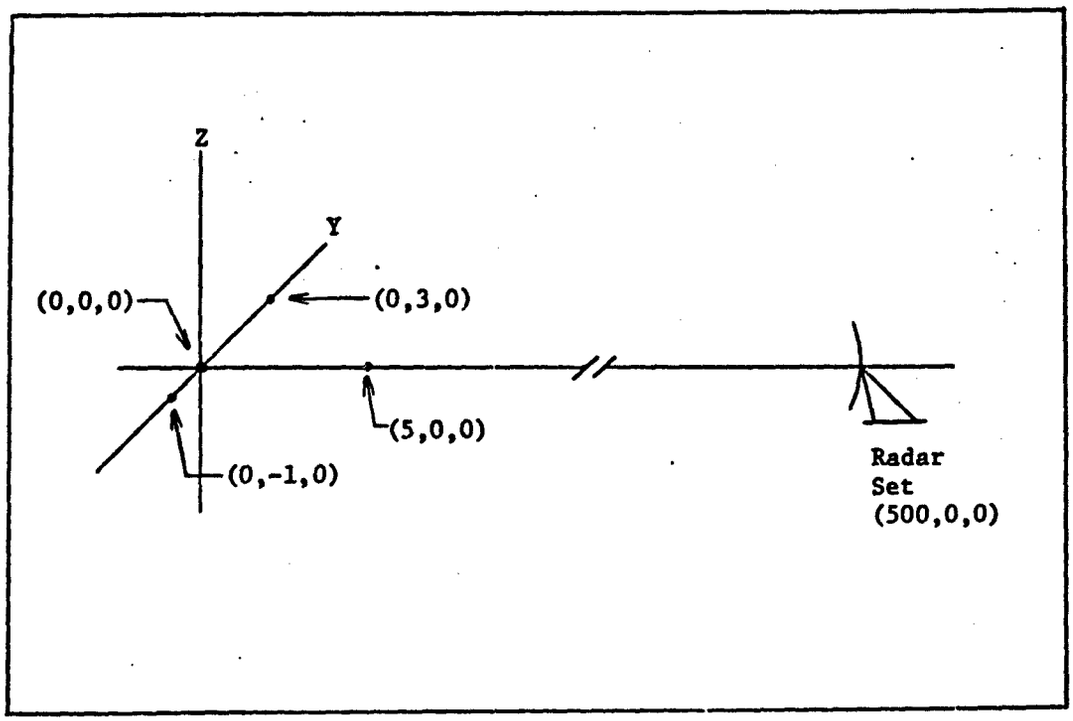



Figure 7. Simulated Target and Radar Configuration

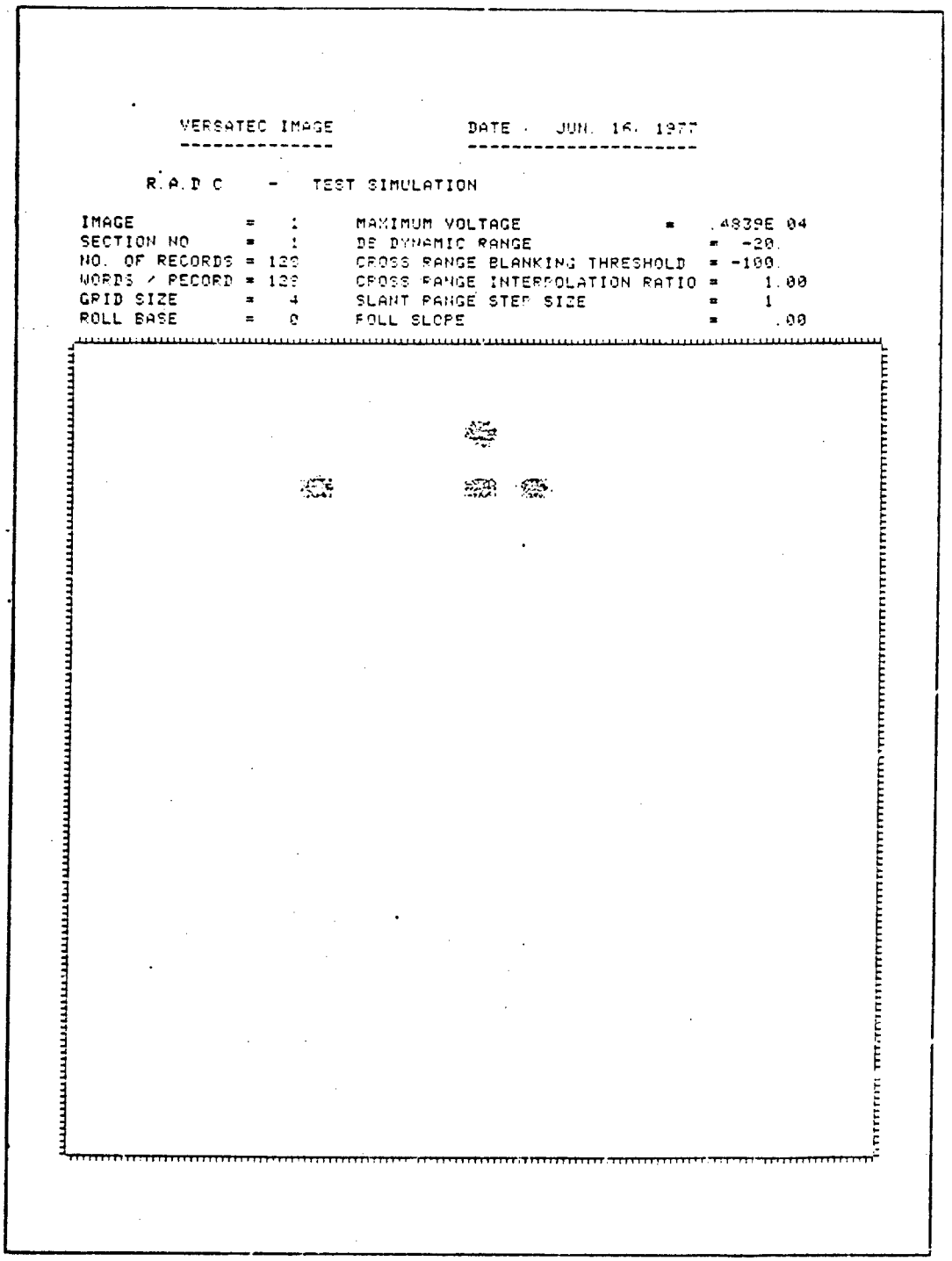



**Figure 8.** Syracuse Research Corporation Generated Sample Image

Subroutine PLOTID (Appendix G) was written to display the generated "image" using a CALCOMP plotter. Each display is generated by comparing each element in the image matrix and plotting a symbol if the value exceeds a threshold level. In this way, only the prominent power spectral pealer or highlights of the scatterers are plotted. Figures 9 through 14 show the images plotted using the Simulation Program TGTID with Subroutine PTOTID and Subroutine FFT6. The threshold set in Figures 9 through 11 are set too "low". The threshold set in Figures 12 and 13 are in the proper range, where the threshold set in Figure 14 is too "high". The threshold level in an actual system would be set dynamically with range information and signal "strength".

All programming is done in FORTRAN IV Extended on the CDC 6600 CYBER 74 System. The plots are generated using a CALCOMP Plotter.

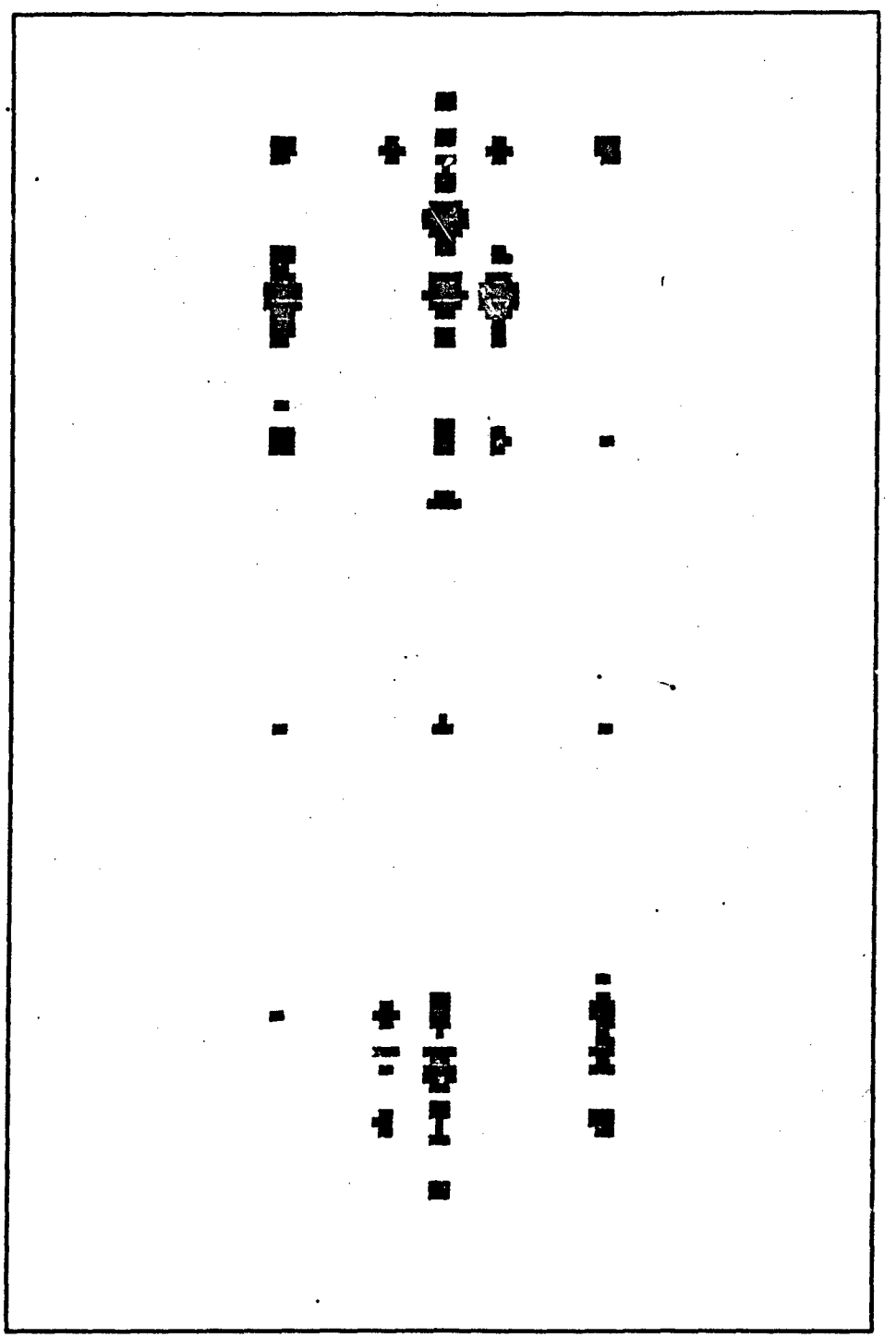
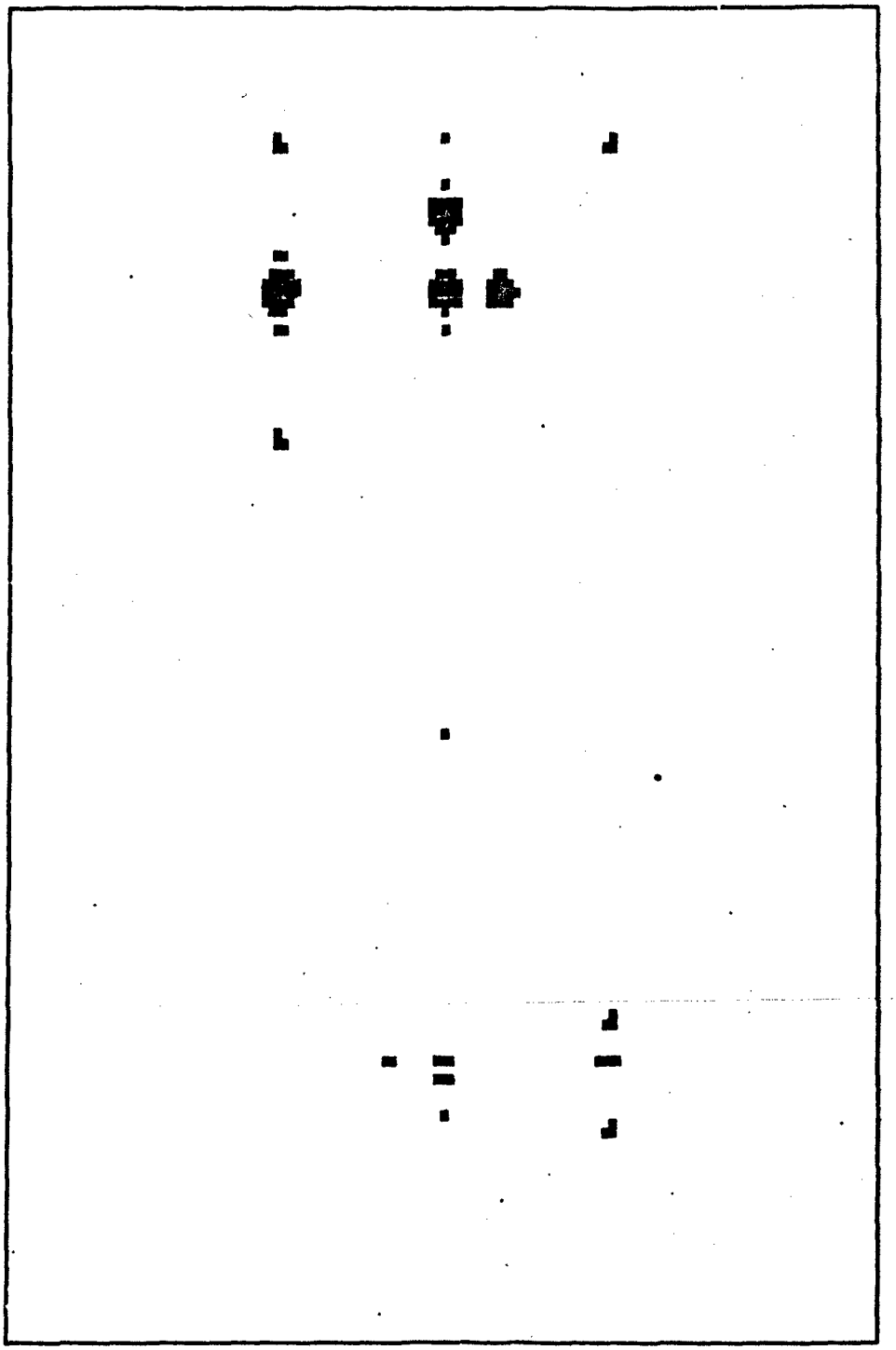


Figure 9. Subroutine FFT6; Threshold = 250.



Figur . 10. Subroutine FFT6, Threshold = 500.

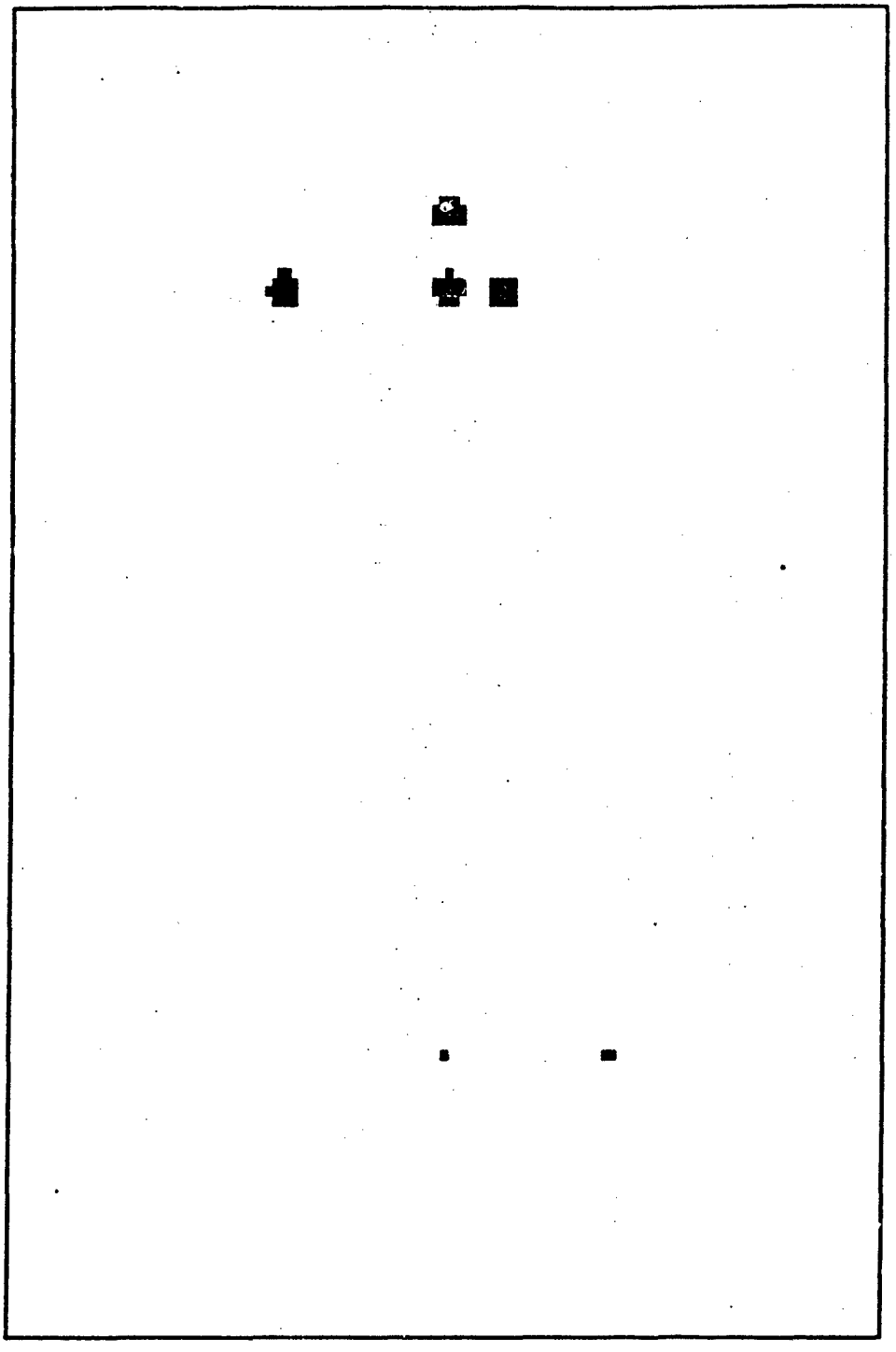


Figure 11. Subroutine FFT6, Threshold = 1000.

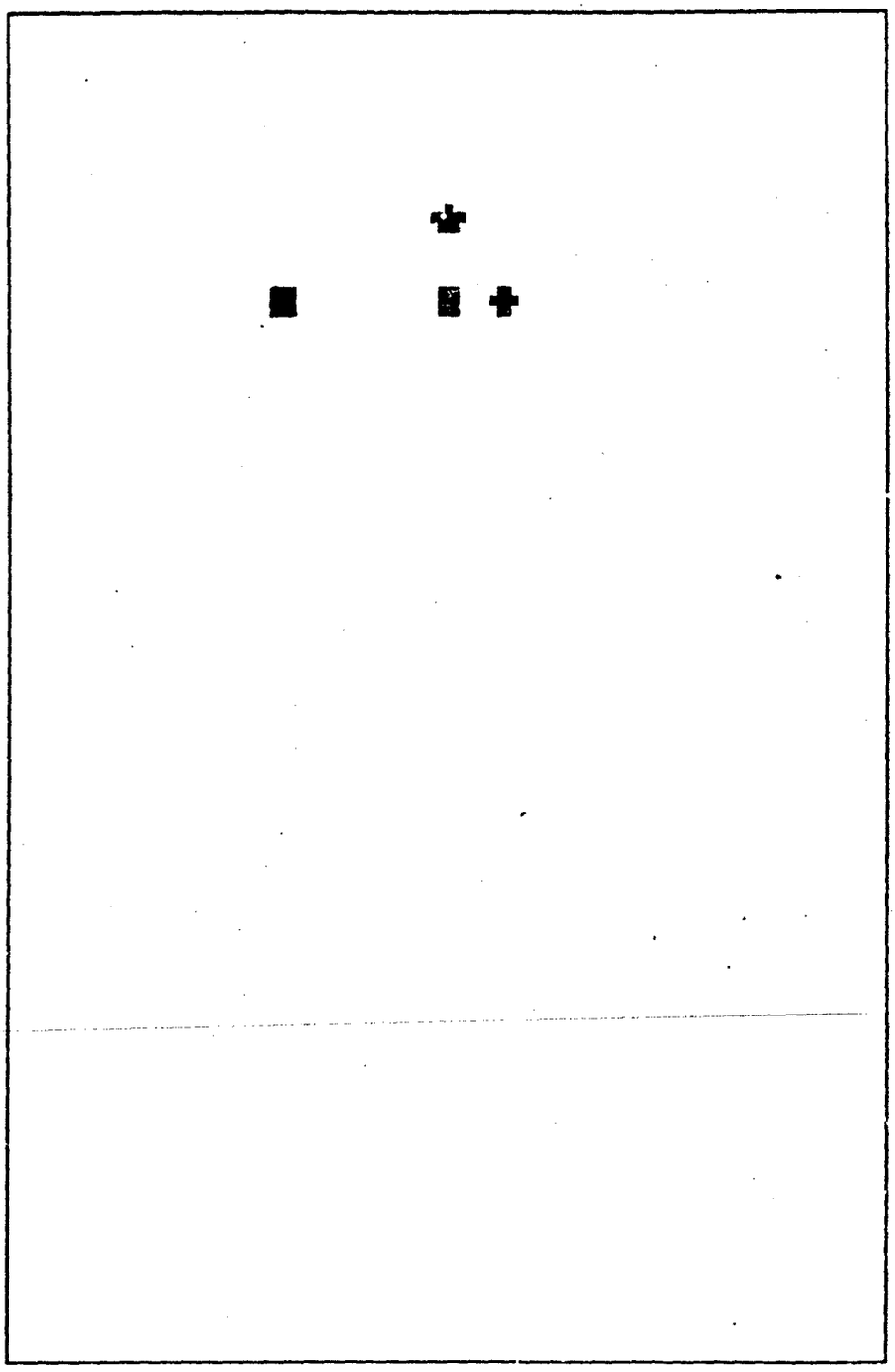


Figure 12. Subroutine FFT6; Threshold = 1200.

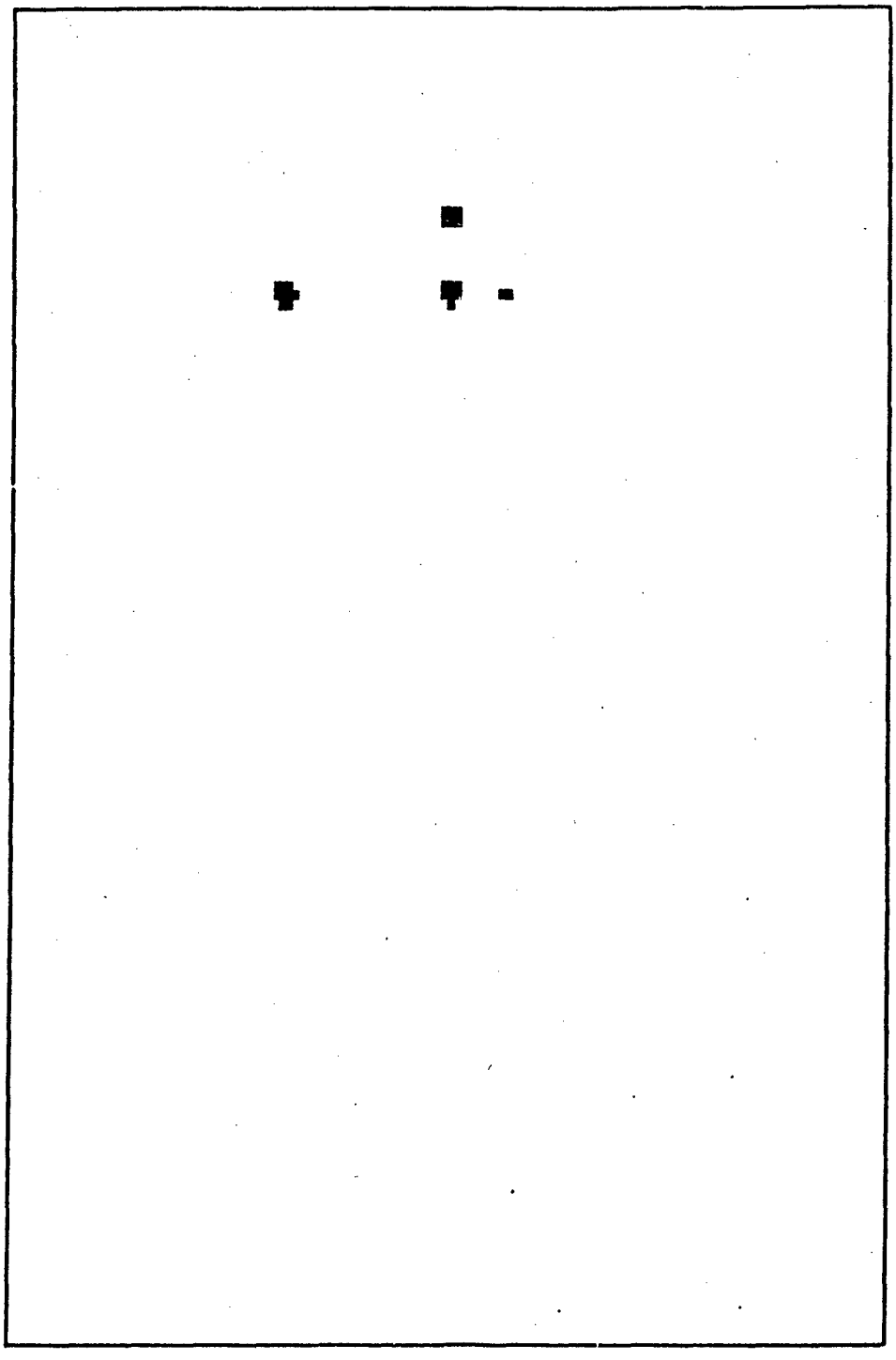


Figure 13. Subroutine FFT6; Threshold = 2000.

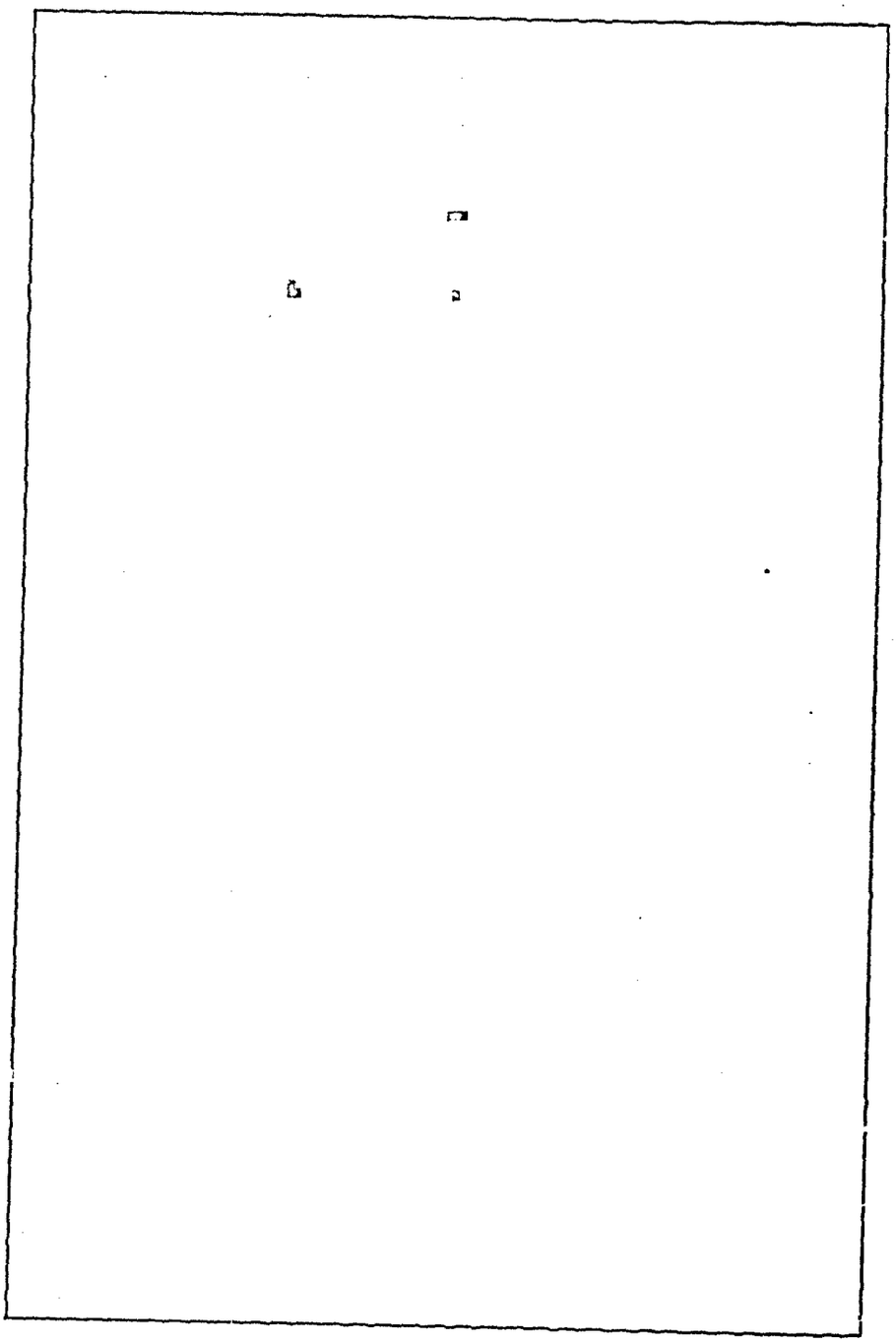


Figure 14. Subroutine FFT6: Threshold = 2500.

## III. Evaluation of Orthogonal Transforms

The use of orthogonal transforms is investigated in this Section. A transform usually possesses some attributes that make it desirable to use in a particular application. These characteristics include: applicability to the problem at hand, mean square error, probability of error, computational advantage, programmable in some computer language, and computer memory required to compute the transform. These characteristics are often used as criteria for determining the acceptability of the transform, especially for implementation on a digital computer.

The Karhunen-Loeve, Cosine (Sine), Mellin, and Hankel transforms are investigated and found not to be suitable for this application. The Walsh (Hadamard) transform is examined in detail because of its similaraties with the Fourier transform, its computational ease and speed, and the existence of an analogous Wiener-Khintchine theorem.

### Karhunen-Loeve Transform

The optimum transform for data compression and for satisfying the minimum mean square error criteria is the Karhunen-Loeve transform (Ref 9:123). Its most common application is found in image and picture transform coding where data compression is highly desired (Ref 48:64, 65). The transform is composed of eigenvectors of the correlation matrix of the original signal, picture, or class of images to be coded (Ref 9:124). The reader is referred to Andrews (Ref 9) for a detailed treatment of the Karhunen-Loeve transform.

There are two major problems associated with the use of the Karhunen-Loeve transform. The first is that a great amount of computation must be performed (Refs 9:125; 10:41). The correlation

matrix must be estimated if it is not known. Next, the correlation matrix must be diagonalized to determine its eigenvalues and eigenvectors. Finally, the transform itself must be taken. In general, there is no fast computational algorithm for the transform. Since it is usually not separable (Ref 9:125) (For example: for N data points, the computational load is $2(n^2)^2$; for 128 points, this is over 500 million multiplications, which is too many to be feasible.) The second difficulty is that the mean square error is not a valid error criterion for most applications (Ref 9:125) including this one.

Cosine (Sine) Transform

The Cosine (Sine) transform has been found to compare favorably with that of the optimal Karhunen-Loeve transform (Refs 4:90; 48:71,72). The Discrete Cosine Transform (DCT) of a data sequence $x(n)$, $n = 0,1,2,\ldots,(n-1)$ is defined as

$$G_x(0) = \frac{\sqrt{2}}{N} \sum_{n=0}^{N-1} x(n) \tag{38}$$

$$G_x(k) = \frac{2}{N} \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)k\pi}{2N} \tag{39}$$

$$k = 1,2,\ldots,(N-1)$$

where N is the total number of samples and $G_x(k)$ is the k th DCT coefficient (Ref 4:90). Ahmed et al. state that Equation 39 can be expressed as

$$G_x(k) = \frac{2}{N} \text{Re} \{\exp((-jk\pi)/2N) \sum_{n=0}^{2N-1} x(n) W_N^{kn}\} \tag{40}$$

where $W = \exp(-j2\pi/2N)$, $j = \sqrt{-1}$, $x(n) = 0$ for $m = N, (N+1),\ldots,(2N-1)$, and $\text{Re}\{\cdot\}$ implies the real part of the term enclosed (Ref 4:91). From Equation 40 it follows that the N DCT coefficients can be computed

using a 2N-point fast Fourier transform (Ref 4:91). Similarly, if a discrete Sine transform were desired, the Re{·} in Equation 40 would be replaced by Im{·}, which denotes the imaginary part of the term enclosed. It can be seen that the computational speed of the DCT or DST is slower as that of the FFT since twice as many points must be transformed. No other fast transform algorithm currently exists for the DCT or the DST.

Mellin Transform

The Mellin transform possesses the unique property of "scale" invariance (Ref 22:78). That is, scale changes in the input do not produce scale changes in the output. The Mellin transform M(u) of a function f(x) is defined by

$$M(u) = \int_0^\infty f(x)\, x^{-u-1} dx \tag{40}$$

and the inverse transform is given by

$$f(x) = \frac{1}{j\pi} \int_{\gamma-ia}^{\gamma+<\infty} M(u)\, x^{-u} du \tag{41}$$

where $\gamma$ is chosen so that the integral exists (Ref 83:13). The discrete Mellin transform is given by

$$M(k\Delta u) = \sum_{i=1}^{N} f(i\Delta x)(i\Delta x)^{jk\Delta u-1}\, \Delta x \tag{42}$$

where N is the number of samples of f(x), and the input and transform space resolutions are $\Delta x$ and $\Delta u$ respectively (Ref 22:79).

No fast computational algorithm has yet been found for implementing the Mellin transform directly. Processing time for a digital computer can be quite long (Ref 22:80). Casasent and Psaltis point out that a digital Mellin transform can also be realized by exponentially sampling

the input and then performing an FFT on this data (Ref 22:78,80). It is concluded, therefore, that the Mellin transform offers no advantage over the FFT processing currently being performed in the simulation program. For a further comment on this subject see Chapter VI.

Hankel Transform

The Hankel transform is useful if symmetry exists about an axis and if polar coordinates are appropriate (Ref 85:46). The Hankel transform pair are defined by

$$F(k) = \int_0^\infty f(x)\ J_n(kx)\ dx \tag{43}$$

and

$$f(x) = \int_0^\infty F(k)\ J_n(xk)\ dk \tag{44}$$

where $J_n(kx)$ is the Bessel function of the first kind of order n (Ref 85:46). This application can be seen to demonstrate some elements of symmetry about an axis, say the aspect angle of the body at T/2 of the observation interval, but no method to convert the received Doppler waveform to polar coordinates could be found. Additionally, this transform requires the calculation of certain boundary conditions which is a very difficult problem (Ref 85:46). It was concluded that the Hankel transform could not be applied to this problem.

Walsh Functions and Walsh Transforms

Walsh functions are a complete orthonormal set of square wave functions that are finding increasing use in various digital signal processing applications (Ref 29:197, Ref 44). They exhibit similarities to the trigonometric sine and cosine functions in many of their

properties (Refs 87; 4; 5). The bivalued characteristic, orthogonality, and computational advantages of these functions are the basis and motivation for their detailed study in this section.

Definition. Continuous Walsh functions may be defined in several ways (Ref 52:211). They may easily be defined as products of Rademacher functions (Refs 52:212; 84:4).

The Rademacher functions (Refs 29:177; 84:4) are defined by

$$R_0(\theta) = \begin{cases} 1, & 0 \leq \theta \leq \frac{1}{2} \\ -1, & \frac{1}{2} \leq \theta \leq 1 \end{cases} \tag{45}$$

$$R_0(\theta + 1) = R_0(\theta) \tag{46}$$

$$R_n(\theta) = R_o(2^n\theta), \quad n = 1,2,3,\ldots \tag{47}$$

Figure 15 shows the first five Rademacher functions (Ref 55:39).

To form the Walsh function wal(n,θ), first, form the binary representation of n, then form the Gray code version of n, and multiply together Rademacher functions according to the 1 bits in the Gray code (Ref 52:212).

If n in binary is

$$N = b_m b_{m-1} b_{m-2} \cdots b_o \tag{48}$$

then n in Gray code is

$$n = g_m g_{m-1} g_{m-2} \cdots g_o \tag{49}$$

where

$$g_m = b_m \tag{50}$$

and

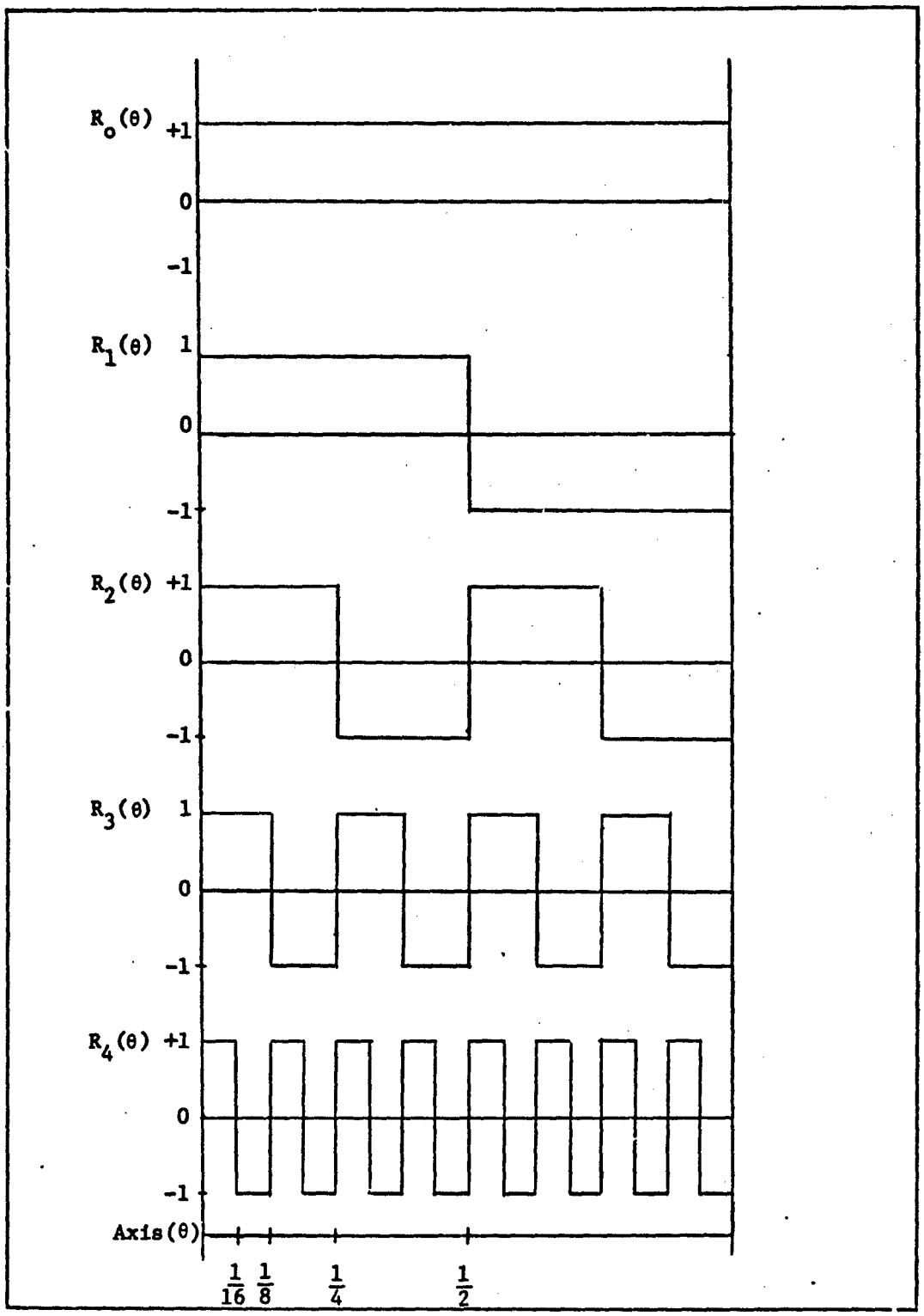



Figure 15. The First Five Rademacher Functions, $R_n(\theta)$.

$$g_i = b_i \oplus b_{i+1}, \qquad i = 1,2,\ldots,m-1 \tag{51}$$

where $\oplus$ is modulo-2 addition with no carries (Ref 52:212). The Walsh functions can now be defined by

$$wal(0,\theta) = 1 \tag{52}$$

$$wal(n,\theta) = R_m^{g_m}(\theta)\, R_{m-1}^{g_{m-1}}(\theta) R_{m-2}^{g_{m-2}}(\theta) \ldots R_o^{g_o}(\theta) \tag{53}$$

where n is the order and $\theta$ is normalized time (Ref 84:4). This can also be written as

$$wal(n,\theta) = \sum_{k=0}^{m} g_k R_k(\theta) \tag{54}$$

where the summation symbol denotes modulo-2 summing (Ref 29:187). For example:

$$6_{10} = 1110_2 = 101 \tag{55}$$

$$wal(6,\theta) = R_2^1(\theta) R_1^0(\theta) R_0^1(\theta) \tag{56}$$

$$= R_2(\theta) R_0(\theta) \tag{57}$$

This procedure to find a particular Walsh function is easily remembered. It should be noted that the first argument of a Walsh function denotes its "sequency". Sequency, as defined by Harmuth, is the number of zero crossings or sign changes of the Walsh function in the half-open interval (0,1) (Ref 44:50).

Harmuth uses the notation Wal(j,$\theta$) to define the Walsh function and further defines

$$Wal(0,\theta) = 1 \tag{58}$$

$$Cal(n,\theta) \triangleq Wal(2n,\theta) \tag{59}$$

$$Sal(n,\theta) \triangleq Wal(2n-1,\theta) \tag{60}$$

where the Cal and Sal names are used to parallel the cosine and sine functions (Ref 44:22). Several notation schemes are used in the literature and are summarized by Meck (Ref 55:11). The first eight continuous Walsh functions are shown in Figure 16.

Discrete Walsh Functions. Discrete Walsh functions are sampled versions of the continuous set (Ref 77:457). Shanks assumes that the discrete functions are infinite in extent, and are periodic with period N, where N is an integral power of two (Ref 77:457). Thus a complete orthogonal set will have N distinct functions, designated as wal(n,m). The complete set is represented over the range n = 0,1,...,N-1 and m = 0,1,...,N-1. The first two discrete Walsh functions are defined as

$$wal(0,m) = 1,\ n = 0,1,2,\ldots,N-1 \tag{61}$$

$$wal(1,m) = 1, m = 0,1,2,\ldots,(N/2)-1 \tag{62}$$

$$= -1, m = N/2,\ (N/2)+1,\ldots,N-1 \tag{63}$$

Various iterative equations have been used to generate the remainder of the set, but Henderson's seems to be the most convenient

$$wal(n,m) = wal([N|2],2m)\ .\ wal(N-2[N/2],m) \tag{64}$$

where [N/2] indicates the integer part of N/2 (Refs 45:51; 77:457).

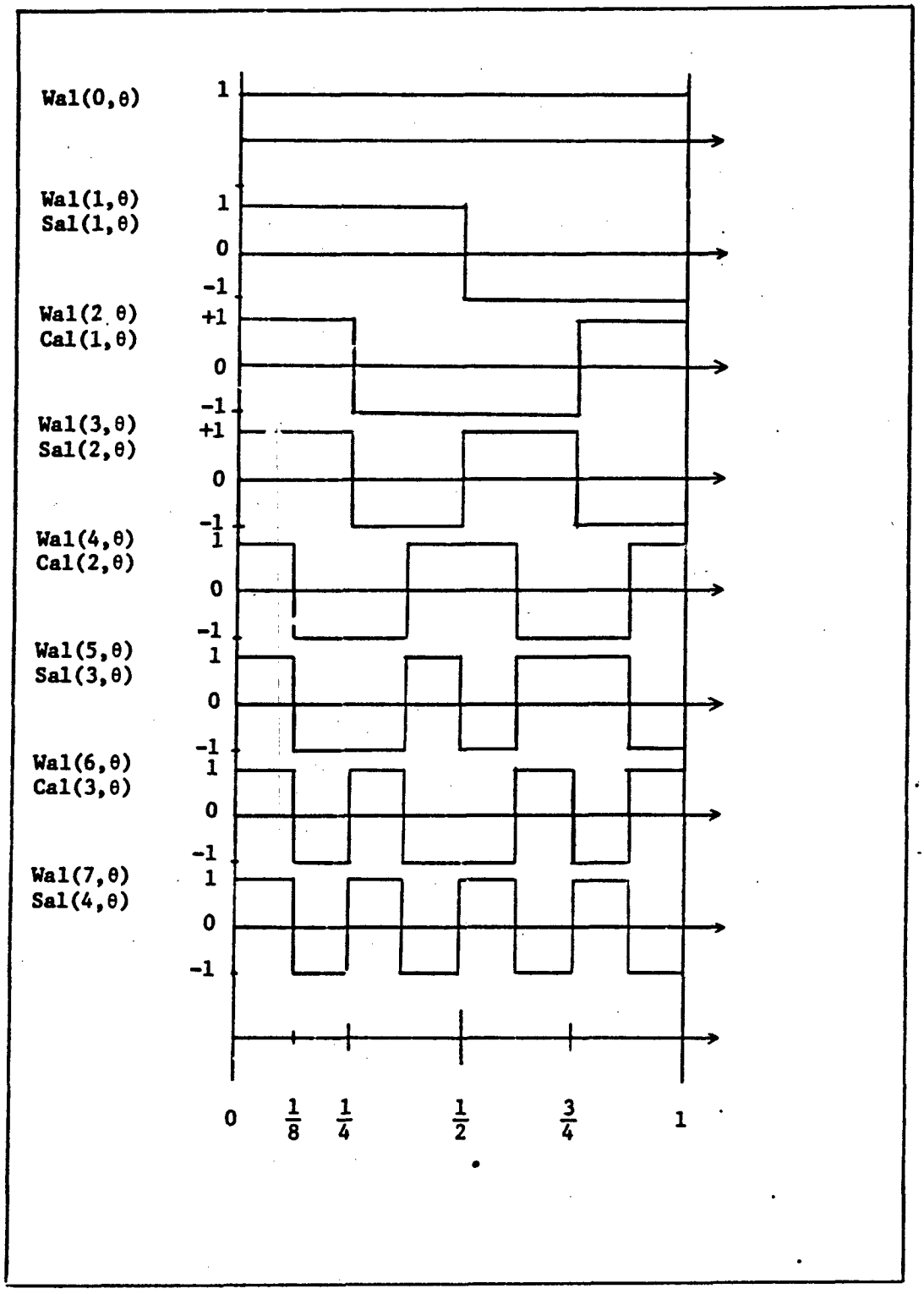



Figure 16. The First Eight Walsh Functions, W(n,θ).

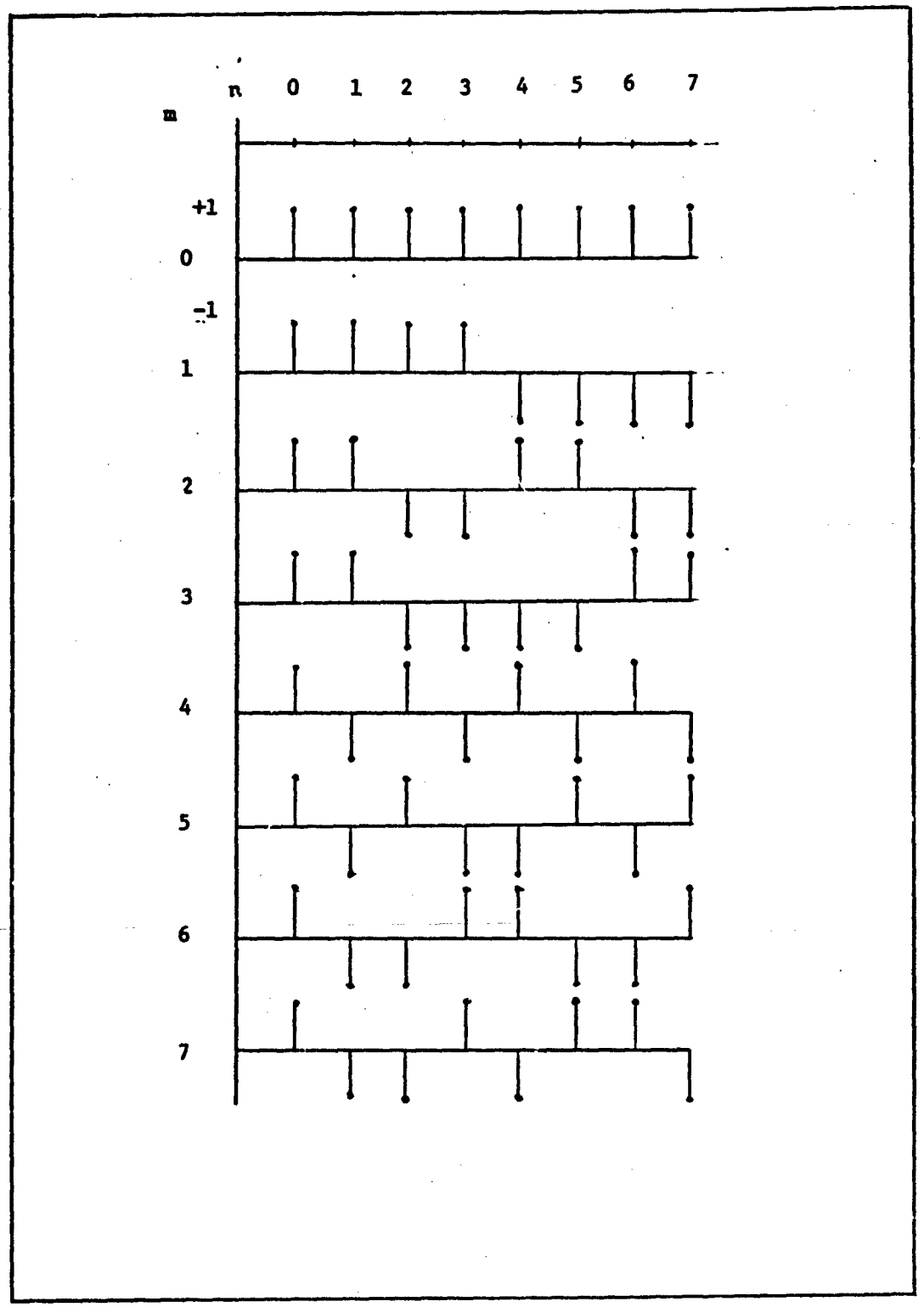



Figure 17. The First Eight Discrete Walsh Functions of Length 8.

Shanks shows that this recursive equation generates a complete orthogonal set (Ref 77:459). Figure 17, shows the first eight discrete Walsh functions of length eight generated by Equations 61, 62, and 63 (Ref 77:457).

The Walsh functions may also be represented in matrix notations. Let $N = 2^k$, where k is a positive integer, then the N th order Walsh matrix is constructed by sampling the first N Walsh functions once in N equal subintervals of (0,1) (Ref 87:5). The matrix constructed from sampling the sequency ordered Walsh functions (Figure 16) results in a sequency ordered NxN Walsh matrix. Figure 18a, shows the sequency ordered Walsh matrix for N = 8.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 \end{bmatrix}$$

a. Sequency Order b. Natural" Order

Figure 19. Walsh Matrices of Order N = 8.

A second, simpler, method which generates Walsh matrices in what is termed "natural" order is a Kroneker product of p second order Walsh matrices (Refs 85:5; 78:177). This form or the Walsh matrix is often referred to as the "Hadamard" matrix. A second order Walsh matrix is defined by

$$W_2 = H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{65}$$

Hadamard matrices of higher order, for N a power of two, are generated by the Kroneker product operation, such that

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix} \tag{66}$$

Figure 18b, shows the natural ordered Walsh matrix or Hadamard matrix for n = 8.

Both the Walsh matrix and the Hadamard matrix are square arrays, whose rows and columns are orthogonal to each other, that is, the product of the matrix and its transpose is the identity matrix times N, where N is the order of the matrix (Ref 78:177).

$$H \cdot H^T = N \cdot I \tag{67}$$

Walsh Transform. Since the Walsh functions form a complete, orthonormal set over the interval (0,1), any absolutely integrable function defined over the interval can be expanded into a series of Walsh functions analogous to the Fourier expansion of such a function (Ref 44:45). The discrete Walsh transform of a function is defined by

$$F(m) = \frac{1}{N} \sum_{n=0}^{N-1} f(n)\ wal\ (m,n) \tag{68}$$

$$m = 0,1,2,\ldots,N-1$$

and the inverse transform is given by

$$f(n) = \sum_{m=0}^{N-1} F(m)\ wal(n,m) \tag{69}$$

$$n = 0,1,2,\ldots,N-1$$

where F(m) is the m th normalized Walsh coefficients, f(n) is the discrete input vector, and wal(m,n) is the m th Walsh function (Ref 77:457). It should be noted that since the Walsh matrix is orthogonal, the following relationship holds

$$wal(n,m) = wal(m,n) \tag{70}$$

Using matrix notation, the Walsh transform matrix equation is given by

$$[A] = \frac{1}{N} [W]\ [F] \tag{71}$$

where [F] is a column vector of sample values of the input signal, [W] is the Walsh matrix, and [A] is the column vector or Walsh coefficients (Ref 84:7).

Similarly, the "Hadamard" transform is given by

$$[A] = \frac{1}{N} [H]\ [F] \tag{72}$$

Where [H] is the Walsh matrix in natural order (Ref 78:178).

The function F(m) or [A], therefore, represents the "sequency" spectrum of f(n) in the same sense that a set of Fourier coefficients represents a frequency spectrum (Ref 44:51,52).

The computational load for either Equation 71 or Equation 72 can be seen to be N(N-1) additions (multiplication by -1 is not really a multiplication but just a "sign" change and an addition).

However, Good has developed a matrix factorization technique which leads to a "fast" transform algorithm (Ref 8:16, 17). Good's technique can be used to factor Kroneker matrices such as in Figure 18a and b. Matrices of order $N = n^p$ can be factored into p matrices of order N (Ref 8:17). If the matrix to be factored has been generated by the Kroneker product of identical matrices, then its factors will also be identical (Ref 84:8). The factors of a Walsh matrix of order $N = 8 = 2^3$ are shown in Figure 19 (Ref 84:9). Since the matrix factors include many zero elements, the number of computations is reduced to $N\log_2 N$ additions (Ref 84:8).

A flow diagram of the Fast Walsh transform is given in Figure 20 (Ref 84:9). Similarly, a flow diagram for the Hadamard transform is shown in Figure.21 (Ref 76:178). The recursive structure of the diagrams leads to an efficient programming of the algorithm on a digital computer (Refs 78:179; 84:8; 1:276). The coefficients of the Fast Walsh transform (FWT) are in "bit reflected" order and those of the Fast Hadamard transform (FHT) are in sequency order. Therefore, the FWT requires a reordering procedure which adds approximately 15 to 20% more to execution time (Ref 51:204). Several algorithms exist for converting from bit reflected order to sequency order (Ref 53:16).

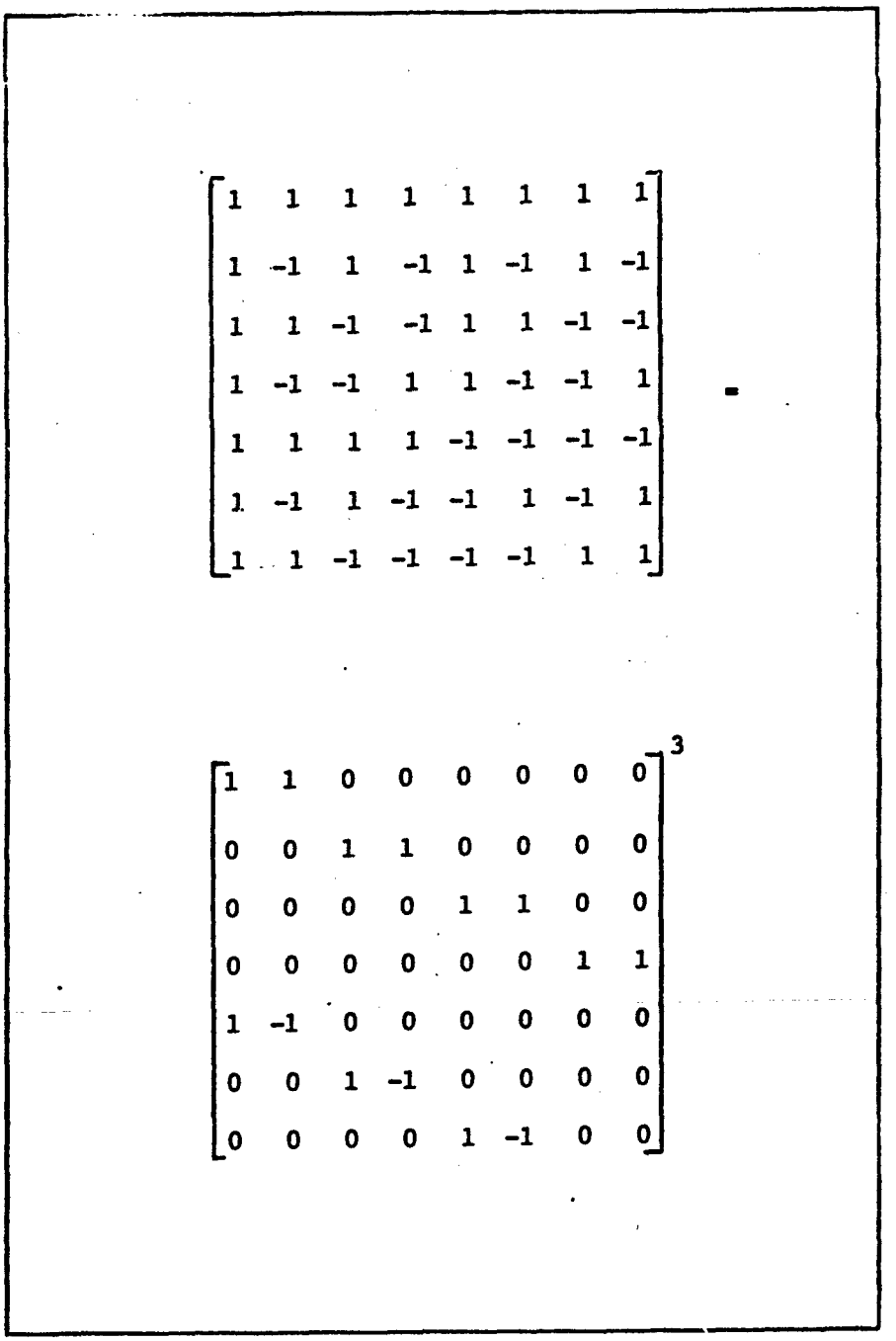


Figure 19. Factorization of the Naturally Ordered Walsh Matrix of Order Eight.

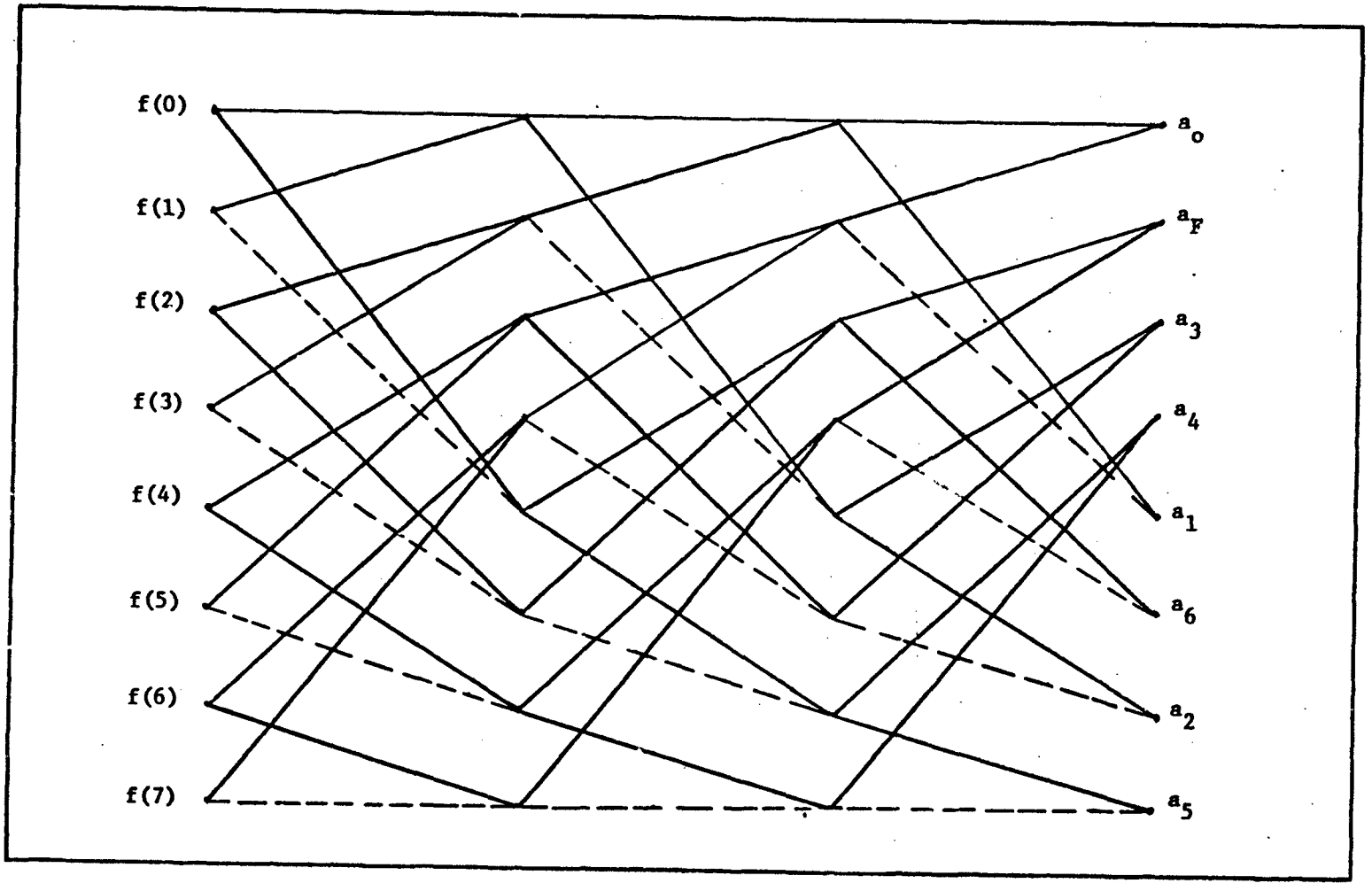



Figure 20. Signal Flow Graph For Fast Walsh Transform for N = 8.

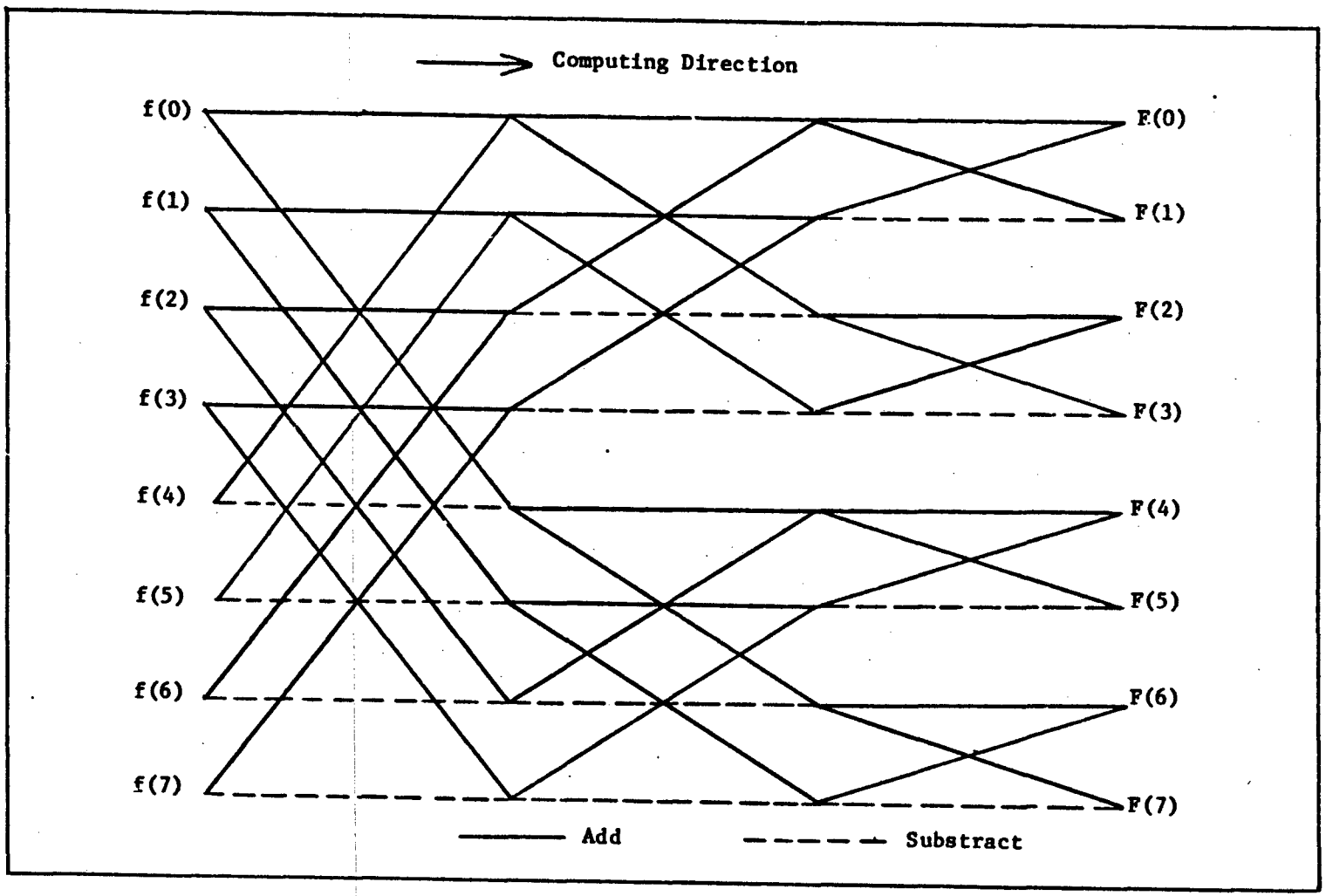



Figure 21. Signal Flow Graph for Fast Hadamard Transform for N = 8.

Dyadic Convolution. The dyadic convolution of two functions f(t) and g(t) is defined as

$$h(t) = f \oplus g$$

$$= \int_{-\infty}^{\infty} f(\tau)g(t \oplus \tau)d\tau \tag{73}$$

where $\oplus$ denotes dyadic or logical convolution and $t \oplus \tau$ denotes addition modulo-2 (Ref 41:616-617). The discrete dyadic convolution of two sequences f(n) and g(n) of length N is defined by

$$h(n) = \frac{1}{N} \sum_{i=0}^{N-1} f(i)g(n \oplus i), \; n$$

$$= 0,1,\ldots,N-1 \tag{74}$$

Logical Wiener-Khintchine Theorem. If the Walsh transform of f(n) and g(n) is defined as F(s) and G(s), respectively, then the following property is true

$$h(n) = f \oplus g \xleftrightarrow{W} F(s) \cdot G(s) \tag{75}$$

where s represents sequency (Ref 55:19). The time-sequency domain logical Wiener-Khintchine theorem is defined as

$$R(\;) = f \oplus f \xleftrightarrow{W} F(s) \cdot F(s)$$

$$= F(s)^2$$

$$= P(s) \tag{76}$$

which is analogous to the "arithmetic" Wiener-Khintchine theorem of the time-frequency domain (Refs 55:19; 72:271; 1:615).

Power Spectral Density. The sequency power spectrum, P(s), is easy to generate from the Hadamard transform and has a physical interpretation quite similar to the frequency power spectrum (Ref 44:51). The sequency power spectrum, as defined by Harmuth is given by

$$P(s) = \begin{cases} A_0^2 & k = 0 \\ A_{2k-1}^2 + A_{2k}^2 & k = 1,2,\ldots,N/2-1 \\ A_{2k-1}^2 & k = N/2 \end{cases} \qquad (77)$$

where $A_{2k-1}$ and $A_{2k}$ are the odd and even sequency discrete Walsh transform coefficients (Refs 44:51; 64:93). However, this spectrum is not invariant to time shifts (Refs 64:92; 41:617). Polge et al. state that the variation of the sequency power spectrum with the time axis position of th input data is a serious drawback in signal processing activities unless only gross spectral features are desired or the possibility of time synchronization exists (Ref 64:93). Andrews and Caspari, in their work on generalized spectral analysis, demonstrate the "shift variance" of the Walsh transform relative to the "shift invariant" Fourier transform (Ref 8:24). Figure 22a shows the spectrums of a block pulse and Figure 22b shows the spectrums of the block pulse shifted relative to the time origin (Ref 8:24).

A time shift invariant power spectrum can be generated by defining the power spectrum as the Hadamard transform of the autocorrelation function (Equation 76) and noting that the autocorrelation function is invariant to time shifts (Ref 64:93). If F(s), s = 0,1,...,N-1, is the Hadamard transform of the sequence f(n), then the time invariant

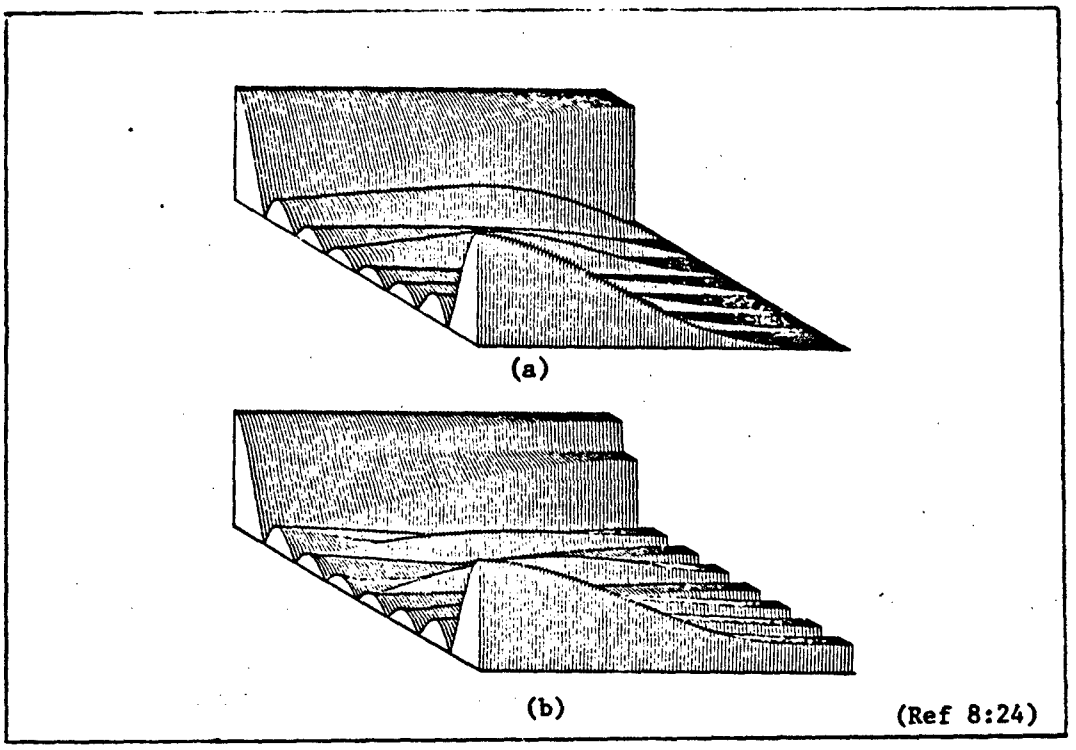



Figure 22. (a) Orthogonal Decomposition of a Block Pulse (Fourier to Walsh Transition). (b) Orthogonal Decomposition of a Shifted Block Pulse (Fourier to Walsh Transition).

power spectrum is

$$P_{TI}(s) = \sum_{s=0}^{N-1} Q_{js} F(s) \qquad (79)$$

where the matrix Q, made up of elements $Q_{js}$, is dependent on F(s) (Ref 62:93). The computation of the power spectrum using the autocorrelation function is time consuming, and the high speed advantage of the Hadamard transform is lost (Ref 62:93).

Simulation Program Test. The possibility of using the Walsh transform in the simulation program TGTID was investigated by substituting the FFT subroutine with a fast Walsh transform.

The test was made with S broutine FHT1 (Appendix D) substituted for Subroutine FFT6 (Appendix C). Appropriate modifications were made to Subroutine CIMAGE (Appendix B, Version 2), and Subroutine ROLL (Appendix B) was replaced by Subroutine WROLL (Appendix B). The "standard" Walsh power spectrum was computed, and each spectral element of the resultant image matrix was compared with a threshold and plotted if it exceeded the threshold. The threshold was varied to determine its effect on the image. The results of the first test are shown in Figure 23 - 27, beginning on page 53. It can be seen that the image does not look entirely like that of an image generated using a Fourier transform. The effect of time shifting the input was tested by "repositioning" the four point scatterers (Appendix H, Data Set 2). A new set of images were constructed and are shown in Figures 28 - 32, beginning on page 58. It can be seen that the scatterer "information" has changed and that the two sets of images have only little similarity. The second data set was used with the FFT subroutine reinserted into the simulation program. It ca be seen that there is no change in the image, except in its location in the "viewing field", and the result is shown in Figure 33, on page 63 .

This difficulty, as Blachman observes, is attributed to the fact that after time shifting, a Walsh function generally becomes the sum of an infinite number of Walsh functions while a sinusoid simply turns into the sum of a sine and cosine of the same frequency. Thus a change in time scale, or a shift of the time origin usually will grossly alter a Walsh spectrum but has no effect on the Fourier spectrum (Ref 16:347).

It is concluded that since the Walsh power spectrum computed by the direct method is not time shift invariant and since no fast algorithm exists for the second method and since there is little possibility of time synchronization, the Walsh transform can not be utilized in this application.

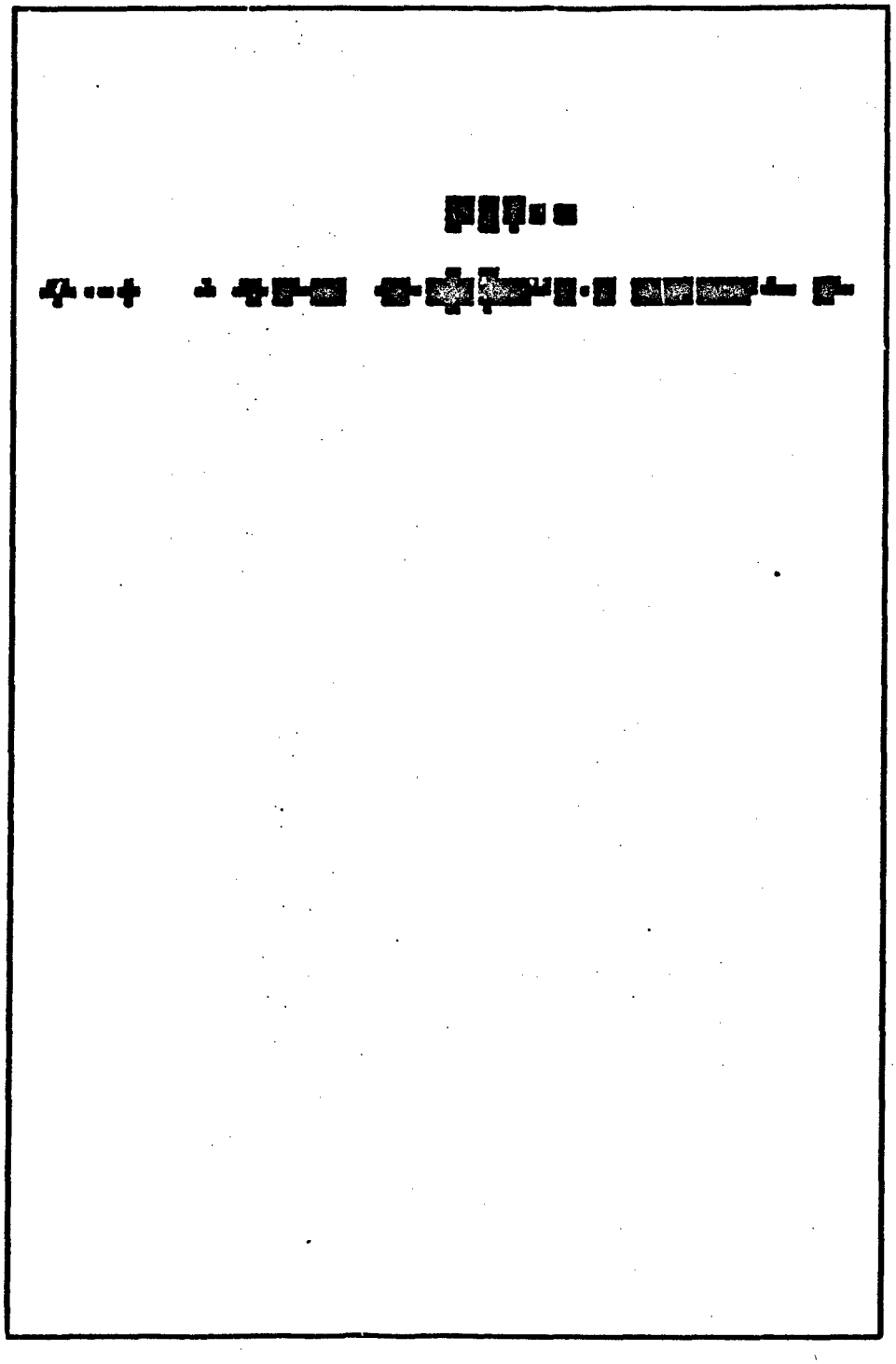


Figure 23. Subroutine FHT1, Data Set 1; Threshold = 1.

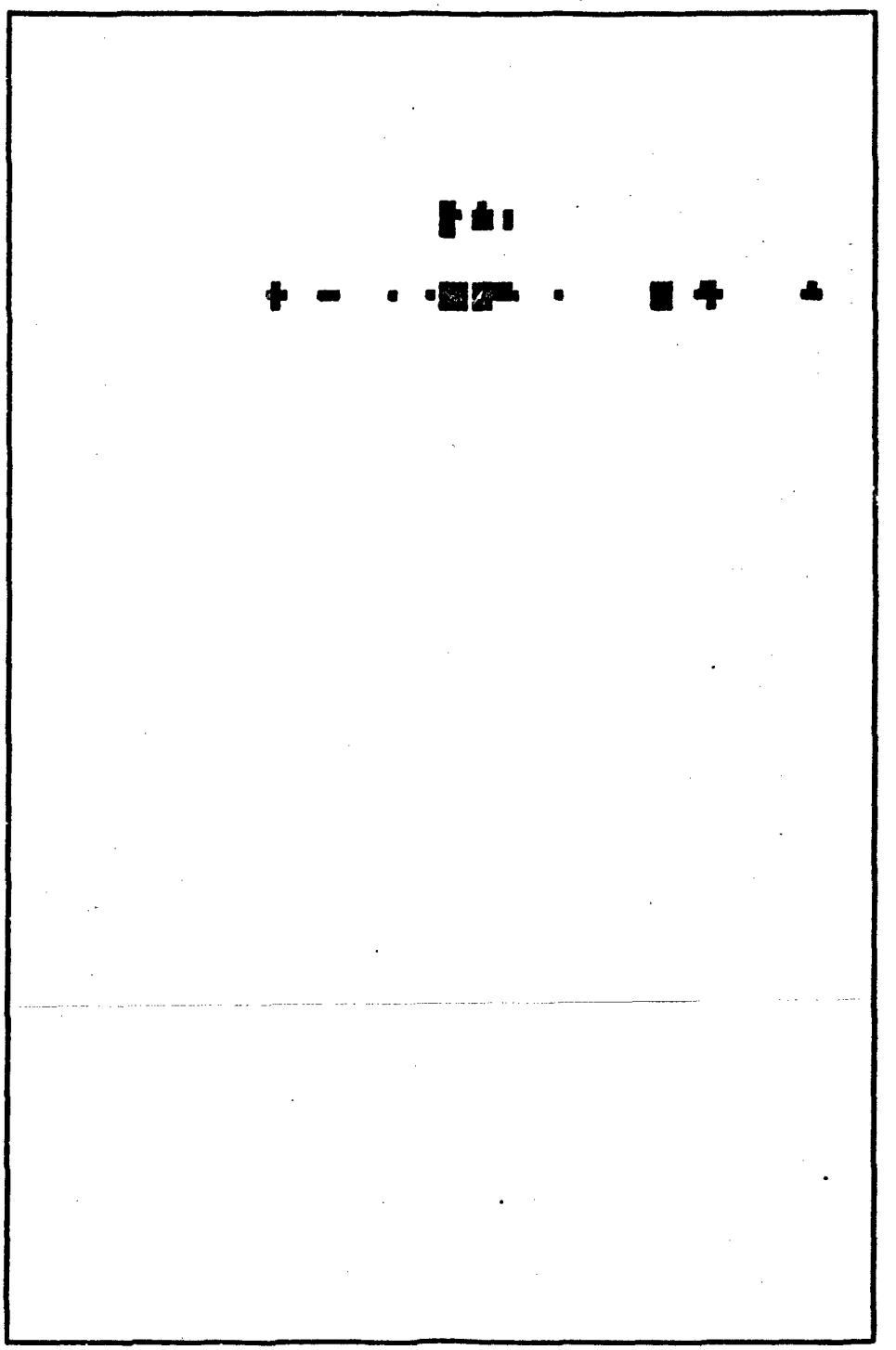


Figure 24. Subroutine FHT1, Data Set 1; Threshold = 10.

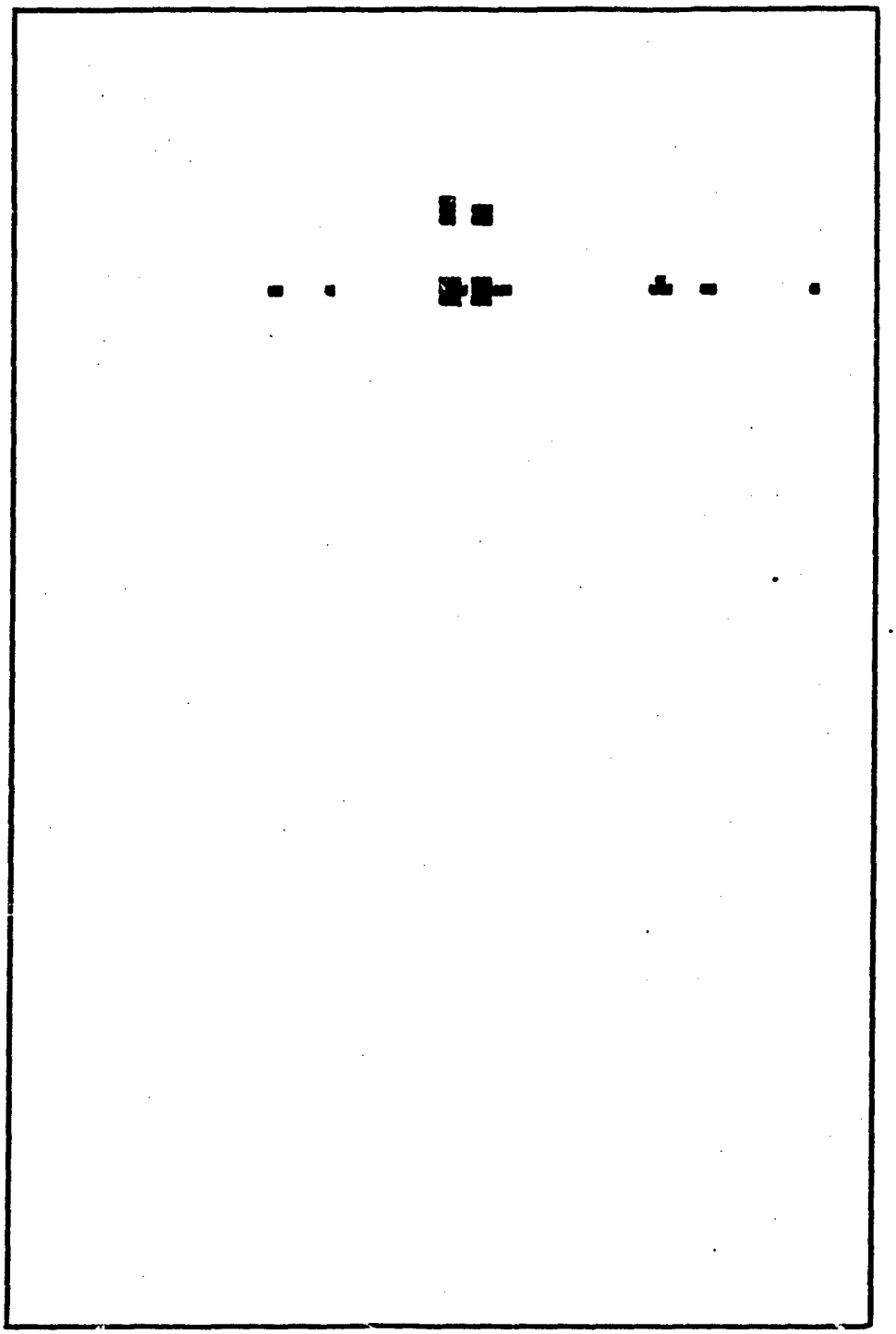


Figure 25. Subroutine FHT1, Data Set 1; Threshold = 20.

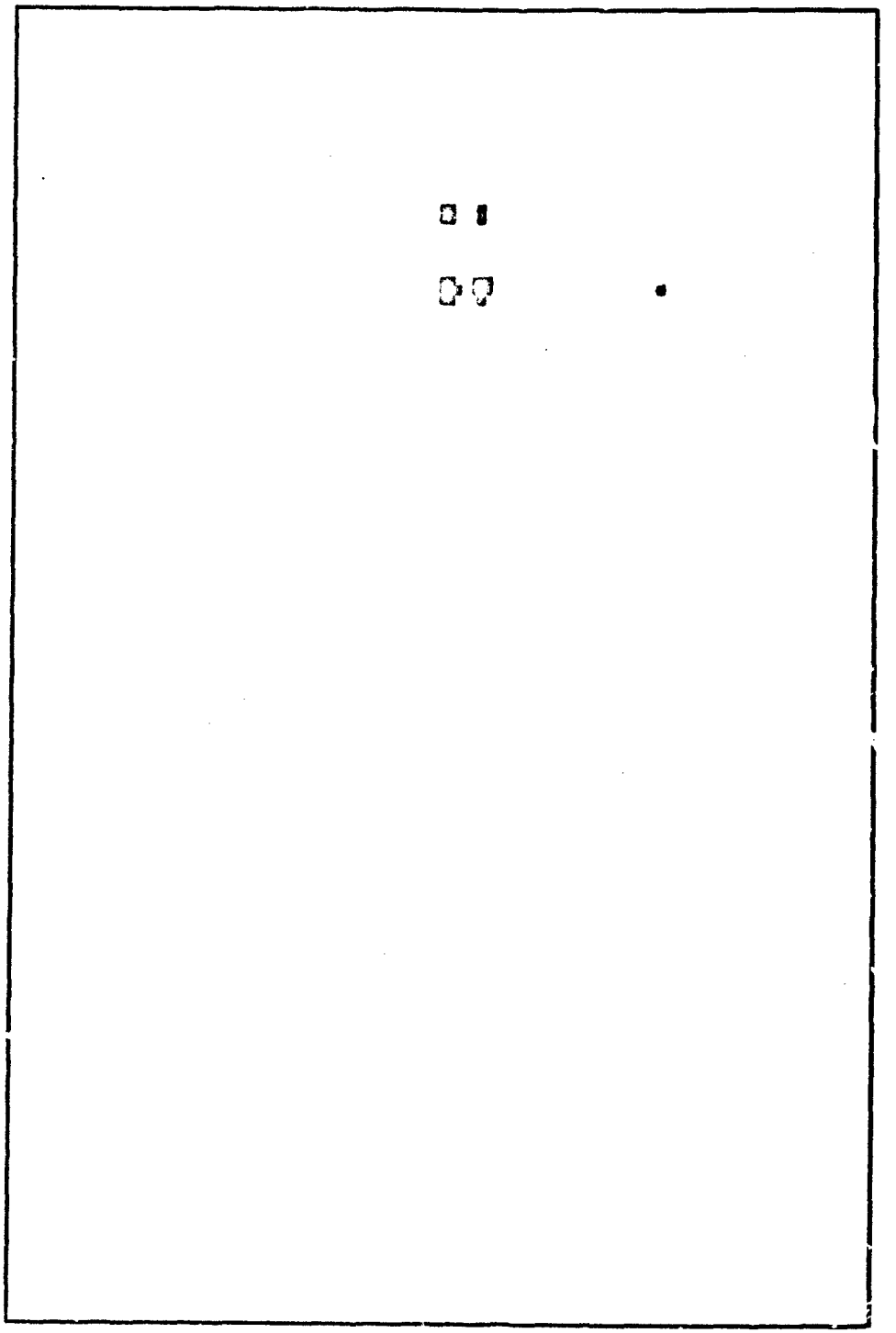


Figure 26. Subroutine FHT1, Data Set 1, Threshold = 50.

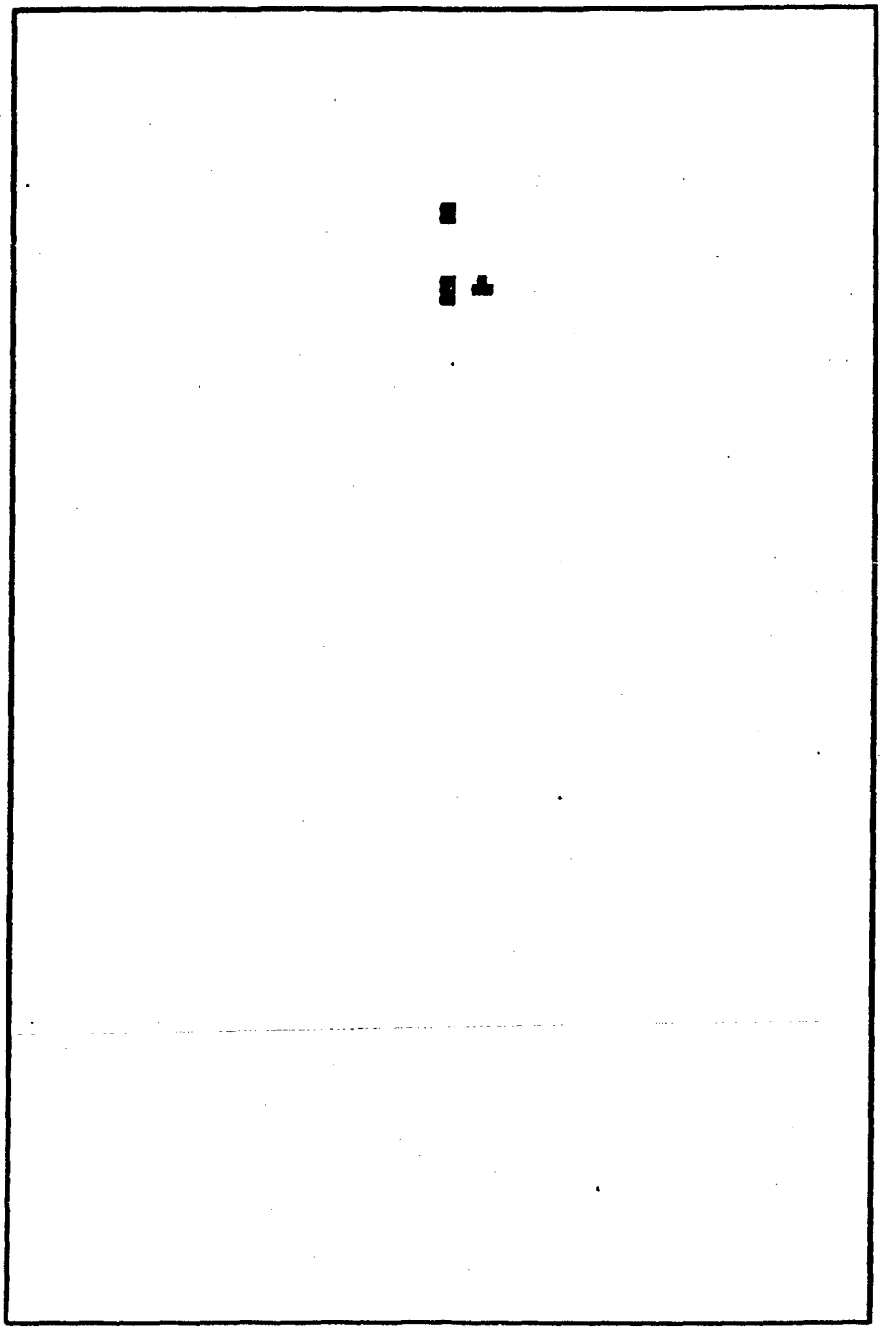


Figure 27. Subroutine FHT1, Data Set 1; Threshold = 100.

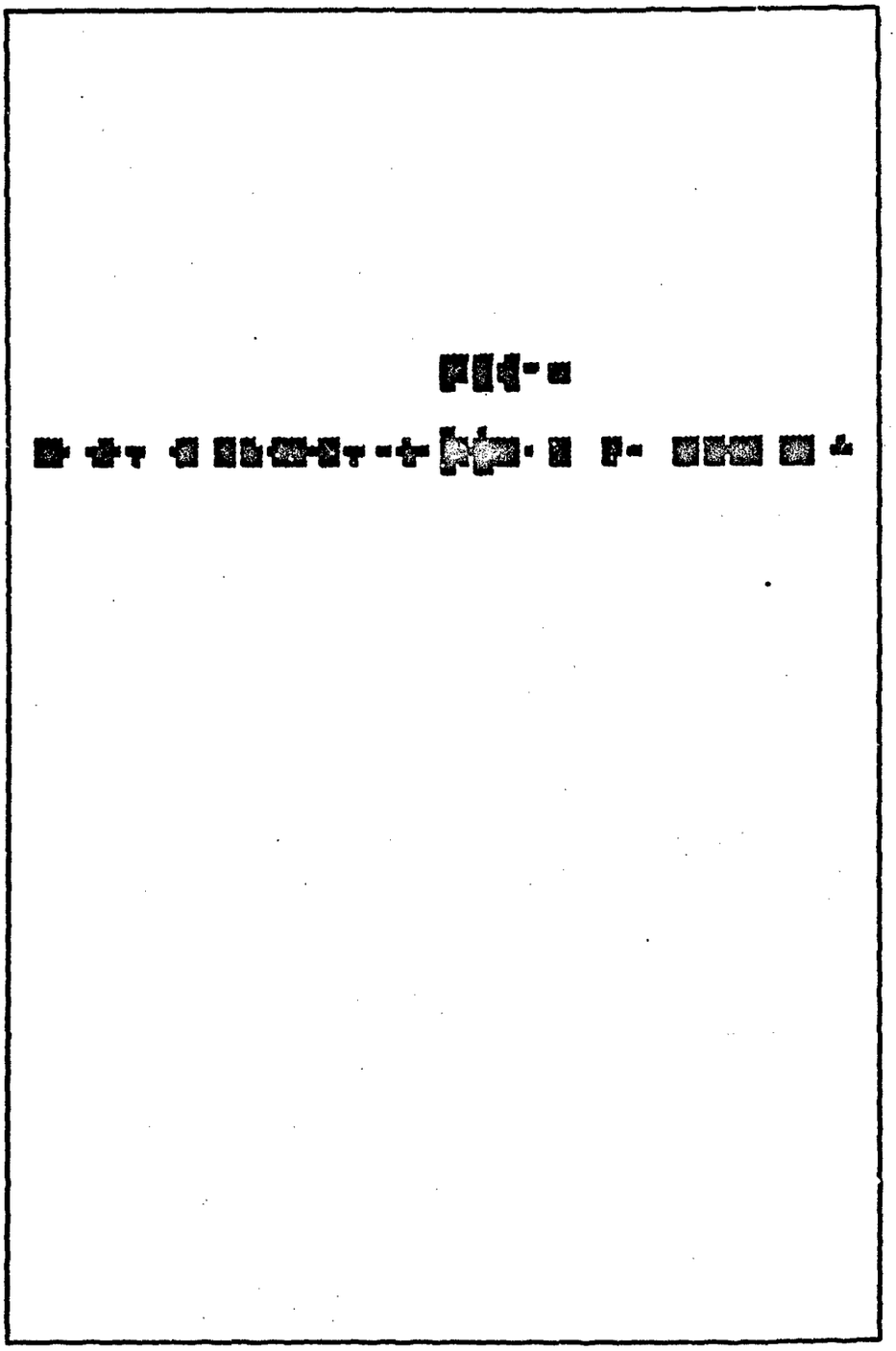


Figure 28. Subroutine FHT1, Data Set 2; Threshold = 1.

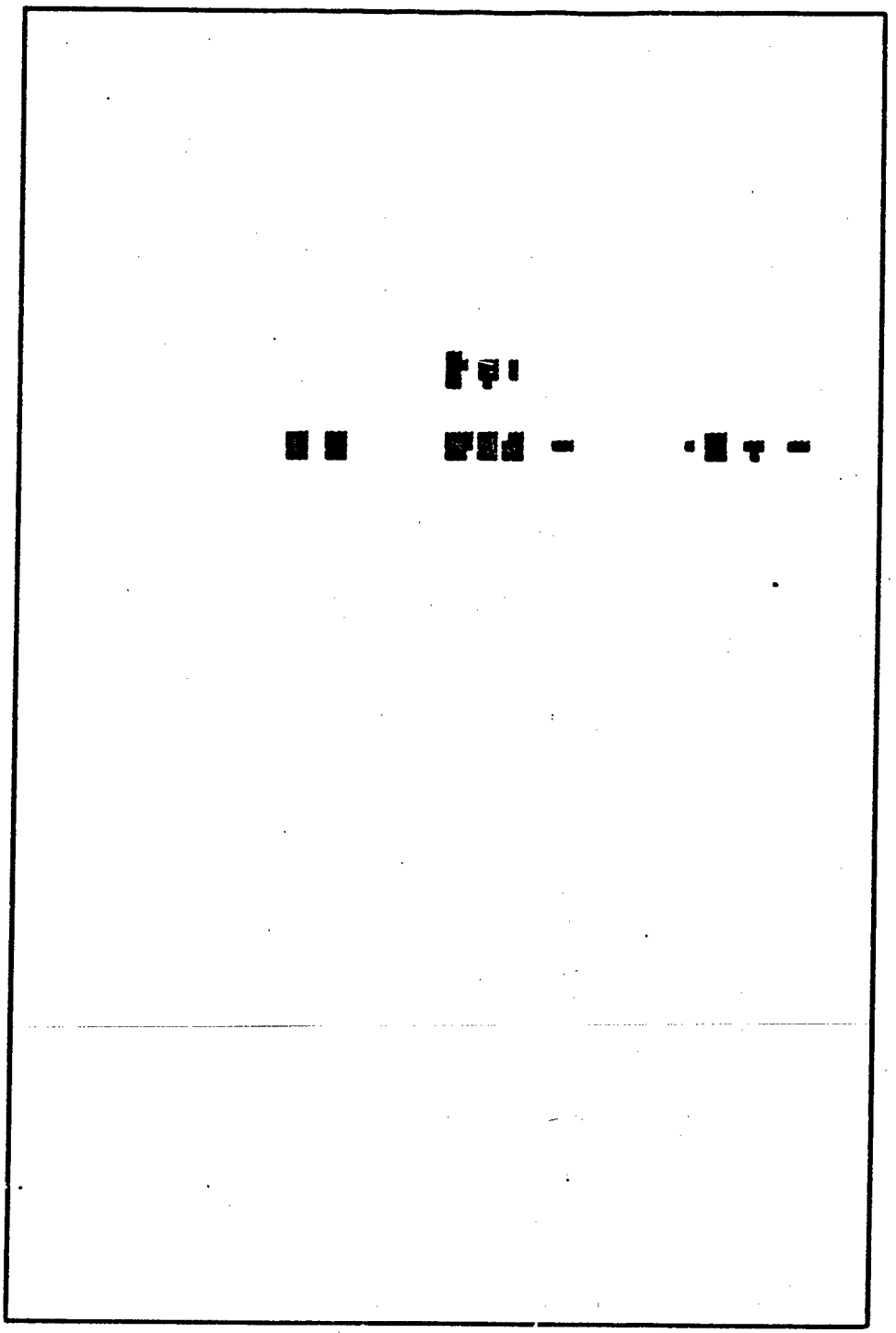


Figure 29. Subroutine 29, Data Set 2; Threshold = 10.

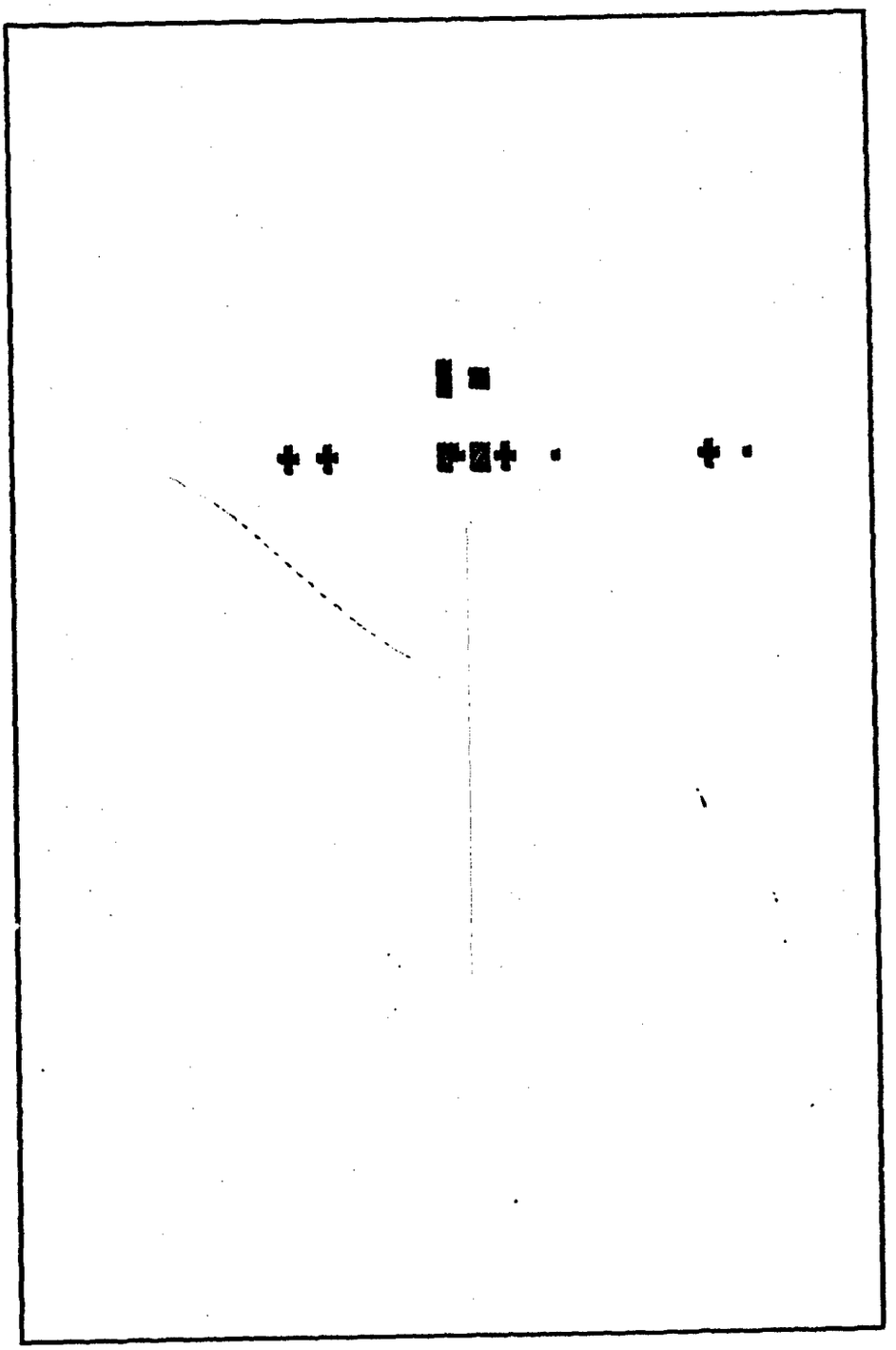


Figure 30. Subroutine FHT1, Data Set 2; Threshold = 20.

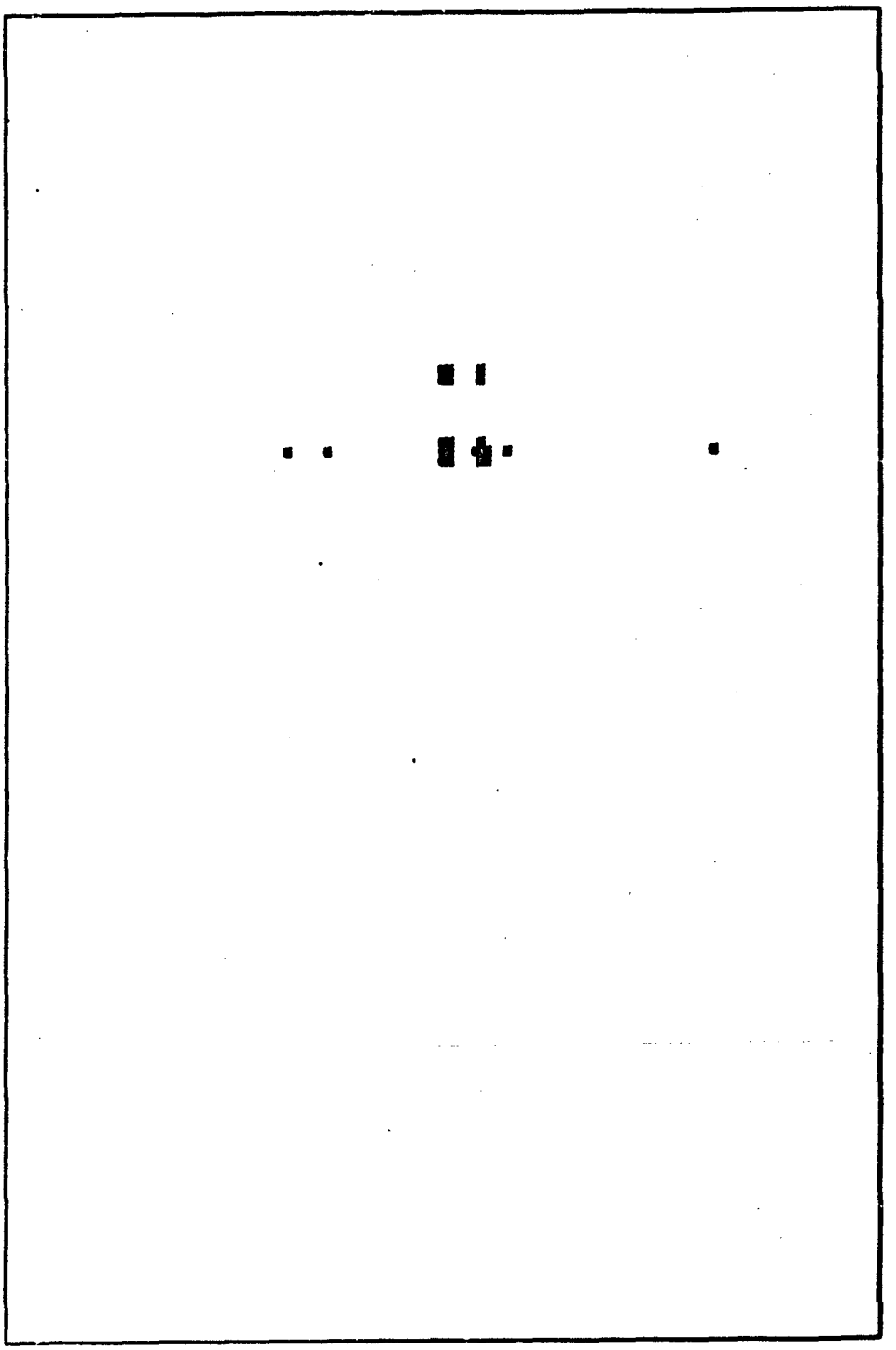


Figure 31. Subroutine FHT1, Data Set 2; Threshold = 50.

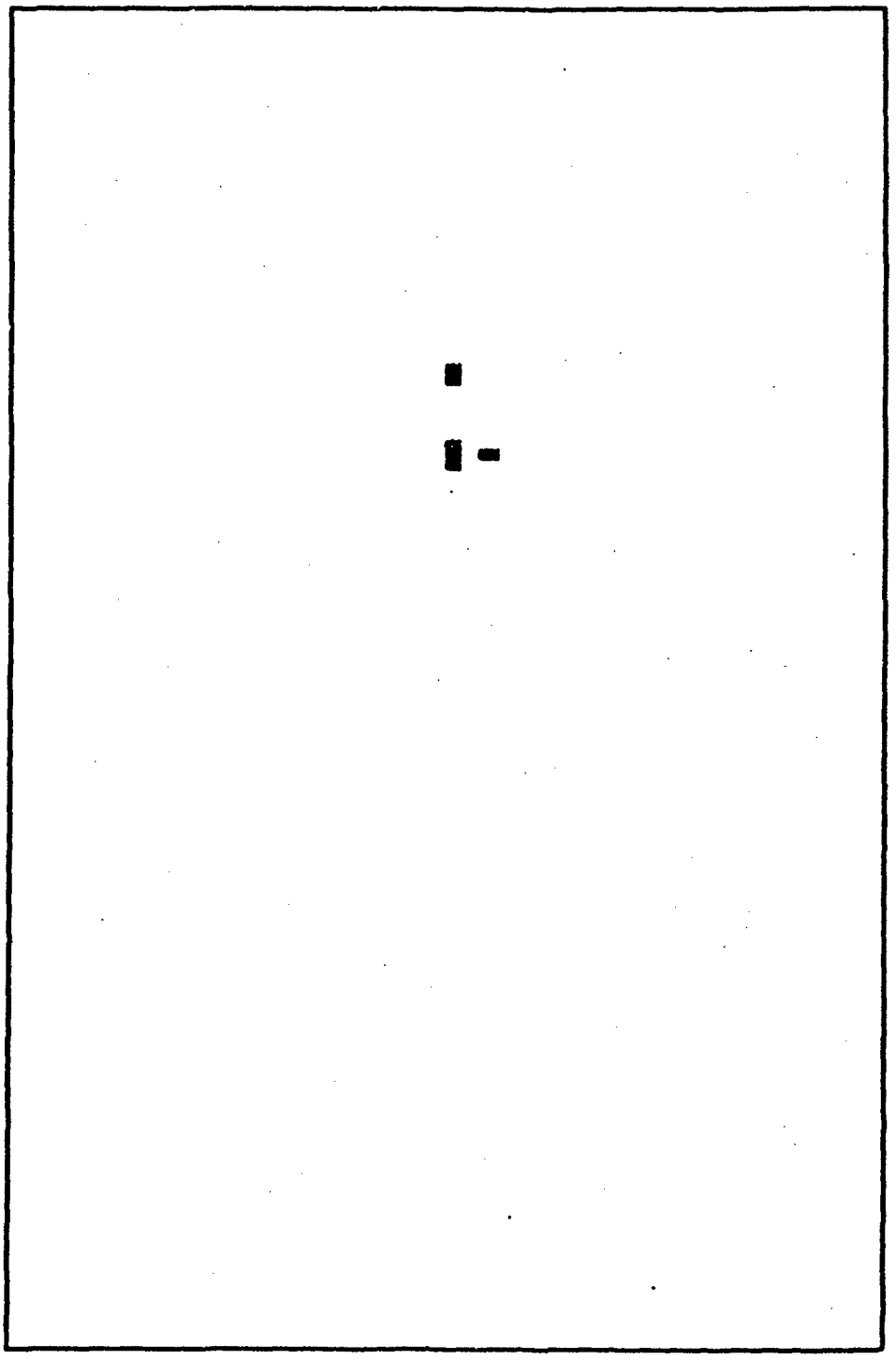


Figure 32. Subroutine FHT1, Data Set 2; Threshold - 100.

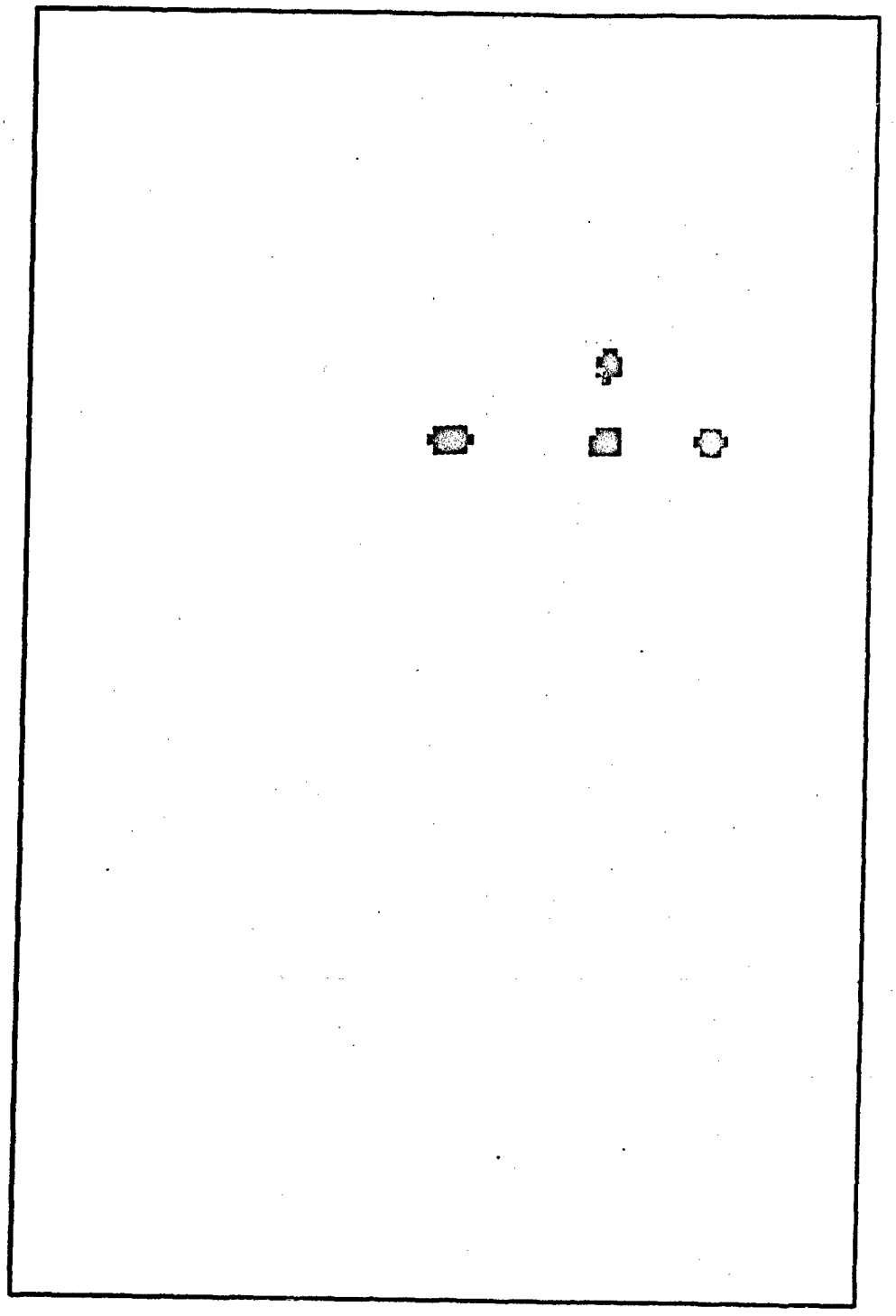


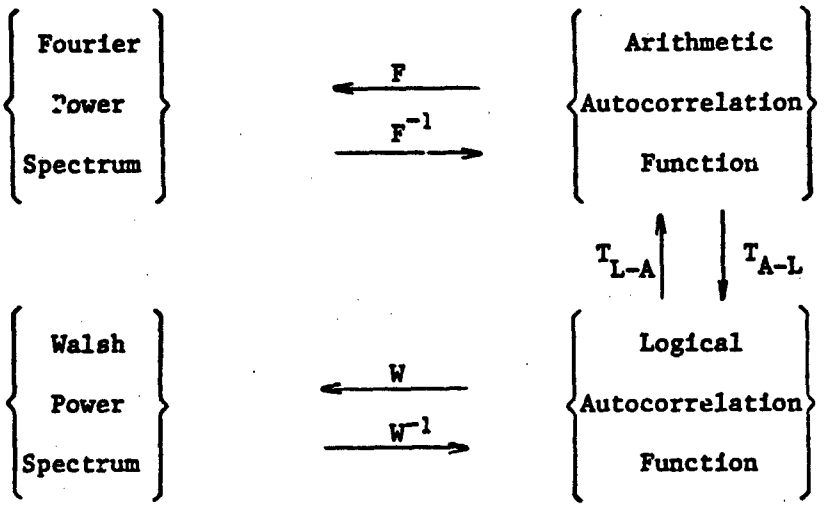
**Figure 33.** Subroutine FFT6; Data Set 2; Threshold - 1000.

## IV. Walsh Domain to Fourier Domain Conversion

A technique developed by Andrews and Caspari (Ref 9) implements a fast Fourier transform, a fast Hadamard transform, and a variety of other orthogonal decompositions suggesting a generalized spectral analysis. Their results imply that a relationship exists between the Walsh domain and the Fourier domain. Robinson (Refs 72, 73) presented a derivation of a recursive relationship between the arithmetic and logical autocorrelation functions of a wide sense stationary process. This work was based on a theorem discovered by Gibbs and proved by Pichler (Ref 38). Siemens and Kitai (Refs 79, 80) and Blachman (Refs 15, 16) describe schemes for converting from Walsh coefficients to Fourier coefficients. Both approaches seem promising because of the computational speed advantage of the FWT/FHT as compared to the FFT and will be analyzed in this section. However, it is shown that Robinson's approach is incorrect and the second approach not feasible for use in radar target identification.

### Walsh Power Spectrum to Fourier Power Spectrum

Robinson defines the Walsh power spectrum of a sequence of random samples as the Walsh transform of the "logical" autocorrelation function of the random sequence, where the logical autocorrelation function is defined in a similar form as the "arithmetic" autocorrelation function (Ref 72:271). Robinson asserts that the Fourier power spectrum, which is defined as the Fourier transform of the arithmetic autocorrelation function, can be obtained from the Walsh power spectrum by a linear transformation (Ref 72:271). The chain of transformations can be summarized as




(79)

However, the procedure to find the transformation matrix, T, was found to be incorrect. Robinson defines the logical autocorrelation function as

$$L^{(m)}(k) = \frac{1}{N} \sum_{J=0}^{N-1} x(j \bullet k)x(j)$$

$$k = 0,1,2,\ldots,N-1 \tag{80}$$

where $x(j)$, $j = 0,1,2,\ldots,N-1$, is a random sequence of length $N = 2^n$ and represents a window or block of N samples of discrete random process. The logical autocorrelation function is then defined as the expected value of the local logical autocorrelation function of Equation 80

$$L(k) = E\{L^{(m)}(k)\} \tag{81}$$

where the expectation operator E denotes the ensemble average of M local logical autocorrelation functions (Ref 73:299)

$$L(k) = \frac{1}{M} \sum_{m=1}^{M} L^{(m)}(k)$$

$$k = 0,1,2,\ldots,N-1 \tag{82}$$

Robinson then defines the arithmetic autocorrelation function as as even function of time difference only

$$E\{x(j+k^*)x(j)\} = R(k^*) \tag{83}$$

where k* is the time shift (Ref 72:272).

If Equation 81 is written in matrix form using the indices of Table I, Robinson asserts that a linear combination of R(k) is obtained for each L(k). As an example, Robinson presents, for N = 4, the following

$$\begin{bmatrix} L(0) \\ L(1) \\ L(2) \\ L(3) \end{bmatrix} = E\left\{ \frac{1}{4} \begin{bmatrix} x(0) & x(1) & x(2) & x(3) & x(0) \\ x(1) & X(0) & x(3) & x(2) & x(1) \\ x(2) & x(3) & x(0) & x(1) & x(2) \\ x(3) & x(2) & x(1) & x(0) & x(3) \end{bmatrix} \right\} \tag{84}$$

Robinson shows that the first row which corresponds to L(0) yields the correlation of N samples for zero time shift, thus L(0) = R(0), which is true. Robinson also states that L(1) = R(1) which is not true. The standard definition of the arithmetic autocorrelation function is given as (Ref 88:13)

$$R(\tau) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)x(n+|\tau|) \tag{85}$$

Using Robinson's relation, Equation 84, L(1) is found to be

$$L(1) = 1/2[x(1)x(0) + x(3)x(2)] \tag{86}$$

Using Equation 85, it can be seen that

$$R(1) = 1/4[x(0)x(1) + x(1)x(2) + x(2)x(3) + x(3)x(0)] \quad (87)$$

Table I

Bitwise Modulo 2 Addition, Given by $j \oplus k$, for Integers Between 0 and 3

| j / k | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 1 | 0 | 2 | 3 |
| 2 | 2 | 3 | 0 | 1 |
| 3 | 3 | 2 | 1 | 0 |

(Ref 72:272)

Table II

Time Shift k*, Given by $k^* = (j \oplus k) - j$, for Integers j and k Between 0 and 3

| j / k | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | -1 | 1 | -1 |
| 2 | 2 | 2 | -2 | -2 |
| 3 | 3 | -1 | -3 | -3 |

(Ref 72:272)

therefore,

$$L(1) \neq R(1) \quad (88)$$

Robinson's error arises from the confusion of the use of k* in

Equation 83 as a dummy variable and the use of k* in Table II as the time shift for computing R(k). This is an incorrect use of the definition of the arithmetic autocorrelation function given in Equation 85, that is, the time shift cannot be varied when computing the arithmetic autocorrelation function for a particular time shift. Robinson uses Table II to find a specific k* for each j in the summation in computing R(k).

It is concluded that this approach is incorrect and that there is no linear relationship between the Walsh power spectrum and the Fourier power spectrum of a signal.

Walsh Series to Fourier Series Coefficeints

Siemens and Kitai, and Blachman have shown that the coefficients of the Walsh series of a function can be used to derive the corresponding Fourier series coefficients. The conversion equation for each Fourier coefficient is in the form of an infinite summation of products of constants and the Walsh coefficients (Ref 79:295). They assume that the signal is frequency-limited so that precise evaluation of the Fourier coefficients in terms of Walsh coefficients is possible and that the highest normalized frequency component (harmonic) N and the highest normalized sequency component M are equal. Thus a finite number of Walsh coefficients can be used for computation of the Fourier coefficients. The conversion computation is further reduced if M is a power of two (Ref 79:295).

Let a function $f(\theta)$ be represented by a sequency-ordered Walsh series

$$f(\theta) = A_o + \sum_{m=0}^{\infty} [A_m \, cal(m,\theta) + B_m \, sal(m,\theta)] \tag{89}$$

The coefficients $A_m$ and $B_m$ of the even and odd Walsh functions, respectively, are defined by

$$A_m = \int_0^1 f(\theta) cal(m,\theta) d\theta \tag{90}$$

$$B_m = \int_0^1 f(\theta) sal(m,\theta) d\theta \tag{91}$$

The same function f(n) has the corresponding Fourier series

$$f(\theta) = \frac{a_o}{2} + \sum_{n=1}^{\infty} [a_n \cos 2\pi n\theta + b_n \sin 2\pi n\theta] \tag{92}$$

where

$$a_n = 2 \int_0^1 f(\theta) \cos 2\pi n\theta \, d\theta \tag{93}$$

$$b_n = 2 \int_0^1 f(\theta) \sin 2\pi n\theta \, d\theta \tag{94}$$

The objective is to use the Walsh coefficients $A_m$ and $B_m$ to derive $a_n$ and $b_n$.

Siemens and Kitai claim that the even terms, $a_n$, of the Fourier series of a signal are functions only of the even terms, $A_m$, of the corresponding Walsh series (Ref 79:295). Similarly, $b_n$ terms depend only on $B_m$ terms. The even, real terms are derived below; similar derivations apply for the odd terms.

The Walsh to Fourier series conversion is derived by equating the terms of each series (Ref 79:296)

$$\sum_{n=1}^{\infty} a_n \cos 2\pi n\theta = \sum_{m=1}^{\infty} A_m \mathrm{cal}(m,\theta) \tag{95}$$

The cal functions are expanded into sets of equivalent Fourier series expressions whose terms have coefficients $a_{n,m}$ where

$$a_{n,m} = 2 \int_0^1 \mathrm{cal}(m,\theta) \cos 2\pi n\theta \, d\theta \tag{96}$$

If $\underline{a}$ represents the nx1 matrix of the set $\{a_n\}$ as $n \to \infty$, and $\underline{A}$ represents the mx1 matrix of the set $\{A_m\}$ as $m \to \infty$, then

$$\underline{a} = \underline{F}\,\underline{A} \tag{98}$$

If only a finite number M of Walsh coefficients are known, then $a_n$ can only be approximated as $\hat{a}_n$, where

$$a_n = \sum_{m=1}^{M} a_{n,m} A_m \tag{99}$$

is the n th Fourier coefficient of cal $(m,\theta)$. The mxn matrix of the set $\{a_{n,m}\}$ is denoted $\underline{F}^T$ (Ref 79:296). In the expansion of the right hand side of Equation 96, terms containing $\cos 2\pi n\theta$ are grouped, yeilding $a_n$ values given by

$$a_n = \sum_{M=1}^{\infty} a_{n,m} A_m \tag{97}$$

However, if the function f(n) is either frequency-limited or sequency-limited, then

$$a_n = \hat{a}_n \tag{100}$$

The coefficients $a_n$ can be considered the Fourier coefficients of the frequency-limited or sequency-limited function (Ref 79:296).

The next step is to find the set $\{a_{n,m}\}$ of $\underline{F}^T$, the Fourier coefficient set of cal(m,θ). Siemens and Kitai developed an alternative expression for the Fourier transform of a Walsh function which differed from earlier expressions in that it incorporated the Gray code representation of the order of the function (Refs 79:81; 15:349). Additionally, the expression is nonrecursive. The definition of the Fourier transform of a Walsh function, wal(m,θ), is defined as

$$\Gamma[\text{wal}(m,\theta)] = \int_0^1 \text{wal}(m,\theta)\exp(j2\pi f\theta)d\theta \qquad (101)$$

The even and odd Walsh functions, cal(m,θ) and sal(m,θ), respectively, have the Fourier transforms

$$F[\text{cal}(s,\theta)] = C(f,s)$$

$$= \int_0^1 \text{cal}(s,\theta)\cos 2\pi f\theta d\theta \qquad (102)$$

and

$$F[\text{sal}(s,\theta)] = jS(f,s)$$

$$= \int_0^1 \text{sal}(s,\theta)\sin 2\pi f\theta d\theta \qquad (103)$$

where f and s are normalized frequency and sequency. Since the Walsh functions are discontinuous, evaluation of Equation 101 - Equation 103 would normally involve a summation of integrals (Ref 80:81).

Siemens and Kitai state that it is convenient to view a contin-

uous Walsh function as the convolution of the sequence of unit impulses that form the discrete Walsh function over the interval $0 \leq \theta \leq 1$ with a rectangular pulse of unit magnitude and the width of $1/2^M$ equal to the spacing of the unit impulse, where M is the number of bits in the binary representation of m. The Fourier transform of the Walsh function is then the product of the transforms of the discrete Walsh function and the rectangular pulse (Ref 80:81). This relation is then

$$F[wal(m,\theta)] = (-1)^{g_0}(-1)^{\alpha}\cdot\left[\prod_{x=0}^{M-1}\cos\left(\frac{\pi f}{2^{x+1}} - g_x\,\frac{\pi}{2}\right)\right]\cdot sinc(f/2^M) \quad (104)$$

where $g_x$ is the $x^{th}$ bit in the Gray code representation of m, and $\alpha$ is the number of Gray code bits of value ONE, and

$$sinc(f/2M) \triangleq \sin(f/2^M)/(f/2^M) \quad (105)$$

If the cal and sal functions are given, the order m is found from

$$wal(m,\theta) = cal(s,\theta),$$
$$m = 2s \quad (106)$$

or

$$wal(m,\theta) = sal(s,\theta),$$
$$m = 2s - 1 \quad (107)$$

Equation 99 is sufficient to compute the Fourier coefficients of f(n) assuming the set $\{A_m\}$ has been found and Equation 104 has been used to determine the $\underline{F}^T$'s necessary to find Fourier coefficients. A total of 2M Walsh coefficients are necessary to find the complex

Fourier coefficients. Therefore, to find $\underline{A}$ requires $2M\log_2 M$ real additions and subtractions. A total of $2M^2$ storage locations are required for the Fourier coefficient sets of cal(m,θ) and sal(m,θ).

The computational load to find the Fourier coefficients of f(n) can be seen to be $2M^2$ multiplies and $(2M^2 + 2m\log_2 M)$ additions. This is compared to $4M\log_2 M$ real multiplies plus 2NM additions to compute the fast Fourier transform.

It can be seen that the Semes conversion approach requires greater computational effort than the direct fast Fourier transform method and a very large storage requirement for the transformation matrices ($\underline{F}^T$). It is therefore concluded that this approach is not feasible for implementation in this problem.

## V. Comparison of FFT Algorithms

It is seen in Section IV that many orthogonal transforms are not feasible for use in tactical target identification signal processing. This section provides a brief description of the fast Fourier transform (FFT) and then selects the optimal implementation of it for this application. Performance criteria, based on execution time, memory requirements, and error size, are developed and serve as a basis of comparison among the implementations considered. Two general types of implementations, floating point and fixed-point, are presented, tested, and evaluated using the criteria.

Several different "floating point" FFT subroutines are tested using the simulation program TGTID as the "driver" program to determine their relative utility. Two of these FFT subroutines are modified to execute more efficiently then one FFT is selected based on the criteria as the most feasible in the simulation program. The FORTRAN IV code of the FFT subroutines is listed in Appendix C.

A "fixed-point" FFT implementation is derived and demonstrated to be feasible in this application. It is considered to be the optimal approach for this problem. Several variations of the fixed point FFT are implemented with the FORTRAN IV listings given in Appendix E. One fixed-point FFT subroutine is selected as the best choice for use in the simulation program.

### Background

The FFT algorithm is a highly efficient procedure for computing the Discrete Fourier transform (DFT)

$$X(k) = \sum_{n=0}^{N-1} x(n)\exp(-j2\pi nk/N) \qquad (108)$$

for k = 0,1,2,...,N-1, where x(n), n = 0,1,2,...,N-1 is an equally spaced complex data series (Ref 62:100). It takes advantage of the fact that the calculation of the Fourier coefficients can be carried out iteratively, which results in a considerable savings of computational time (Ref 23:45). The DFT is a reversible mapping or unitary operation for time series and has the same mathematical properties as those of the Fourier integral transform given in Equation 8 (Ref 23:45). In particular, it defines a frequency spectrum of a time series.

If digital analysis techniques are used for analyzing a continuous waveform then it is necessary that the data be sampled at equal time intervals in order to produce a time series of discrete samples which can be used in a digital computer. So that the time series completely represents the continuous waveform, it is necessary that the waveform be band-limited and sampled at twice the highest frequency in the waveform (Ref 24:45). This rate is known as the Nyquist rate and the equispaced samples are known as Nyquist samples.

The speed of the FFT algorithm depends on the factorability of N

$$N = \prod_{i=1}^{M} n_i \qquad (109)$$

and then decomposing the transform into M stages with $N/n_i$ transformations of size $n_i$ within each stage (Ref 18:93). The FFT algorithm is most efficient when $N = 2^M$ or $n = 4^M/2$ since a computational advantage is gained by decomposing the computation of the DFT into successively smaller DFT's (Refs 54:2; 62:285-287). The improvement in computational efficiency of the FFT is proportional to $N \log_2 N$ as opposed to the

computational load of $N^2$ for the straight DFT approach (Ref 60:287).

There are two basic FFT algorithms. The first, called "Decimation-in-time", derives its name from the fact that in the process of arranging the computation into smaller transformations, the sequence x(n) is decomposed into successively smaller sequences. In the second general class of algorithm, the sequence of Fourier transform coefficients of X(k) is decomposed into smaller subsequences, hence the name "decimation-in-frequency" (Ref 62:286-287). When either one of the two algorithms is used, the ordering of the sequence (either input or output) must be taken into account. If the decimination-in-frequency algorithm is used the output coefficients will be in bit-reversed order; and if the decimation-in-time algorithm is used, the input samples must be put into bit-reversed order. Therefore, in addition to performing the FFT, a reordering procedure must also be implemented which accounts for approximately 15-20% of the execution time of the FFT process (Ref 64:17). Figure 34 and Figure 35 show the signal flow graphes of the decimation-in-time and decimation-in-frequency algorithms, respectively.

The literature is rich in material on the FFT: its derivation, its properties, its uses, and its pitfalls. The reader is referred particularly to Cooley and Tukey (Ref 25). Bergland (Ref 12), Cochran et al (Ref 24), Gentleman and Sande (Ref 36), Singleton (Refs 81, 82), and Oppenheim and Schafer (Ref 62). The Bibliography of this thesis contains several other related and excellent references.

Criteria

The criteria generally applied to the evaluation of an FFT algorithm include program execution time, accuracy, storage requirement , and

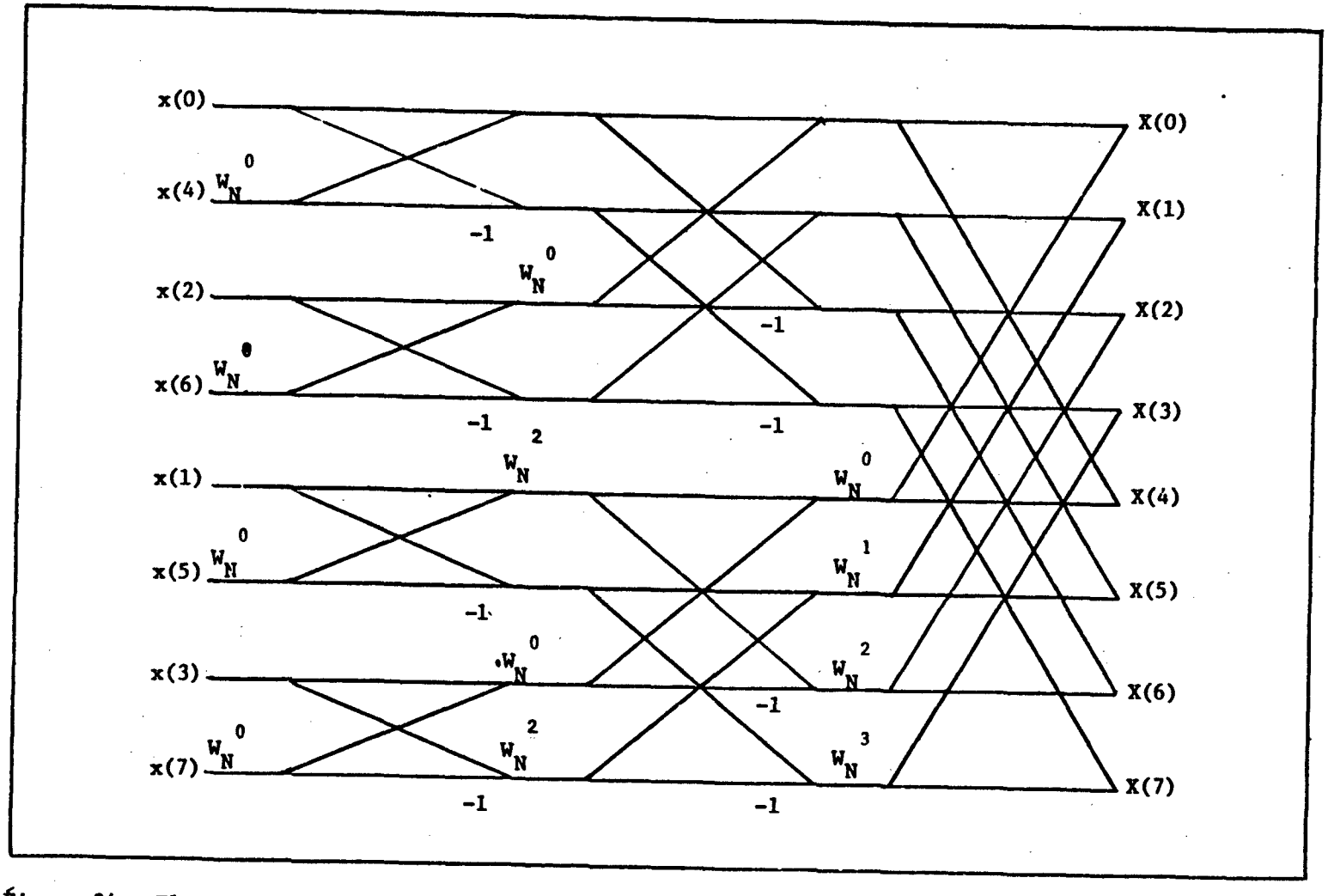



Figure 34. Flow Graph of Complete Decimation-in-time Decomposition of an Eight-point DFT Computation.

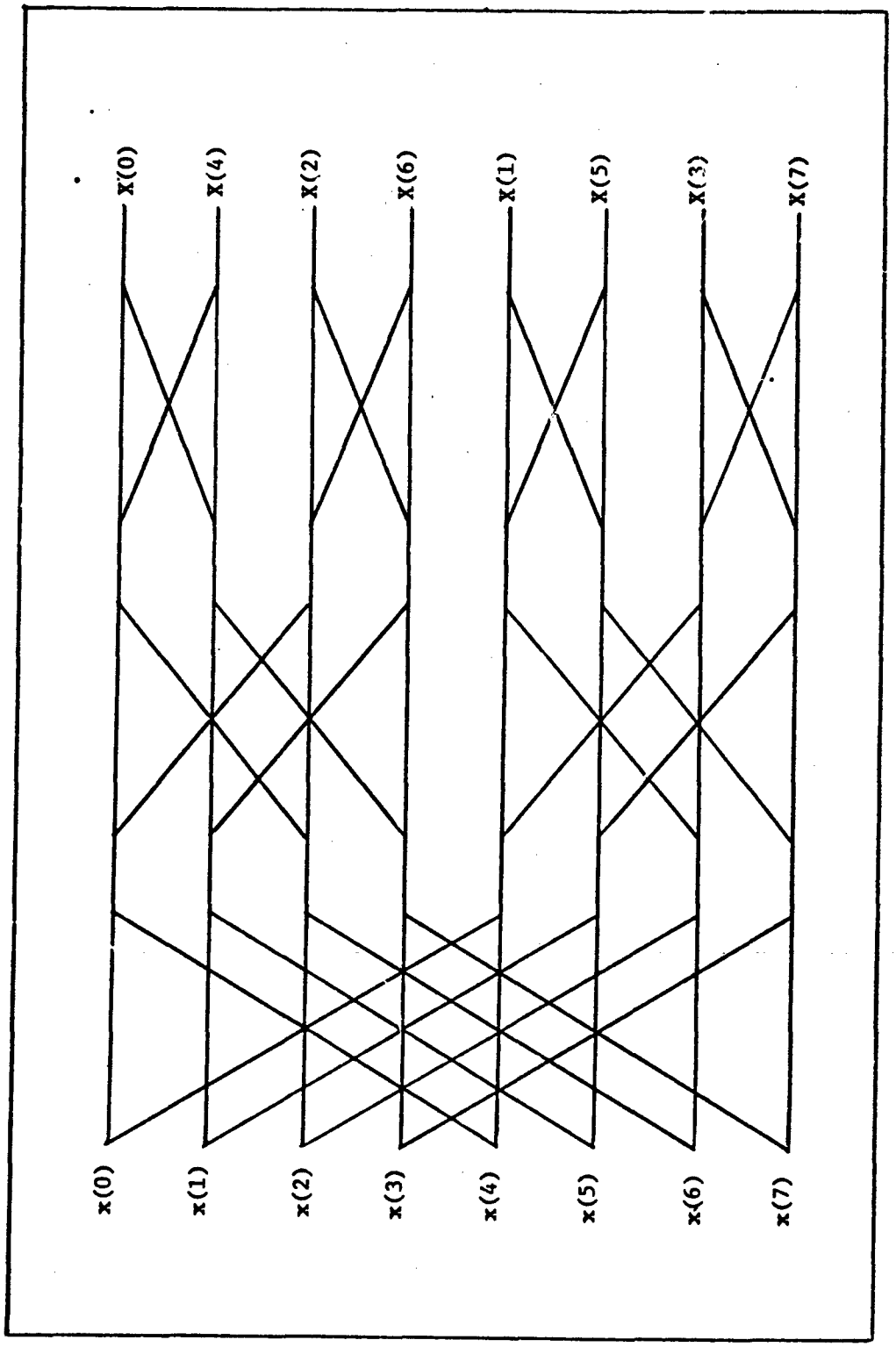



Figure 35. Flow Graph of Complete Decimation-in-frequency Decomposition of an Eight-point DFT Computation.

adaptibility to the digital computer to be used (Ref 34:1) Bergland points out that the number of variations of the FFT algorithm appears to be directly proportional to the number of people using it and that most of these implementations are based on either the Cooley-Tukey or the Sdnde-Tukey methods, but are formulated to exploit properties of the time series being analyzed or the properties of the computer being used (Ref 12:50). Ferrie states that best accuracy is achieved only at the expense of increased execution time and storage (Ref 34:10). In essence, then, no single FFT algorithm represents a "best" choice: it depends upon the application (Ref 71:25).

For this application, short execution time is considered the most important criterion, and the next most important is small storage requirements. Since, only major spectral lines are of interest, then only coarse calculations are important, therefore, accuracy is not considered an important criterion. Additionally, the type of digital computer for implementation is the DEC PDP-11/40, and its high speed integer arithmetic capability motivated the development of a fixed-point FFT subroutine.

Floating Point FFT Algorithm Comparison

The eight FFT subroutines evaluated are briefly described in this section. All the subprograms are written in FORTRAN IV. In their original form most were named "FFT"; for this thesis they are designated FFT1 through FFT6. Two of the subroutines, FFT1 and FFT4, are modified to improve their efficiency. All the subroutines, except FFT2, perform the transform in-place. And all handle complex exponentials in terms of sine and cosine functions according to Euler's rule

$$\exp(-j\theta) = \cos\theta - j\sin\theta \qquad (110)$$

FFT1A and B (Ref 62:331). FFT1A and B are radix-2, decimation-in-time transform subprograms. They can handle N equal a power of two points. The data are treated as complex, and the trigonometric functions are computed as they are used. They are simply implemented and straightforward. Computations are performed in-place and calculations use complex arithmetic. Memory storage requirements depend only on N. Initial bit reverse ordering is done within the subroutine.

FFT1B is made more efficient by performing the first state separately from the rest of the stages by exploiting the fact that the value of $W_N^{nk} = \exp(-j2\pi nk/N)$ is zero for the first stage butterflies, the basic unit of computation. Therefore, the cosine and sine values are one and zero, respectively, and the arithmetic operations are simple additions and subtractions. Figure 34 shows this property.

FFT2 (Ref 71:21). FFT2 is a radix-2 decimation-in-time algorithm. It uses two arrays to perform the transform, so that no separate reordering routine is required, but the storage requirement is doubled. Complex arithmetic is used. It is known for its accuracy but its execution is slow (Ref 71:25).

FFT3 (Ref 71:21). FFT3 is a simple, straight forward radix-2 implementation of the decimation-in-frequency algorithm. It uses real arithmetic to compute the butterfly. It requires Subroutine RBITS to reorder the bit-reversed Fourier coefficients. Computations are performed in-place. Internal storage requirements for this subroutine are greater than those of the previous subroutines.

FFT4A and B (Ref 62:332). FFT4A performs the transform approximately the same as FFT3 except that the indexing scheme is simpler. Complex arithmetic is used and calculations are performed in-place.

Reordering is accomplished within the subroutine. FFT4B is modified to make it more efficient by performing the last stage separately, similar in concept to FFT1B.

FFT5 (Ref 52:27). FFT5 is based on the original Cooley-Tukey FORTRAN subroutine. It is a radix-2 transform subprogram and performs computations in-place. It is neither a fast or efficient implementation of the FFT algorithm (Ref 54:28).

FFT6 (Ref 45:A-3). FFT6 is a mixed-radix decimation-in-frequency algorithm. It can handle $2^{13}$ points which can be increased with minor modification to the subroutine. It requires significantly more internal storage than any of the other FFT subroutines, but it is also known for its speed (Ref 82).

Evaluation of the Floating Point FFT Subroutines

The eight FFT subroutines were tested in the simulation program TGTID with appropriate modifications made to Subroutine CIMAGE (Appendix B).

Execution times, memory requirements, and the number of FORTRAN IV statements for each FFT subroutine are summarized in Table III. The number of FORTRAN IV statements is the total number of statements required to perform the transform and reordering routines, but does not include comment cards. Storage requirements were obtained from the cross-reference maps produced by the CYBER SCOPE Operating System. Execution time was determined by using the CEC library subroutine SECOND (CP) which returns time in seconds to three decimal places. In Subroutine CIMAGE, the FFT subroutine is called 256 times (NTM = 2 and MBSZ = 128; it is called 128 times for NTM = 1 and 128 times for

NTM = 2). The FFT size is 128 points (NBSZ = 128 = $2^{NBORD}$, where NBORD = 7). Thus each FFT tested performed 256 128-point transforms to produce the image. The execution time was measured by taking a time "hack" (STIME) just before the FFT subroutine was called and just after it returned (ETIME) to the calling subprogram. The difference between the two time hacks was computed (RTIME = ETIME - STIME) and added to the total accumulated execution time (TTIME). The final value of TTIME was the total execution time to compute the 256 128-point FFT subroutines needed to generate one image. The execution time entry in Table III is an average of the total execution time for each FFT subroutine to generate at least three separate images.

Results of Floating Point FFT Comparison

From Table III, it can be seen that FFT1B and FFT 6 are the fastest subroutines. FFT1A required the least amount of memory. However, it is concluded that FFT1B is the best floating point FFT subroutine based on its speed and storage requirements. Figures 36-39 show the output images generated by FFT1B for increasing threshold settings. Images generated using FFT6 are shown in Figures 9 through 14.

Fixed-Point FFT

The implementation and use of a fixed-point FFT is motivated by several factors. The DEC PDP-11/40 minicomputer to be used in the actual applications work of radar target imaging processes an extremely fast integer arithmetic processor and has no floating point processor. The 16-bit word is considered sufficient to provide an adequate resolution since only gross, relative spectral line

Table III

Summary of Floating Point FFT Performance Statistics

| FFT | Number of FORTRAN Statements | Number of Instruction Words (Total) | Aux Arrays (Words) | Execution Time (Sec) |
|---|---|---|---|---|
| FFT1A | 36 | 127 | 2N | 1.412 |
| FFT1B | 40 | 143 | 2N | 1.367 |
| FFT2 | 31 | 142 | 4N | 2.59 |
| FFT3 | 21[1] | 197 | 2N +10 | 2.007 |
| FFT4A | 35 | 130 | 2N | 1.550 |
| FFT4B | 40 | 148 | 2N | 1.51C |
| FFT5 | 54 | 176 | 2N +15 | 2.38 |
| FFT6 | 112 | 350 | 2N +26 | 1.367 |

[1]Subroutine RBITS = 32 statements, 120 words

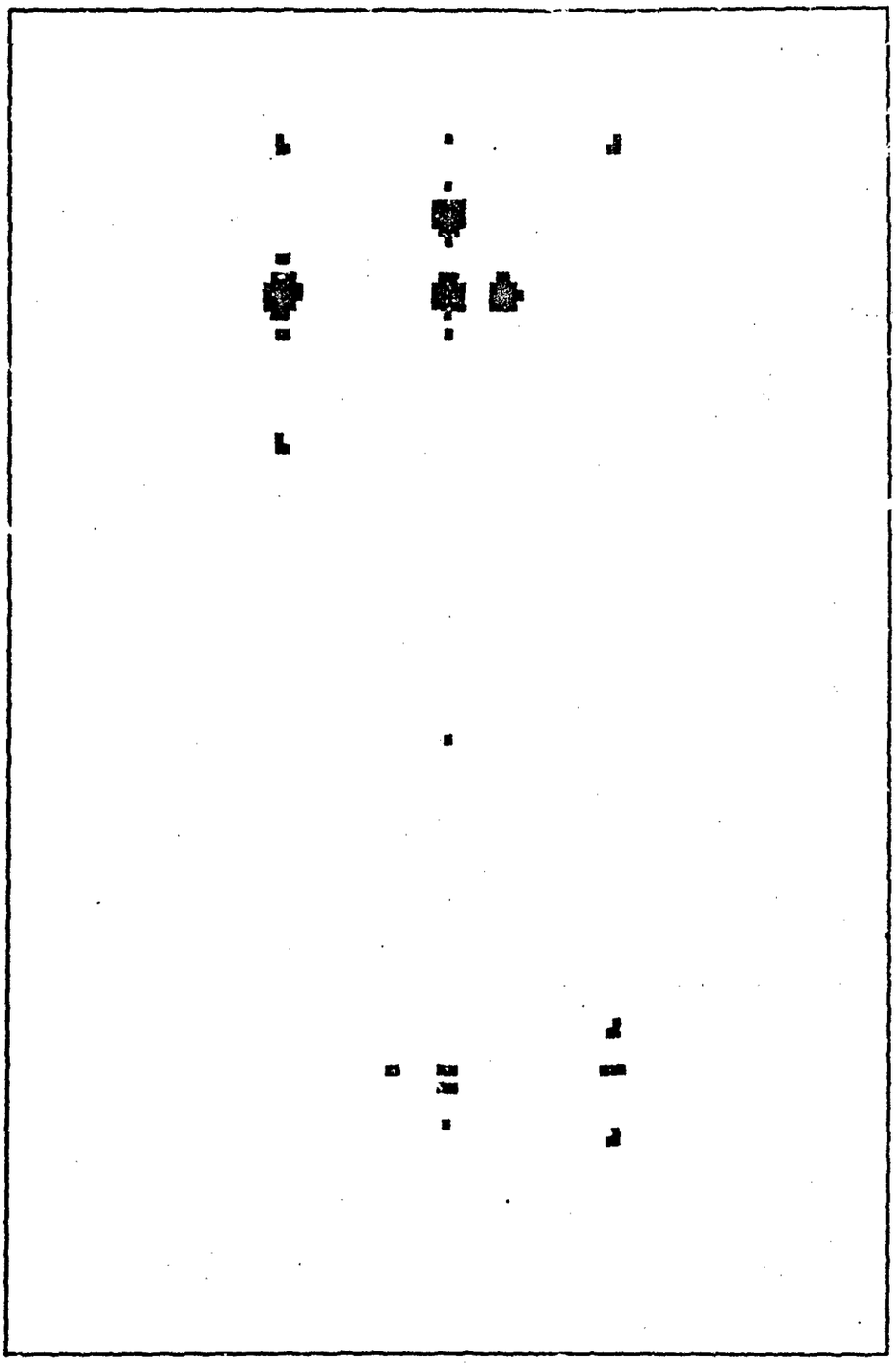


Figure 36. Subroutine FFT1B; Threshold = 500.

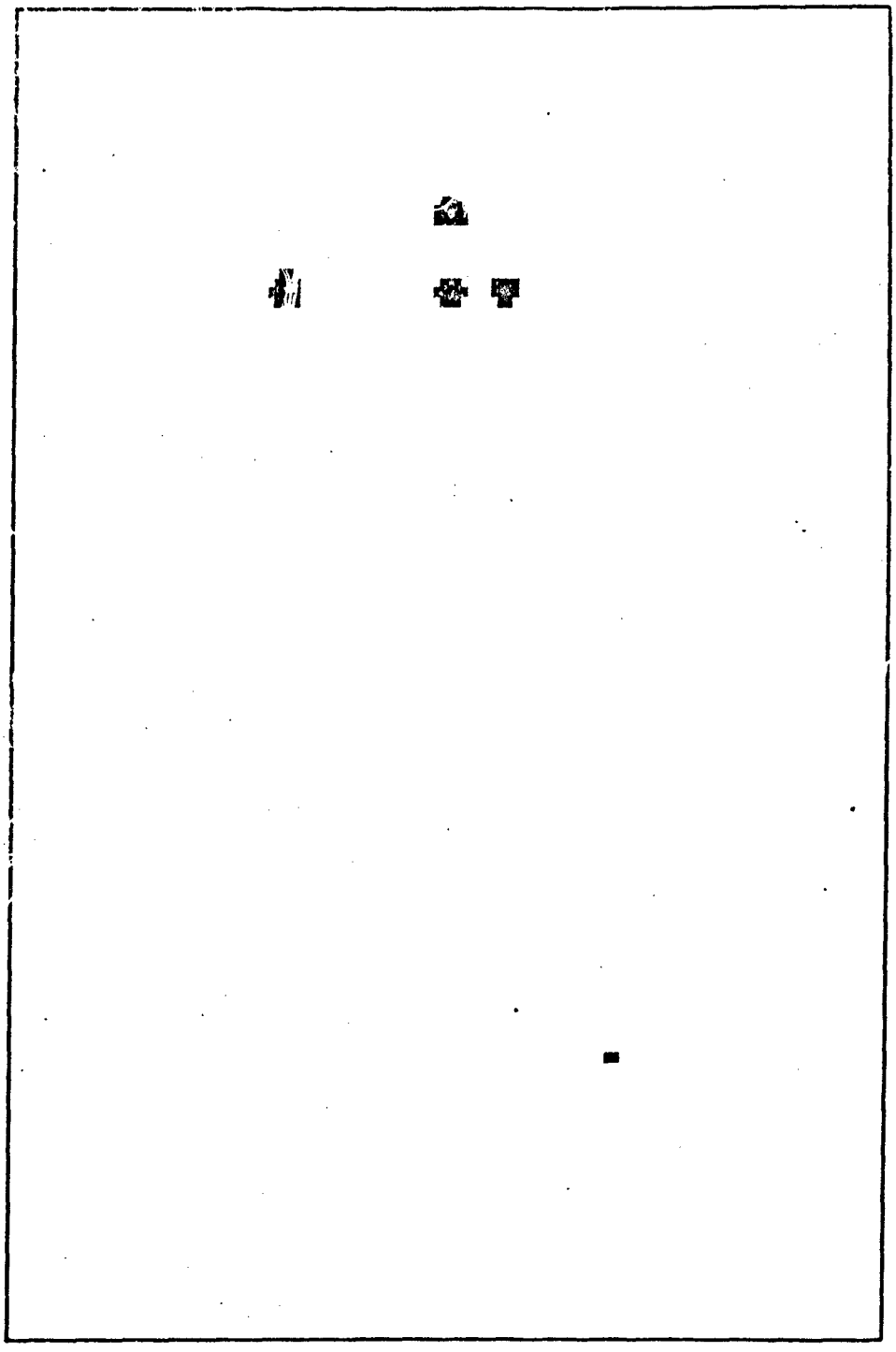


Figure 37. Sburoutine FFT1B; Threshold = 1000.

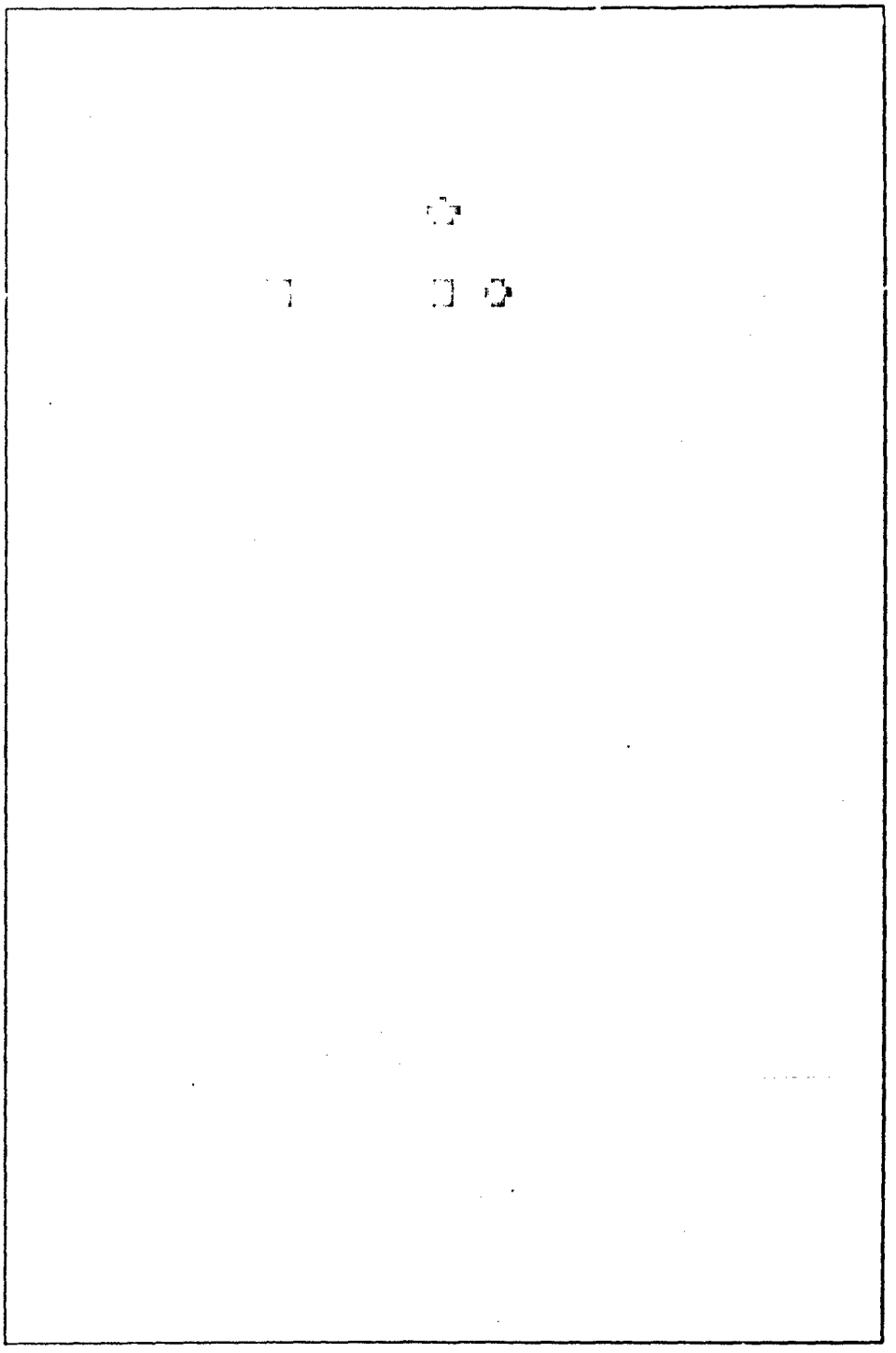


Figure 38. Subroutine FFT1B; Threshold = 1500.

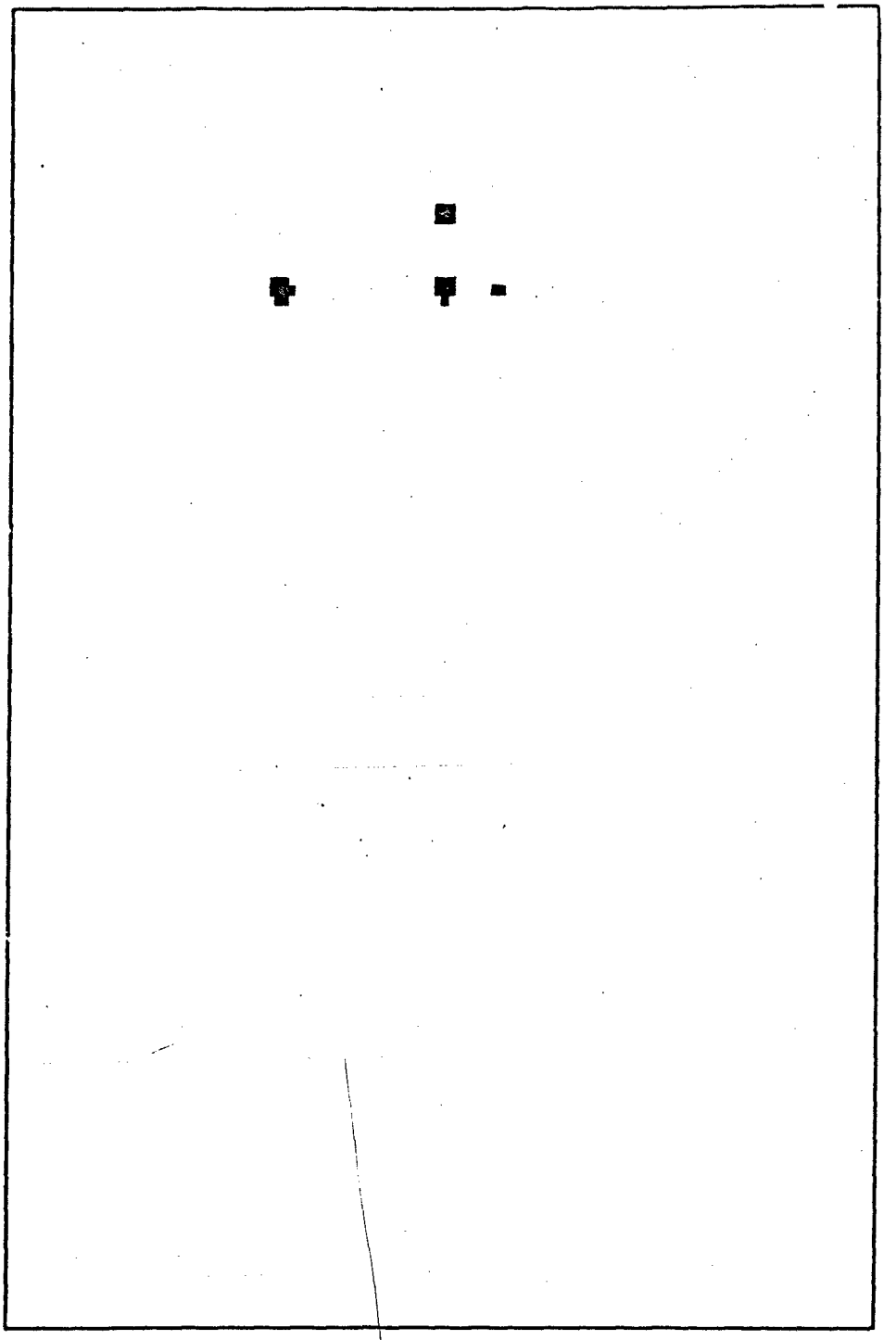


Figure 39. Subroutine FFT1B; Threshold = 2000.

magnitudes are really required for this application. Fixed point division by two is easily performed by shifting the binary point to the left.

The fixed-point FFT is based on the 16-bit word of the DEC PDP-11/40. The data word is divided into two parts: the lower 15 bits for the numbers are scaled so that the binary point lies at the extreme left.

The input data must be put into the proper format. First, each point is multiplied by $2^{15} - 1 = 32767$ to put it into a scaled integer format. Second, each point is divided by the total number of points, N, in the transform (in this case N = 128) to negate the gain that is generated as the computation of the FFT progresses from stage to stage to prevent the possibility of an overflow when two points are added together in the butterfly calculation. The real and imaginary parts of each data set are scaled and are sorted in two separate, real and imaginary, arrays, respectively.

The trigonometric functions are computed and then multiplied by 32767 to integer scale each sine and cosine value. When the trigonometric terms are multiplied by the difference of two points in the computation of a butterfly, the product results in a 31-bit number plus sign, which is greater than one. To rescale this number, it is divided by $2^{15} = 32768$ or more simply, the binary point is shifted 15 places to the left and the 16 least significant bits are truncated. This, likewise, prevents an overflow condition from occurring. The basic butterfly computational algorithm (decimation-in-frequency) is shown in Figure 40, and the corresponding equations are

$$T1 = X_m(p) - X_m(q) \tag{111}$$

$$T2 = Y_m(p) - Y_m(q) \tag{112}$$

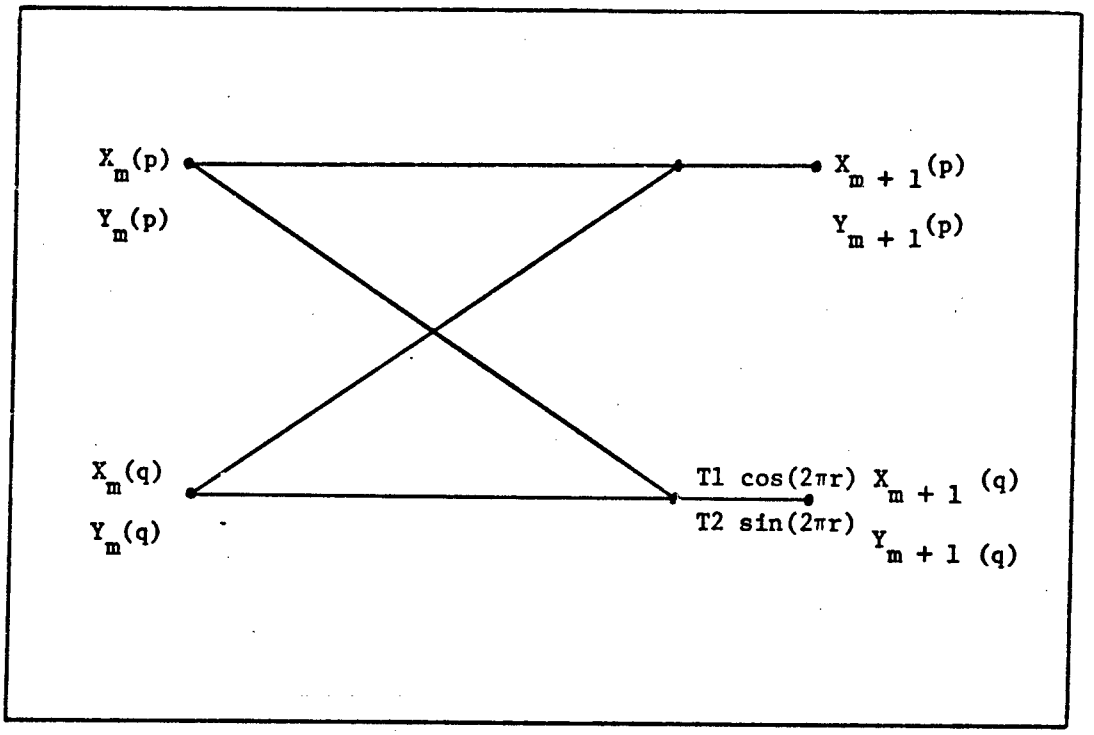



Figure 40. Flow Graph of Two-Point Integer FFT Butterfly Using Real Arithmetic

$$X_{m+1}(p) = X_m(p) + X_m(q) \tag{113}$$

$$Y_{m+1}(p) = Y_m(p) + Y_m(q) \tag{114}$$

$$X_{m+1}(q) = (\cos(2\ r)*32767*T1 + \sin(2\ r)*32767)/32768 \tag{115}$$

$$Y_{m+1}(q) = (\cos(2\ r)*32767*T2 + \sin(2\ r)*32757)/32768 \tag{116}$$

where X and Y represent the real and imaginary parts, respectively.

The accuracy of the fixed-point power of two FFT algorithm is addressed by Welch (Ref 91). His error analysis led to approximate upper and lower bounds on the root-mean-square error. Based on Welch's findings, it was determined that the theoretical upper bound for this application (B = 16 and N = 128) is (Ref 91:156)

$$\frac{\text{RMS(error)}}{\text{RMS (result)}} = 1.5\text{x}10^{-3} \qquad (117)$$

This bound is considered to be more than adequate for radar target imaging. Welch further states that if the transform is taken to estimate the power spectrum of a signal, averaging over frequency in a single periodogram, or averaging over time in a sequence of periodograms, or using simple wieghting will decrease the error (Ref 91:157). It is concluded that the use of the fixed-point FFT algorithm is feasible for implementation in radar target imaging signal processing.

Five "simulated" fixed-point FFT subroutines were written in FORTRAN IV for this thesis. They are based on the floating point FFT algorithms previously discussed, excpet Subroutine FFTI4. They are written in sumulated form for use on the CDC 6600 CYBER 74 System instead of the minicomputer for which they are intended. Shifting of the binary point is accomplished with the use of the CDC intrinsic library function SHIFT (A,-N).

FFTIi. FFTI1 is based on FFT. It is converted from complex, floating point arithmetic to real, fixed-point arithmetic. Indexing and reordering are not changed.

FFTI2. FFTI2 is based on FFT1. It is not as efficient to implement as the decimation-in-frequency algorithm.

FFTI3. FFTI3 is based on FFT4. Reordering is performed within the subroutine.

FFTI4. (Ref 34:15-18) FFTI4 is based in part on an algorithm presented by Fisher (Ref 35). Trigonometric functions are precomputed, scaled, and stored in a table. The subroutine is called once by the calling program and passes the table to the main FFTI4 subroutine.

Subroutine COSINE (Appendix IO is used to generate the trigonometric table. It is called by Subroutine CIMAGE (Appendix E, Version 3), and used in Subroutine FFT14. An address loopup routine is used to obtain the proper sine/cosine value from the table. Subroutine UNSCR1 (Appendix J) is called by FFTI4 to reorder the Fourier coefficients. A complete derivation and floating point implementation is reported by Fisher (Ref 34).

FFTI5. FFTI5 incorporates the best features of FFT4B and FFTI4. A cosine table is generated from zero to two pi. FFTI5 uses a lookup address routine to obtain the required trigonometric values for the butterfly computation. The last stage of the transform is performed separately to eliminate the lookup and unnecessary multiplies. Subroutine UNSCR1 is called to reorder the fixed-point Fourier coefficients.

Evaluation of the Fixed-point FFT Subroutines

The five "simulated" fixed-point FFT subroutines were tested in the simulation program TGTID with appropriate modifications made to Sburoutine CIMAGE (Appendix B, Version 3). They were evaluated in the same manner as the floating point FFT subroutines. Table IV summarizes the number of FORTRAN IV statements, the amount of memory required, and the average execution time (over at least three runs of program TGTID) for each fixed point FFT subroutines.

The execution time was measured in exactly the same manner as the floating point FFT subroutines. The first time "hack" (STIME) was taken just after the data were converted to fixed-point format and just after the data were converted to fixed-point format and just before the fixed-point FFT subroutine was called. The second time hack (ETIME)

was taken just after the return from the subroutine. The difference was computed (RTIME) and added to the accumulated total execution time TTIME) of the fixed-point FFT subroutine. This time included obth the transform and reordering routines. It does not include the time to compute the trigonometric functions of Subroutines FFTI4 and FFTI5, which are only done once.

Results of the Fixed-point FFT Subroutine Comparison

It can be seen in Table IV, that FFTI5 is the fastest fixed-point FFT subroutine. FFTI3 requires the least amount of memory, while FFTI1 reuqires the most amount of storage. Based on the speed and moderate storage requirements, FFTI5 is considered to be the best fixed-point FFT subroutine to use in tactical radar target imaging using the DEC PDP-11/40 minicomputer. Figures 41 through 44 show the image created using FFTI5 for increasing threshold. It should be noted that the threshold is set too low in Figure 41 and too high in Figure 44.

Table IV

Summary of Fixed Point FFT Performance Statistics

| FFT | Number of FORTRAN Statements | Number of Instruction Words (Total) | Aux Array (Words) | Execution Time (Sec) |
|---|---|---|---|---|
| FFTI1 | 47 | 180 | 2N | 2.27 |
| FFTI2 | 36 | 161 | 2N | 2.5 |
| FFTI3 | 36 | 112 | 2N | 2.185 |
| FFTI4 | 42[1] | 154[3] | 2N + N/4 + 12 | 1.95 |
| FFTI5 | 31[2] | 126[3] | 2N + N/2 + 12 | 1.20 |

[1]Subroutine COSINE has 10 statements, 29 words

[2]Subroutine COSINE has 8 statements, 19 words

[3]Subroutine UNSCR1 has 29 statements, 125 memory words

NOTE: Subroutine UNSCR1 is faster than Subroutine UNSCR (Appendix F).

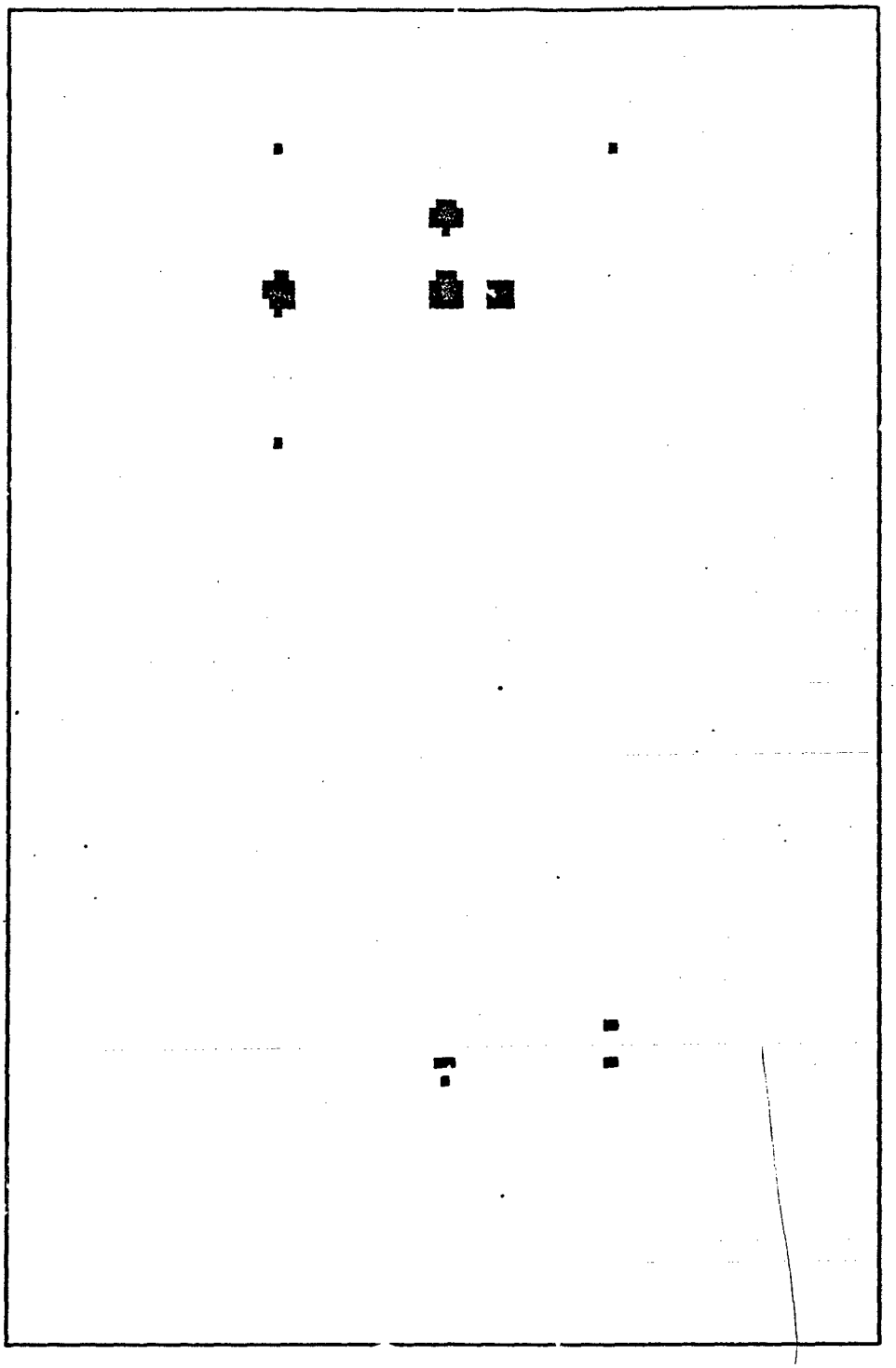


Figure 41. Subroutine FFTI5; Threshold = 30.

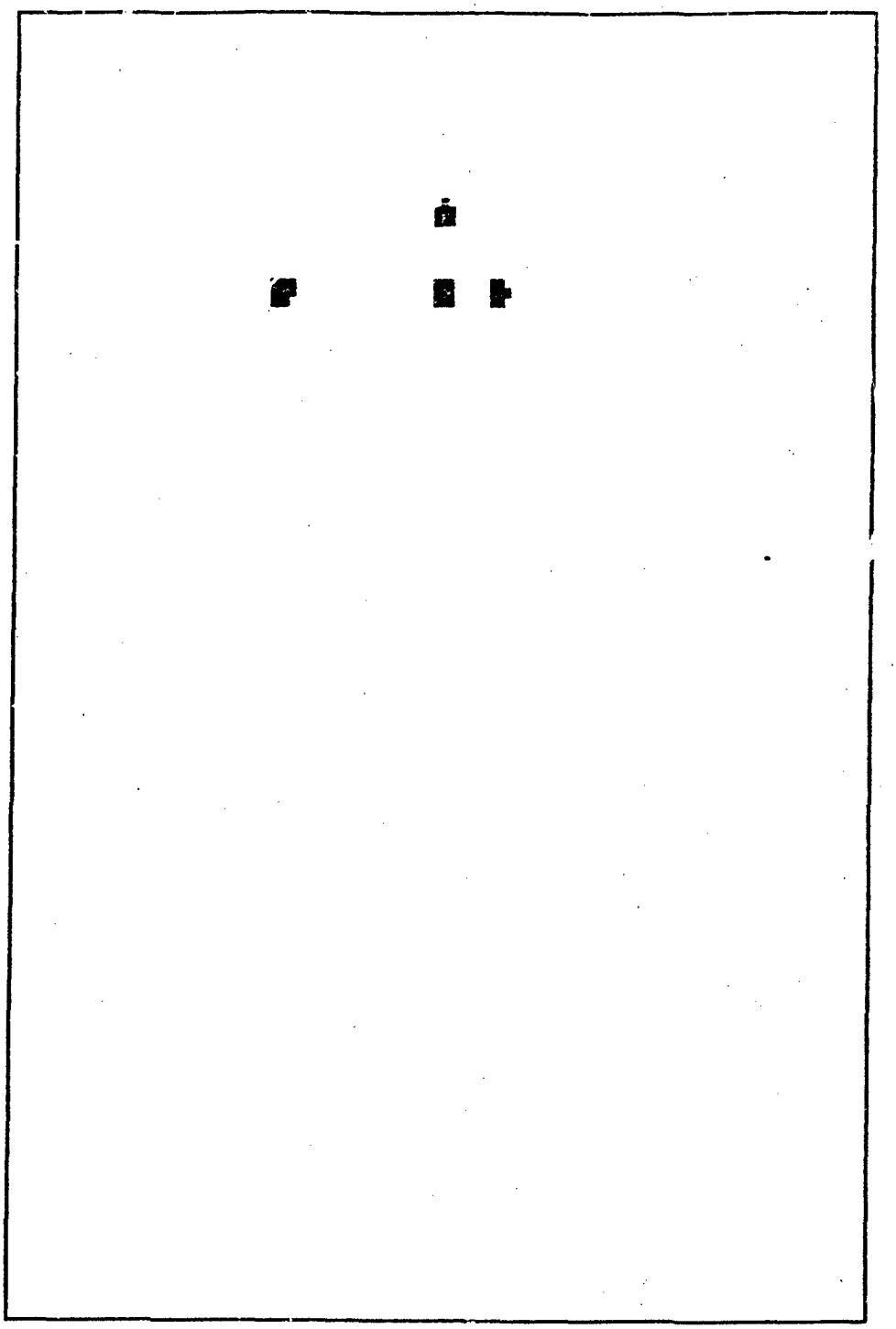


Figure 42. Subroutine FFTI5; Threshold = 90.

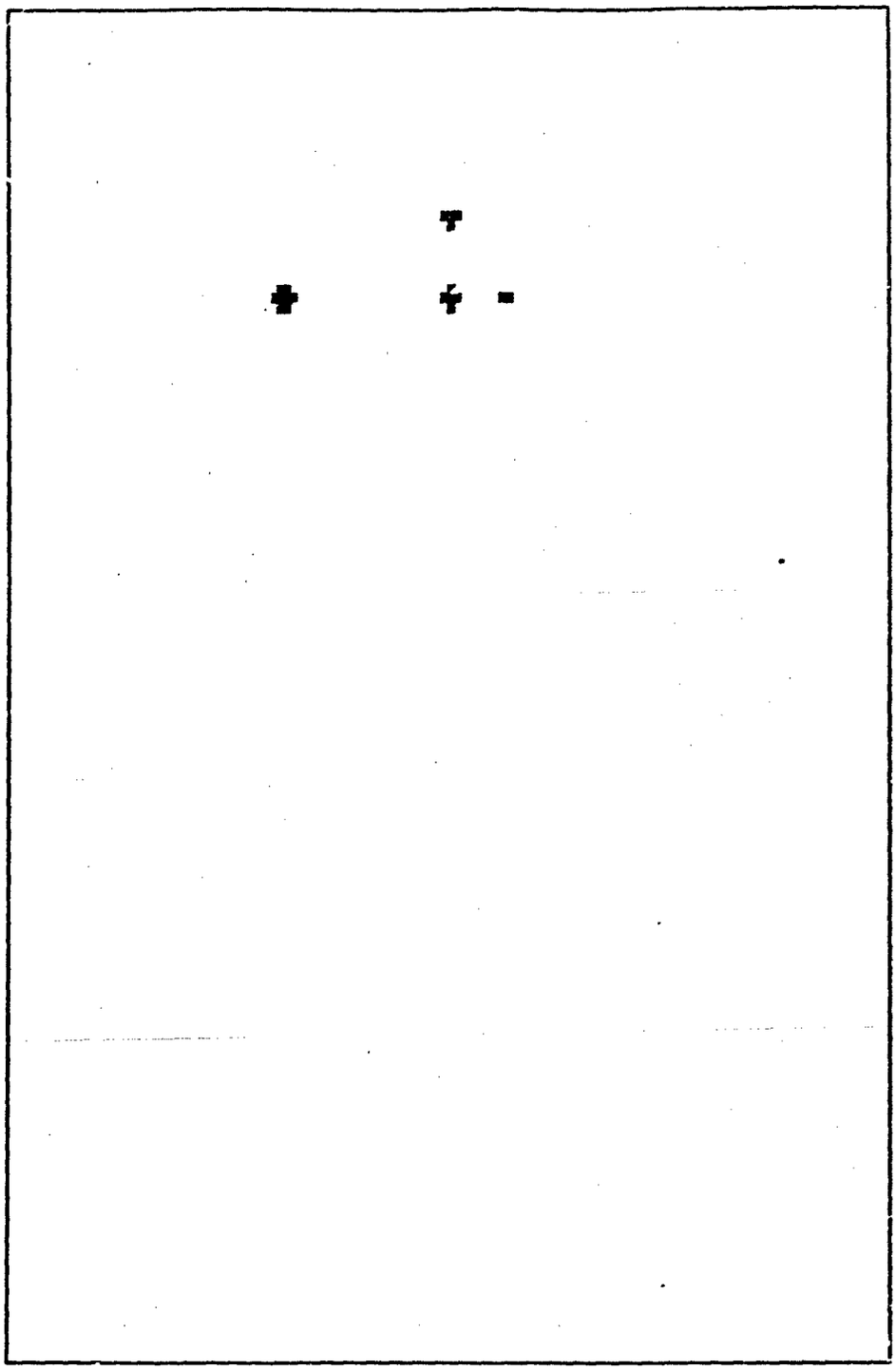


Figure 43. Subroutine FFTI5; Threshold = 150.

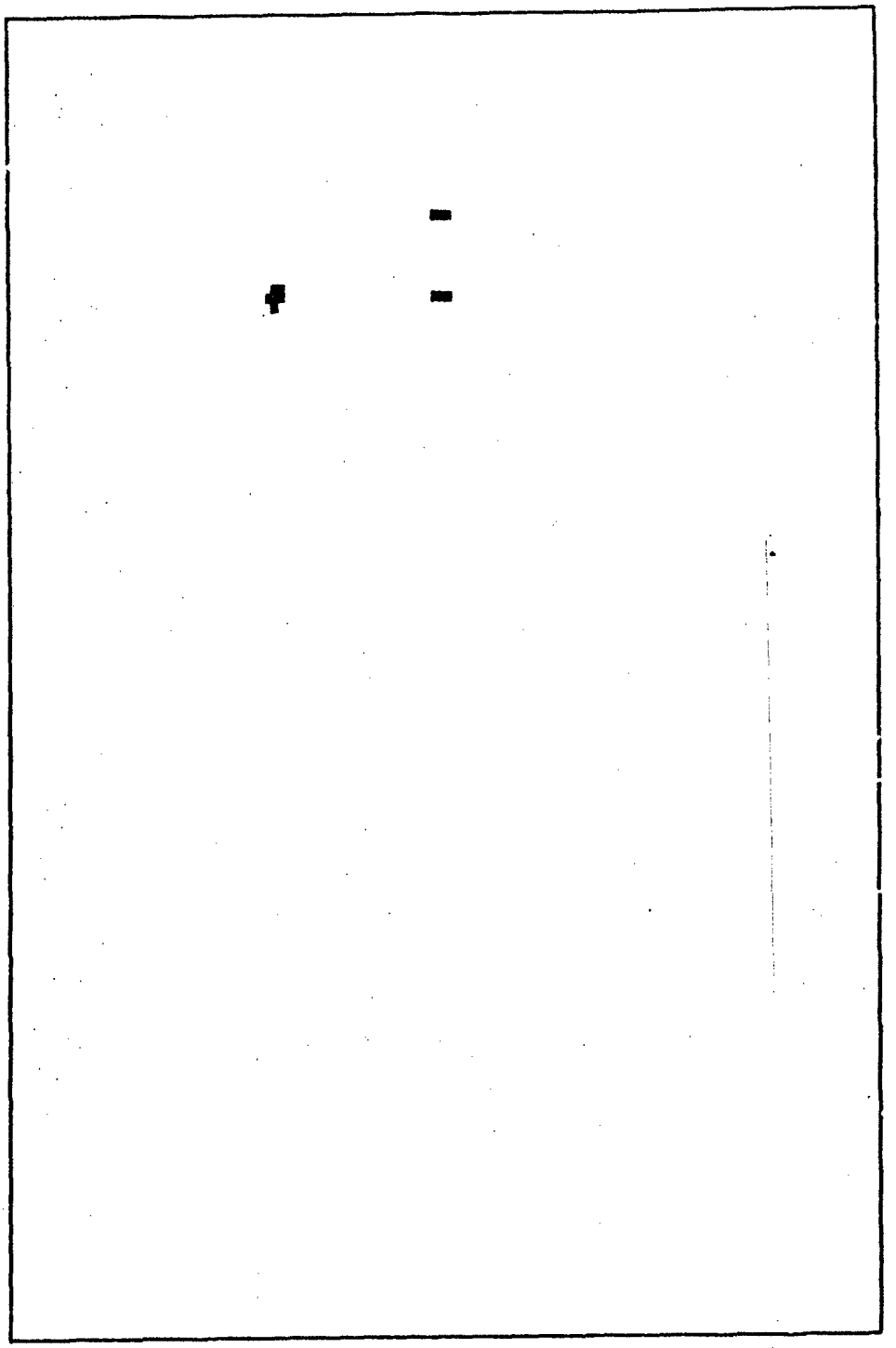


Figure 44. Subroutine FFTI5; Threshold = 90.

# VI. Conclusions and Recommendations

## Conclusions

The major conclusion of this thesis is that no other orthogonal transform should be substituted for the Fourier transform in the application of digital signal processing techniques in TTI radar imaging. The Karhunen-Loeve transform has no fast computational algorithm. The Hankel transform was not usable for implementation since it is a two-dimensional transform. The Mellin and Cosine (sine) transforms employ the FFT to compute the transform, thus no speed advantage would be realized for either of these two transforms. It was found that the Walsh power spectrum computed using the fast Walsh (Hadamard) transform, did not isolate the individual scatterers of a complex target as the Fourier spectrum was able to do. The conversion from the Walsh sequency domain to the Fourier frequency domain was found to be computationally excessive and more than offset the high speed advantage of the FWT/FHT. The assertion made by Robinson that there exists a linear transformation between the Walsh power spectrum and the Fourier power spectrum was found and proved to be incorrect.

The fixed-point FFT algorithm was the fastest implementation for TTI signal processing. It was considered that the increase in speed without a significant increase in error was significant. It is believed that execution will be even faster when programmed on the DEC PDP-11/40 minicomputer.

## Recommendations

It is recommended that the scale-invariance property of the Mellin transform using exponential sampling and the FFT be studied to

determine if it is feasible for compensating for the nonlinearities of the Doppler shift frequency.

It is recommended that hardware implementations of the Cosine (Sine) transform be investigated for implementation into the TTI imaging system.

Although no specific system has yet been designed and built, some theoretical systems have been designed. They utilize very sophisticated tactical radars, but well within the state-of-the-art. It is recommended that these or possibly other systems be designed and evaluated identifying important parameters and operating characteristics. Additionally, noise and statistical analysis of such systems should be conducted.

## Bibliography


1. Ahmed, N., A. L. Abdussattar, K. R. Rao, "Efficient Computations of the Walsh Hadamard Transform Spectral Modes," Proceedings of the 1973 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, 276-279, (April, 1973).

2. Admed, N., R. M. Bates., "A Power Spectrum and Related Physical Interpretation for the Multi-Dimensional BIFORE Transform," Proceedings of the 1971 Symposium on the Application of Walsh Functions, Washington D. C.: Naval Research Laboratories, 47-50, (13-15 April, 1971).

3. Ahmed, N., T. Natarajan, K. R. Rao, R. B. Schultz, "A Time-Varying Power Spectrum," IEEE Transactions on Audio and Electroacoustics, Vol. AU-19, No. 4: 327-328 (December 1971).

4. Ahmed, N., T. R. Natarajan, K. R. Rao, "Discrete Cosine Transform; IEEE Transactions on Computers, Vol C-23, No 7: 90-93 (January, 1974).

5. Ahmed, Nasis, K. R. Rao, "Discrete Fourier and Hadamard Transforms," Electronics Letters, Vol. 6, No. 7: 221-224, (April, 1970).

6. Ahmed, Nasis, K. R. Rao, A. L. Abdussattar, "BIFORE or Hadamard Transform" IEEE Transactions on Audio and Electroacoustics, AU-19, 3: 225-234 (September, 1971).

7. Andrews, Harry, "A High-Speed Algorithm for the Computer Generation of Fourier Transforms," IEEE Transactions on Computers; C-17, No. 4: 373-375 (April, 1968).

8. Andrews, Harry C., K. L. Caspari, "A Generalized Technique for Spectral Analysis", IEEE Transactions on Computers, C-19: 16-25, (January , 1970).

9. Andrews, Harry C., Computer Techniques on Image Processing, Academic Press: New York, 1970.

10. Angel, Edward S., "Image Representation and Transform Coding" Unpublished Lecture Notes.

11. Angel, Edward S., "Fourier Theory and Picture Processing" Unpublished Lecture Notes.

12. Bergland, G. D., "A Guided Tour of the Fast Fourier Transform," IEEE Spectrum; 41-52, (July, 1969).

13. Berkowitz, Modern Radar (Analysis, Evaluation, and System Design), John C. Wiley & Sons, New York. 1965.

14. Bingham, Christopher, Michael D. Godfrey, John W. Tukey, : Modern Techniques of Power Spectrum Estimation", IEEE Transactions on

Audio and Electroacoustics, Vol. AU-15, No. 2: 56-66, (June, 1967).

15. Blachman, Nelson M., "Sinusoids Versus Walsh Functions," Proceedings of the IEEE, Vol. 62, No. 3: 346-354, (March, 1974).

16. Blachman, Nelson M., "Spectral Analysis with Sinusoids and Walsh Functions," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-7, No. 5: 900-905, (September, 1971).

17. Blacksmith, P. Jr. R. E. Hiatt, R. B. Mack, "Introduction to Radar Cross-Section Measurement," Proceedings of the IEEE, Vol. 57: 901-920, (August, 1965).

18. Brigham, E. O., R. E. Morrow, The Fast Fourier Transform (FFT) LIV Electro Systems, Inc. Technical Report G 3000.17.04: Greenville, N. C.: LIV Electro Systems, Inc., (27 August, 1969) AD 697 872.

19. Carl, Joseph W., Generalized Harmonic Analysis for Pattern Recognition: A Biologically Derived Model. MS Thesis GE/BE/FOS-1. WPAFB, Ohio: Air Force Inst of Tech, (September, 1969).

20. Carl, J. W., et al "A Hybrid Walsh Transform Computer" AMRL, Technical Report. AMRL-TR-72-31 WPAFB, Ohio, (1 September 1972) AD 770 762.

21. Carl, Joseph W., Instructor, Air Force Institute of Technology, WPAFB, Ohio, Personal Communication, (1977).

22. Casasent, David, Demetri Psaltis, "New Optical Transforms for Pattern Recognition", Proceedings of the IEEE Vol 65, No. 1: 77-84, (January 1977).

23. Claire, E. J., S. M. Farber, R. R. Green, "Acoustic Signature Detection and Classification Techniques" 124-129.

24. Cochran, W. T., J. W. Cooley, D. C. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling Jr., D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the Fourier Transform?" IEEE Transactions on Audio and Electroacoustics, AU-15; No 2: 45-55, (June, 67).

25. Codley, J. W., J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series." Mathematics of Computation, No. 90: 297-301, 1965.

26. Crispin, J. W. Jr., A. L. Maffett, "Radar Cross-Section Estimation for Complex Shapes," Proceedings of the IEEE, Vol. 57: 972-982 (August, 1965).

27. Crittenden, R. B., "On Hadamard Matrices and Complete Orthonormal Systems", Proceedings of the 1973 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, 82-84, (16-18 April, 1973).

28. Davenport, William B., Jr., William L. Root, An Introduction to the

Theory of Random Signals and Noise, McGraw-Hill Book Co., Inc. New York, (1958).

29. Davies, Anthony C., "On the Definition and Generation of Walsh Functions", IEEE Transactions on Computers, C-18: 187-189, (Feburary, 1972).

30. Deutsch, Ralph, Estimation Theory, Prentice-Hall, Inc., Englewood Cliffs, N. J. 1965.

31. Dike, G., Improvement in Aircraft Imaging Simulation. Draft Syracuse Research Corporation Technical Note. SRC TN 77-016, Syracuse, N. Y.: Syracuse Research Corporation, (January, 1977).

32. Dike, G., Aircraft Image Simulation Program. Draft Syracuse Research Corporation Technical Note. SRC TN 76-241. Syracuse, N.Y.: Syracuse Research Corporation, (October, 1976).

33. DiFranco, J. V., W. L. Rubin, Radar Detection,. Prentice-Hall, Inc., Englewood Cliffs, N. J. 1968.

34. Ferrie, James F., Albert H. Nuttall, "Comparison of Four Fast Fourier Transform Algorithms". NUSC Report No. 4113. New Port, RI: Naval Underwater Systems Center (3 June, 1971) AD 720 034.

35. Fisher, James R., FORTRAN Program for Fast Fourier Transform. Naval Research Laboratory's Report No. 7041, Washington, D. C.: Naval Research Laboratory, (April 16, 1970) AD 706 003).

36. Gentleman, W. M., G. Sande, "Fast Fourier Transforms for Fun and Profit," 1966 Fall Joint Computer Conference, AFIPS Proc., Vol. 29, Washington, D. C.: Spartan, 563-578 (1966).

37. Gibbs, J. E., "Discrete Complex Functions", Proceedings of the 1970 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, 106-122, (31 March-3 April, 1970).

38. Gibbs, J. E., F. R. Pichler, "Comments on Transformation of 'Fourier' Power Spectra into 'Walsh" Power Spectra.", Proceedings of the 1971 Symposium on the Application of Walsh Functions, Washington, D. C.: Nasal Research Laboratories, 51-54 (13-15 April, 1971).

39. Goggins, William B, Jr., Identification of Radar Targets by Pattern Recognition, PhD Dissertation. WPAFB, Ohio: Air Force Inst of Tech, (June, 1973).

40. Good, I. J., "The Interaction Algorithm and Practical Fourier Analysis." Journal of the Royal Statistical Society, 20: 361-372, (1958).

41. Gulamhusein, Mohamedn, Frank Fallside, "Short-Time Spectral and Autocorrelation Analysis in the Walsh Domain", IEEE Transactions on Information Theory, IT-19, No. 5,: 615-673 (September 1973).

42. Harabick, Robert M., Norman Griswold, Nimitra Kattiyakulwanich, "A Fast Two-Dimensional Karhunen-Loeve Transform" Efficient Transmission of Dictorial Information, Vol. 66; Society of Photo-Optical Instrumentation Engineers, Palos Verdes Estates, California, 144-159, (1975).

43. Harmuth, H. F., "A Generalized Concept of Frequency and Some Applications." IEEE Transactions on Information Theory, IT-14, No 3: 375-382, (May, 1968).

44. Harmuth, Henning F., Transmission of Information by Orthoyonal Signals, Springer-Verlag New York Inc., New York, 1969.

45. Henderson, Keith L., "Some Notes on the Walsh Functions." IEEE Transactions on Electronic Computers, EC-13: 50-52 (February, 1964).

46. Hoy, R. W., Fast Fourier Transform. NUWL Report TD No. 20, New Port, RI: Naval Underwater Weapons Research and Engineering Station, (August, 1968).

47. Hynes, Robert, Robert E. Gardner, "Doppler Spectra of S-Band." Supplement to IEEE Transactions on Aerospace and Electronics Systems, AES-3, No. 6, 356-365, (November 1967).

48. Jain, Anilk, "Digital Image Coding/Data Compression" Unpublished Lecture Notes.

49. Kaneko, Toyohisa, Bede Liu, Computation Error in Fast Fourier Transform. Proceedings Third Asiloma Conference on Circuits and Systems, Dec 10-12, 1969, (November, 1970) AD 717 236.

50. Kolman, Bernard, Elementary Linear Algebra, The MacMillan Company, New York 1970.

51. Ksienski, Aharon A., Yau-tang Lin, Lee James White, "Low Frequency Approach to Tartet Identification", Proceedings of the IEEE, Vol.63, No. 12 1651-1660, (December, 1975).

52. Lackey, Robert B., David Meltzer, "A Simplified Definition of Walsh Functions." IEEE Transactions on Computers,: 211-213, (February, 1971).

53. Manz, Joseph W., "A Sequency-Ordered Fast Walsh Transform", IEEE Transactions on Audio and Electroacoustics, AU-20, No. 3,: 204-205, (August, 1972).

54. Mayard, Harry W., "An Evaluation of Ten Fast Fourier Transforms", Atmospheric Sciences Lab. (TR-ECOM-5476), White Sands Missle Range, N. M.: USA Electronics Command, Ft. Monmouth, N. J. (March, 1973) AD 758 451.

55. Meck, Norman C., "Application of Discrete Walsh Functions to Digital Filter Techniques" MS Thesis, Monterey, California:

June 1972. Naval Post Graduate School, AD 749 044.

56. Moceyunnas, A. J., M. L. Rafael, Radar Imaging Techniques (U), Vol. I: Review of Imaging Techniques and Theories, and Results from Objects 4392, 5721, and 5625 (U). Semi Annual Report prepared by Syracuse Research Corporation for Lincoln Laboratories Under Contract F 19628-FO-C-0230, SURC TR 72-109, (ESD TR 72-262, October 1972, (Secret).

57. Moceyunnas, A. J., Z. Zenon, "Radar Target Scattering Diagnosis With Wideband Measurements and Coherent Processing." Solicited Proposal prepared by Syracuse Research Corporation, SRC TR 70-150, October 1970, revised April 1972, reprinted April 1975.

58. Nathanson, Fred E., Radar Design Principles (Signal Processing and the Environment). McGraw-Hill Book Company: New York, 1969.

59. Noble, Ben. Applied Linear Algebra, Prentice-Hall Inc., Englewood Cliffs, N. J., 1969

60. Ohnsorg, F. R., "Application of Walsh Functions to Complex Signals" Proceedings of the 1970 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, 123-127, (31 March - 3 April, 1970).

61. Ohnsorg, Ferdinand R., "Spectral Modes of the Walsh-Hadamard Transform," Proceedings of the 1971 Symposium on the Application of Walsh Functions, Washington D. C.: Naval Research Laboratories, 55-59, (13-15 April, 1871).

62. Oppenheim, Alan V., Ronald W. Schafer, Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1975.

63. Peterson, H. L., "Generation of Walsh Functions", Proceedings of the 1970 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, 55-59, (13-15 April, 1971).

64. Polge, R. J., B. K. Bhagavon, J. M. Carswell, E. R. McKee, "Theory and Implementation of Fast Fourier and Hadamard Transforms", UAH Report #143, Huntsvale, Alabama: University of Alabama, (April, 1973). AD 762-437.

65. Pratt, W. K., et al, "Hadamard Image Coding." Proceedings of the IEEE, 57: 58-68 (January, 1969).

66. Pratt, William K., Julius Kane, Harry C. Andrews, "Hadamard Transform Image Coding", Proceedings of the IEEE, Vol. 57, No. 1: 58-68 (January, 1969).

67. Radar, Charles M., "An Improved Algorithm for High Speed Autocorrelation with Application to Spectral Estimations", IEEE Transactions on Audio and Electroacoustics, Vol. AU-18, No. 4: 439-441 (December, 1970).

68. Rafael, Murray L., "Aircraft Identification by Radar Imaging Feasibility Study". Syracuse Research Corporation Technical Report TR 74-223: Syracuse, N. Y.: Syracuse Research Corporation, (September, 1974).

69. Richards, Paul I., "Computing Reliable Power Spectra," IEEE Spectrum; 83-90, (January, 1967).

70. Rihaczek, August W., Principles of High-Resolution Radar, McGraw-Hill Book Company, New York, 1969.

71. Ritea, H. Barry, A Comparison of FFT Algorithms, (Systems Development Corporation), Technical Memorandum TM 4857/100, Santa Monica, CA: System Development Corporation, (5 January, 1972).

72. Robinson, Giiner S., "Logical Convolution and Discrete Walsh and Fourier Power Spectra " IEEE Transactions on Audio and Electroacoustics. AU-20. No. 4: 271-279, (October, 1972).

73. Robinson, G. S., "Discrete Walsh and Fourier Power Spectra.", Proceedings of the 1972 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, 298-309.

74. Robinson, Guner S., "Logical Convolution and Discrete Walsh and Fourier Power Spectra" IEEE Transactions on Audio and Electroacoustics, AU-20, No. 4: 271-280, (October, 1972).

75. Schwartz, Mischa, Leonard Shaw, Signal Processing: Discrete Spectral Analysis, Detection, and Estimation. McGraw-Hill Book Co.: New York, 1975.

76. Sentman, Davis D. "Basic Elements of Power Spectral Analysis." Icwa University Report 74-5, Iowa City, Iowa: The University of Iowa, 16 Jan 1974. AD 773 936.

77. Shanks, J. L., "Computations of the Fast Walsh Transform." IEEE Transactions on Computers; C-18: 457-459. (May, 1969).

78. Shum, Y. Y., A. R. Elliot, "Computation of the Fast Hadamard Transform," Proceedings of the 1972 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, (177-180).

79. Siemens, K. H., R. Kitai, "Walsh Series to Fourier Series Conversion.", Proceedings of the 1973 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, 295-297 (16-18 April, 1973).

80. Siemens, Karl H , Reuven, Kitai, "A onrecursive Equation for the Fourier Transform of a Walsh Function." IEEE Transactions on Electromagnetic Compatibility, Vol. EMC-15, 81-83, (May, 1973).

81. Singleton, Richard C., The Fast Fourier Transform. SRI Project 181531-132, Menlo Park, CA: Stanford Research Institute, (December, 1976) (AD 643998).

82. Singleton, Richard C., "An Algorithm for Computing the Mixed Radix Fast Fourier Transform," IEEE Transactions on Audio and Electroacoustics, Vol. AU-17, No. 2: 93-103 (June, 1969).

83. Stratton, Robert D., Real-World Limitations of Radar Signatures for Target Identification. 76 USAF-ASEE Summer Faculty Research Program, Griffiss AFB, N. Y.: Rome Air Development Corporation, (13 August, 1976).

84. Swartwood, Richard V., A Two-Dimensional Walsh Transform Computer. MS Thesis, GE/EE/72-25. WPAFB, Ohio: Air Force Inst of Tech., (December, 1972).

85. Tranter, C. J., Integral Transforms in Mathematical Physics. John Wiley and Sons, Inc.: New York, 1956.

86. Yuen, C. K., "A Fast Algorithm for Computing Walsh Power Spectrum". Proceedings of the 1971 Symposium on the Application of Walsh Functions, 279-283.

87. Walsh, J. L., "A Closed Set of Normal Orthogonal Functions", Amer Jour of Math, Vol. 45: 5-24, (1923).

88. Webb, Carol, Practical Use of the FFT Algorithm in Time-Series Analysis. ARL Technical Report TR 70-22, Austin, TX: University of Texas, (22 June, 1970).

89. Welch, L. R., "Walsh Function and Hadamard Matrices:, Proceedings of the 1970 Symposium on the Application of Walsh Functions, Washington, D. C.: Naval Research Laboratories, 31 March - 3 April, 1970, 163-165.

90. Welch, Peter D., "The Use of Fast Fourier Transforms for the Estimation of Power Spectra: A Method Based on Time Averaging Over Slot, Modified Periodograms", IEEE Transactions on Audio and Electroacoustics, AU-15, No. 2: 70-73, (June, 1967).

91. Welch, Peter D., "A Fixed-Point Fast Fourier Transform Error Analysis," IEEE Transactions on Audio and Electroacoustics, Vol AU-17, No. 2: 151-157 (June 1969).

92. Wood, Richard J., Rome Air Development Center, Griffiss AFB, New York, Project Engineer and Thesis Sponsor. Personal Communication. 1977.

Appendix A

## Main Program TGTID

Program TGTID reads in the radar parameters RLOC, PRI, STARTF, and DELTAF which describe the location of the radar relative to the target and the waveform desired. NBURST and NFREQ are the number of bursts transmitted to the target and the number of frequencies with each burst, respectively. Two other parameters NSCAT and SCORM describe the target, where NSCAT is the number of scatterers within the target and SCPRM is the set of seven parameters of each scatterer. The first three parameters are the coordinates of the scatterer on the coordinate system, $\sigma_i$ is the fourth parameter, and the last three are cosine weighting terms.

Program TGTID calls Subroutine CINIT which computes eight additional parameters based on the input parameters. They determine the size as a power of two and other characteristics of the time and frequency arrays. Subroutine SLANTV is called which returns the simulated target Doppler returns. Then, subroutine CIMAGE processes the Doppler signal returns and displays the synthetic image by calling either of the display subroutines (Appendix G).

Program TGTID finally prints out the radar and scatterer parameters along with three calculated values: Number of Words/Record, Number of Records, and the maximum power.

```
      PROGRAM TGTID(INPUT=/80,OUTPUT=/132,
     *TAPE5=INPUT,TAPE7=OUTPUT,TAPE6,TAPE8,PLOT)
C
C           SIMULATE RADAR IMAGE RETURN
C
      COMMON /RADAR/ RLOC(3),PRI,STARTF,DELTAF
      COMMON /SCATR/ NSCAT,SCPRM(7,30)
      COMMON /ANGLE/ A0,B0,G0,ADOT,BDOT,GDOT
      COMMON /LIMIT/ NBURST,NBORD,NBSZ,NFREQ,NFORD,NFSZ
      COMMON /PIVOT/ NDSZ,ND1,ND2,ND3,NTM
      COMMON /IMAGE/ RMAX,NWRDS,NRECS
      COMMON /PARAM/ PI,C,H1,H2
      COMMON /UNITS/ NVLT
      COMMON /POINTS/ NIMG
      COMMON DATA(10000)
      DATA PI/3.1415926/,C/299792500./,H1/.54/,H2/.46/
      DATA NVLT/6/
      DATA NIMG/8/
      DATA NDSZ/10000/
C
C
C           INPUT PARAMETERS
C           SIMULATE RADAR RETURN
C           OUTPUT IMAGE PARAMETERS
C
      READ*, RLOC,PRI,STARTF,DELTAF
      READ*, A0,B0,G0,ADOT,BDOT,GDOT
      READ*, NBURST,NFREQ
      READ*, NSCAT
      READ(5,*) ((SCPRM(I,K),I=1,7),K=1,NSCAT)
C
      CALL CINIT
      CALL SLANTV
      CALL CIMAGE(DATA,ND1,ND2,ND3,NTM)
C
      WRITE(7,12) RLOC,PRI,STARTF,DELTAF,
     *            A0,B0,G0,ADOT,BDOT,GDOT,
     *            NBURST,NFREQ,
     *              NSCAT,((SCPRM(I,K),I=1,7),K=1,NSCAT)
      WRITE(7,13) RMAX,NWRDS,NRECS
C
C
   12 FORMAT(///,T30,"IMAGE PARAMETERS",/,T30,15(1H-),//,
     *   T10,"RADAR PARAMETERS     =",3F10.0,F10.6,2E12.6,/,
     *   T10,"ANGLE DATA           =",5F10.2,/,
     *   T10,"NO. OF BURSTS        =",I10,/,
     *   T10,"NO. OF FREQUENCIES   =",I10,/,
     *   T10,"NO. OF SCATTERERS    =",I10,/,
     *   T10,"SCATTERER PARAMETERS =",(T32,7F10.3))
   13 FORMAT(///,T10,"MAXIMUM POWER         =",F10.2,/,
     *          T10,"NO. OF WORDS/RECORD  =",I10,/,
     *          T10,"NO. OF RECORDS       =",I10,/)
      STOP  $ END
```

## Appendix B

### Support Subroutines

#### Subroutine CINIT

Subroutine CINIT calculates NBORD, NBSZ, NFSZ, NDSZ, ND1, ND2, ND3, and NIM based on the values of NBURST and NFREQ. NFSZ is the size of the frequency response data array and NBSZ is the size of the time response data array. They are related to NBORD by the following relationships

$$NFSZ = 2^{NBORD} \qquad (B.1)$$

$$NFSZ = 2^{NBORD} \qquad (B.2)$$

```
      SUBROUTINE INIT
      COMMON /LIMITS/ NVRUPST,NVORD,NVSZ,NFSED,NFORD,NFST
      COMMON /PRVATE/ NVSZ,NV1,NV2,NV3,NTM
C
C
C           INITIAL CALCULATIONS
C
      DO 26 I=1,8
      IF(2**I.GE.NFSED) GO TO 27
   26 CONTINUE
   27 NFORD=I
      NFST=2**I
      NV1=NVRUPST
      NV2=2
      NV3=NFST
      NTM=1
      DO 24 I=1,8
      IF(NV1*NV2*NV3.LE.NVSZ) GO TO 25
      NV3=NV3/2
   24 NTM=NTM*2
   25 DO 22 I=1,8
      IF(2**I.GE.NVRUPST) GO TO 23
   22 CONTINUE
   23 NVORD=I
      NVSZ=2**I
C
      RETURN
      END
```

Subroutines SLANTV, CTRAN, and DOTP

Subroutine SLANTV is the driving subroutine that generates the simulated "slant voltages", that is, the Doppler signal received by the radar set. Subroutine SLANTV calls Subroutines HAMWGT, CTRAN, DOTF, FFT6 (Appendix C), and ROLL. The slant voltage values are written to a temporary file (NVLT) which is returned to Program TGTID for processing. Subroutine SLANTV uses the radar and target parameters read in by Program TGTID and computed by Subroutine CINIT. No other input is required.

```
      SUBROUTINE SLANTV
      COMMON /RADAR/ RLOC(3),PRI,STARTR,DELTAR
      COMMON /SCATR/ NSCAT,SCPRM(7,30)
      COMMON /ANGLE/ AA,BB,GG,AROT,BROT,GROT
      COMMON /LIMIT/ NAZPST,NAZPRD,NRNG,NFREQ,NFCPRD,NFFT
      COMMON /PARAM/ PI,C,W1,W2
      COMMON /UNITS/ NVLT
      COMMON /VOLT(256,2),W(256),RANGE(30),CW(30)
      COMMON S(3),T(3),PHI(3)
      COMPLEX VOLTC(256)
C
C
C           GENERATE SLANT RANGE VOLTAGE VALUES
C           -----------------------------------
C
C               COMPUTE HAMMING WEIGHT VECTOR
C               CLEAR VOLTAGE ARRAY
C               COMPUTE ROTION ANGLES
C               COORDINATE TRANSFORMATION
C               COMPUTE RANGE VECTOR
C               COMPUTE WEIGHT VECTOR - DOT PRODUCT
C               COMPUTE FREQUENCY COMPONENETS
C               INVERSE TRANSFORM, SWAP HA_VES
C               OUTPUT TO TEMPORARY FILE
C
   40 CALL HAMWGT(NFREQ,W)
      PI2C=4.*PI/C
C
      DO 48 L=1,NAZPST
      DO 41 I=1,NFFT
      VOLT(I,1)=0.
   41 VOLT(I,2)=0.
      PR=PRI*(L-1)
      PHI(1) = AA + AROT*PR
      PHI(2) = BB + BROT*PR
      PHI(3) = GG + GROT*PR
      NFC=1
      DO 43 K=1,NSCAT
      CALL CTRAN(PHI,SCPRM(1,K),T,NFC)
      CALL CTRAN(PHI,SCPRM(5,K),S,NFC)
      DO 42 I=1,3
   42 T(I)=RLOC(I)-T(I)
      RANGE(K)=SQRT(T(1)**2+T(2)**2+T(3)**2)
   43 CALL DOTP(T,S,CW(K))
      DO 45 I=1,NFREQ
      PR=PI2C*(STARTR+DELTAR*(I-1))
      DO 45 K=1,NSCAT
      IF(CW(K).LE.0.) GO TO 45
      A=PR*RANGE(K)
      VOLT(I,1)=VOLT(I,1)+SCPRM(4,K)*SIN(A)*CW(K)*W(I)
      VOLT(I,2)=VOLT(I,2)+SCPRM(4,K)*COS(A)*CW(K)*W(I)
   45 CONTINUE
```

```
      CALL FFT5(-NFORD,VOLT(1,1),VOLT(1,2))
      CALL ROLL(NFS7,VOLT)
   48 WRITE(NVLT) ((VOLT(I,K),I=1,NFS7),K=1,2)
C
      REWIND NVLT
      RETURN
      END
```

```
      SUBROUTINE STRAN (PHI,PT,P,NFC)
      REAL RHO(3,3),PHI(3),PT(3),P(3)
C
C
C           COORDINATE TRANSFORMATION
C           -------------------------
C
C
      GO TO (30,40),NFC
C
C
C           INITIALIZE RHO MATRIX
C
   30 NFC=2
      SA=SIN(PHI(1))
      SB=SIN(PHI(2))
      SG=SIN(PHI(3))
      CA=COS(PHI(1))
      CB=COS(PHI(2))
      CG=COS(PHI(3))
C
      RHO(1,1) =  CA*CG - SA*SB*SG
      RHO(1,2) = -SA*CB
      RHO(1,3) =  SG*CA + SA*SB*CG
      RHO(2,1) =  SA*CG + CA*SB*SG
      RHO(2,2) =  CA*CB
      RHO(2,3) =  SA*SG - CA*SB*CG
      RHO(3,1) = -SG*CB
      RHO(3,2) =  SB
      RHO(3,3) =  CB*CG
C
C
C           MATRIX MULTIPLICATION
C           COMPUTE NEW COORDINATES
C
   40 DO 44 I=1,3
      P(I)=0.
      DO 44 K=1,3
   44 P(I) = P(I) + PT(K)*RHO(I,K)
C
C
      RETURN
      END
```

```
      SUBROUTINE DOTP (V1,V2,VAL)
      REAL V1(3),V2(3)
C
C
C          COMPUTE DOT PRODUCT OF 2 VECTORS
C
      VAL=0.
      A=0.
      B=0.
      DO 40 I=1,3
      A=A+V1(I)**2
      B=B+V2(I)**2
   40 VAL = VAL + V1(I)*V2(I)
      VAL=VAL/SQRT(A*B)
C
      RETURN
      END
```

## Subroutines ROLL, WROLL, AND IROLL

Subroutine ROLL rearranges the FFT exit array into normal viewing in Subroutines SLANTV and CIMAGE. This can be done either before or after the power spectrum is computed. Subroutines WROLL and IROLL perform the same function for the Fast Walsh/Hadamard transforms (Appendix D) and the fixed-point FFT's (Appendix E), respectively, in the appropriate CIMAGE subroutine.

```
      SUBROUTINE ROLL (NSZ,V)
      REAL V(256,2)
C
C
C          ROLL THE IMAGE
C
      NSZ2=NSZ/2
      DO 44 K=1,NSZ2
      L=K+NSZ2
      DO 44 I=1,2
      T=V(K,I)
      V(K,I)=V(L,I)
   44 V(L,I)=T
C
      RETURN
      END
```

```
      SUBROUTINE WROLL(NSZ,V)
C
      REAL V(256)
C
C
C     ROLL THE IMAGE
C
C
      NSZ2=NSZ/2
      DO 10 K=1,NSZ2
      L=K+NSZ2
      T=V(K)
      V(K)=V(L)
      V(L)=T
C
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE IROLL(NSZ,IVR,IVI)
      INTEGER IVR(256),IVI(256),T1,T2
C
C
C           ROLL THE IMAGE
C
      NSZ2=NSZ/2
      DO 44 K=1,NSZ2
      L=K+NSZ2
      T1=IVR(K)
      IVR(K)=IVR(L)
      IVR(L)=T1
      T2=IVI(K)
      IVI(K)=IVI(L)
      IVI(L)=T2
   44 CONTINUE
C
      RETURN
      END
```

Subroutine CIMAGE

Subroutine CIMAGE constructs the image from the simulated Doppler returns generated by Subroutine SLANTV. The data are first Hamming weighted as the sample values are read from the temporary data file (NVLT). The orthogonal transform is taken along constant range, the power spectrum computed, and the data are "flipped" by Subroutine ROLL. The "image" data are then written onto a second temporary file (NIMG). Subroutines PLOTID or BUFOUT (Appendix G) are then called to display the image. Subroutine CIMAGE is written in three versions; the first is used for floating point FFT subroutines, the second is for Walsh transform subroutines, and the third is for fixed-point FFT subroutines.

```
*
      SUBROUTINE STMAGE (PCR,ND1,ND2,ND3,NTM)
      COMMON /LIMIT/ NBURST,NPORD,NBSZ,NFREQ,NFORD,NFSZ
      COMMON /IMAGE/ PMAX,NWRDS,NRECS
      COMMON /UNITS/ NVLT
      COMMON /POINTS/ NIMG
      REAL PCR(ND1,ND2,ND3)
      REAL VOLT(256,2),W(256)
C
C
C          CONSTRUCT THE IMAGE
C          -------------------
C
C              COMPUTE HAMMING WEIGHT VECTOR
C              INPUT VOLTAGES
C              PIVOT  -  SLANT RANGE TO CROSS RANGE
C              EXTEND THE ARRAY
C              TAKE FFT ALONG CONSTANT RANGE
C              SWAP HALVES
C              COMPUTE POWER SPECTRUM
C              COMPUTE MAXIMUM POWER
C              COMPUTE FFT EXECUTION TIME
C              OUTPUT THE IMAGE
C
      CALL HAMWGT(ND1,W)
C
      TTIME=0.0
      DO 46 M=1,NTM
      NS=ND3*(M-1)
      DO 42 N1=1,ND1
      READ(NVLT) ((VOLT(I,J),I=1,NFSZ),J=1,2)
      DO 41 N3=1,ND3
      PCR(N1,1,N3)=VOLT(NS+N3,1)
   41 PCR(N1,2,N3)=VOLT(NS+N3,2)
   42 CONTINUE
      DO 46 N3=1,ND3
      DO 43 I=ND1,NBSZ
      VOLT(I,1)=0.
   43 VOLT(I,2)=0.
      DO 44 I=1,ND1
      VOLT(I,1)=PCR(I,1,N3)*W(I)
   44 VOLT(I,2)=PCR(I,2,N3)*W(I)
      STIME=SECOND(CP)
C-----INSERT FAST TRANSFORM CALL HERE-----------------
      CALL FFT3(NBORD,VOLT(1,1),VOLT(1,2))
C-----------------------------------------------------
      ETIME=SECOND(CP)
      PTIME=ETIME-STIME
      TTIME=TTIME+PTIME
      CALL ROLL(NBSZ,VOLT)
      DO 45 I=1,NBSZ
      VOLT(I,1)=SQRT(VOLT(I,1)**2+VOLT(I,2)**2)
   45 PMAX=AMAX1(PMAX,VOLT(I,1))
      WRITE(NIMG) (VOLT(I,1), I=1,NBSZ)
   46 CONTINUE
```

```
      CR REWIND NVLT
         REWIND NIMG
C
         WRITE(7,15)
      15 FORMAT(///,1X,"FFT5")
         WRITE(7,10) TTIME
      10 FORMAT(1X,"TRANSFORM EXECUTION TIME IS",F7.4," SEC.")
C
C-----INSERT IMAGE DISPLAY SUBROUTINE CAL_(S) HERE-----
         CALL PLOTID(NIMG,ND3,NTM,NBSZ)
         CALL BUFOUT(NIMG,ND3,NTM,NBSZ)
C-------------------------------------------------------
C
         NPTOS=NFSZ
         NWDS=NBSZ
         RETURN
         END
```

```
      SUBROUTINE CIMAGE (PCR,ND1,ND2,ND3,NTM)
      COMMON /LIMIT/ NBURST,NBORD,NBSZ,NFREQ,NFORD,NFSZ
      COMMON /IMAGE/ RMAX,NWRDS,NRECS
      COMMON /UNITS/ NVLT
      COMMON /POINTS/ NIMS
      REAL PCR(ND1,ND2,ND3)
      REAL VOLT(256,2),W(256)
      COMPLEX VOLTC(256)
      REAL VOLTR(256),VOLTW(256),T(256),B(10)
C
C
C          CONSTRUCT THE IMAGE
C          -------------------
C
C              COMPUTE HAMMING WEIGHT VECTOR
C              INPUT VOLTAGES
C              PIVOT  -  SLANT RANGE TO CROSS RANGE
C              EXTEND THE ARRAY
C              TAKE FHT/FWT ALONG CONSTANT RANGE
C              SWAP HALVES
C              COMPUTE POWER SPECTRUM
C              COMPUTE MAXIMUM POWER
C              COMPUTE FHT/FWT EXECUTION TIME
C              OUTPUT THE IMAGE
C
      NBSZ1=NBSZ+1
      NBSZ2=NBSZ*2
      NBSZM1=NBSZ-1
      NBORD1=ALOG(FLOAT(NBSZ2))/ALOG(2.)+.1
      CALL HAMWGT(ND1,W)
      TTIME=0.0
      DO 48 M=1,NTM
      NS=ND3*(M-1)
      DO 42 N1=1,ND1
      READ(NVLT) ((VOLT(I,J),I=1,NFSZ),J=1,2)
      DO 41 N3=1,ND3
      PCR(N1,1,N3)=VOLT(NS+N3,1)
   41 PCR(N1,2,N3)=VOLT(NS+N3,2)
   42 CONTINUE
      DO 46 N3=1,ND3
      DO 43 I=ND1,NBSZ
      VOLT(I,1)=0.
   43 VOLT(I,2)=0.
      DO 44 I=1,ND1
      VOLT(I,1)=PCR(I,1,N3)*W(I)
   44 VOLT(I,2)=PCR(I,2,N3)*W(I)
      DO 50 I=1,NBSZ
      VOLTC(I)=CMPLX(VOLT(I,1),VOLT(I,2))
      VOLTR(I)=CABS(VOLTC(I))
   50 CONTINUE
      DO 55 I=NBSZ1,NBSZ2
      VOLTR(I)=0.0
   55 CONTINUE
```

```
      STIME=SECOND(CP)
C-----INSERT FAST TRANSFORM CALL HERE-----------------
      CALL FHT1(VOLTR,NBSZ2,NBORD1,1)
C-----------------------------------------------------
      ETIME=SECOND(CP)
      RTIME=ETIME-STIME
      TTIME=TTIME+RTIME
      DO 49 I=1,NBSZ
      IF(I.EQ.1)GO TO 51
      IF(I.EQ.NBSZ)GO TO 47
      VOLTR(I)=VOLTR(2*I)**2+VOLTR(2*I+1)**2
      GO TO 45
   51 VOLTR(1)=VOLTR(1)**2
      GO TO 45
   47 VOLTR(NBSZ)=VOLTR(NBSZ2)**2
   45 RMAX=AMAX1(RMAX,VOLTR(I))
   49 CONTINUE
      CALL WROLL(NBSZ,VOLTR)
      WRITE(NIMG) (VOLTR(I), I=1,NBSZ)
   46 CONTINUE
   48 REWIND NVLT
      REWIND NIMG
C
      WRITE(7,15)
   15 FORMAT(///,1X,"FHT1")
      WRITE(7,10) TTIME
   10 FORMAT(1X,"TRANSFORM EXECUTION TIME IS",F7.4," SEC.")
C
C-----INSERT IMAGE DISPLAY SUBROUTINE CALL(S) HERE-----
      CALL BUFOUT(NIMG,ND3,NTM,NBSZ)
      CALL PLOTIO(NIMG,ND3,NTM,NBSZ)
C-----------------------------------------------------
      NRECS=NFSZ
      NWRDS=NBSZ
      RETURN
      END
```

```
      SUBROUTINE SIMAGE (PCR,ND1,ND2,ND3,NTM)
      COMMON /LIMITS/ NPULST,NTORD,NBSZ,NFREQ,NFORD,NFST
      COMMON /IMAGE/ RMAX,NWFDS,NREGS
      COMMON /UNITS/ NVLT
      COMMON /POINTS/ NIMG
      REAL PCR(ND1,ND2,ND3)
      REAL VOLT(256,2), W(256)
      INTEGER IVOLTR(256),IVOLTI(256)
      INTEGER IC(256)
C
C
C     CONSTRUCT THE IMAGE
C     -------------------
C
C          COMPUTE THE HAMMING WEIGHT VECTOR
C          INPUT THE VOLTAGES
C          PIVOT - SLANT RANGE TO CROSS RANGE
C          EXTEND THE ARRAY
C          TAKE INTEGER FFT ALONG CONSTANT RANGE
C          SWAP HALVES
C          COMPUTE POWER SPECTRUM
C          COMPUTE MAXIMUM POWER
C             COMPUTE INTEGER FFT EXECUTION TIME
C          OUTPUT THE IMAGE
C
      CALL COSINE(IC,NBSZ)
      CALL HAMWGT(ND1,W)
C
      TTIME=0.0
      FACTOR=2**15-1
      DO 48 M=1,NTM
      NS=ND3*(M-1)
      DO 42 N1=1,ND1
      READ(NVLT) ((VOLT(I,J),I=1,NFSZ),J=1,2)
      DO 41 N3=1,ND3
      PCR(N1,1,N3)=VOLT(NS+N3,1)
   41 PCR(N1,2,N3)=VOLT(NS+N3,2)
   42 CONTINUE
      DO 46 N3=1,ND3
      DO 43 I=ND1,NBSZ
      VOLT(I,1)=0.
   43 VOLT(I,2)=0.
      DO 44 I=1,ND1
      VOLT(I,1)=PCR(I,1,N3)*W(I)
   44 VOLT(I,2)=PCR(I,2,N3)*W(I)
      DO 50 I=1,NBSZ
      IVOLTR(I)=VOLT(I,1)*FACTOR/NBSZ
      IVOLTI(I)=VOLT(I,2)*FACTOR/NBSZ
   50 CONTINUE
C
      STTME=SECOND(CP)
```

```
C-----INSERT TRANSFORM SUBROUTINE CALL HERE-----------
      CALL FFTIF(IVOLTR,IVOLTI,IC,NPORD,NBSZ)
C----------------------------------------------------
      ETIME=SECOND(CP)
      RTIME=ETIME-STIME
      TTIME=TTIME+RTIME
      CALL IPOLL(NBSZ,IVOLTR,IVOLTI)
      DO 45 I=1,NBSZ
      IVOLTR(I)=SHIFT(IVOLTR(I)**2+IVOLTI(I)**2,-15)
      VOLT(I,1)=IVOLTR(I)
   45 PMAX=AMAX1(PMAX,VOLT(I))
      WRITE(NIMG) (VOLT(I,1), I=1,NBSZ)
   46 CONTINUE
   48 REWIND NVLT
C
      REWIND NIMG
      WRITE(7,15)
   15 FORMAT(///,1X,"FFTIS")
      WRITE(7,10) TTIME
   10 FORMAT(1X,"TRANSFORM EXECUTION TIME IS",F7.4," SEC.")
C
C-----INSERT IMAGE DISPLAY SUBROUTINE CALL(S) HERE-----
      CALL PLOTIO(NIMG,ND3,NTM,NBSZ)
      CALL BUFOUT(NIMG,ND3,NTM,NBSZ)
C----------------------------------------------------
C
      NREQS=NFSZ
      NWRQS=NBSZ
      RETURN
      END
```

Subroutine HAMWGT

This subroutine generates a Hamming weight vector (Equation (23)) for sidelobe control. Subroutine HAMWGT is used in Subroutines SLANTV and CIMAGE.

```
      SUBROUTINE HAMWGT (N,W)
      COMMON /PARAM/ PI,C,H1,H2
      REAL W(1)
C
C
C           COMPUTE HAMMING WEIGHT VECTOR
C           BELL CURVE FOR SIDELOBE REDUCTION
C
      N02=N/2
      M=N02+1
      P=2.*PI/(N-1)
C
      W(M)=1.
      DO 40 I=1,N02
      X = H1 + H2*COS(P*I)
      W(M+I)=X
   40 W(M-I)=X
C
      RETURN
      END
```

## Appendix C

### Floating Point FFT Subroutines

Subroutines SLANTV and CIMAGE (Version 1) call an FFT subroutine. The data must be placed into the proper form before the specific FFT subroutine is called. Each subroutine is documented internally and will not be discussed further.

Reordering subroutines are listed in Appendix G.

```
      SUBROUTINE FFT1A(A,M,N,IS)
C
C        A = INPUT ARRAY OF SAMPLES
C        N = 2 ** M =NUMBER OF SAMPLES
C        M = LOG2(N)
C       IS = -1, FORWARD TRANSFORM
C          = +1, INVERSE TRANSFORM (UNNORMALIZED)
C        TIMING=  .0033*LOG2(N)
C
      DIMENSION A(N)
      COMPLEX A,U,W,T
      DATA PI/3.14159265/
      NV2=N/2
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF(I.GE.J) GO TO 10
      T=A(J)
      A(J)=A(I)
      A(I)=T
   10 K=NV2
   20 IF(K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 20
   30 J=J+K
      DO 80 L=1,M
      LE=2**L
      LE1=LE/2
      U=CMPLX(1.0,0.)
      IF(IS) 40,50,50
   40 W=CMPLX(COS(PI/LE1),-SIN(PI/LE1))
      GO TO 60
   50 W=CMPLX(COS(PI/LE1),SIN(PI/LE1))
   60 DO 80 J=1,LE1
      DO 70 I=J,N,LE
      IP=I+LE1
      T=A(IP)*U
      A(IP)=A(I)-T
   70 A(I)=A(I)+T
   80 U=U*W
      RETURN
      END
```

```
      SUBROUTINE FFT1B(A,M,N,TS)
C
C     THIS FFT SUBROUTINE IS A MODIFIED VERSION
C     OF FFT1A.  IT ELIMINATES THE FIRST SET OF
C     COMPLEX MULTIPLIES IN THE DECIMATION-IN-
C     TIME ALGORITHM.
C
C     THE VARIABLES HAVE THE SAME MEANING AS IN
C     THE FFT1A SUBROUTINE.
C
      DIMENSION A(N)
      COMPLEX A,U,W,T
      DATA PI/3.141592653/
      NV2=N/2
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF(I.GE.J) GO TO 10
      T=A(J)
      A(J)=A(I)
      A(I)=T
   10 K=NV2
   20 IF(K.GE.J)GO TO 30
      J=J-K
      K=K/2
      GO TO 20
   30 J=J+K
      DO 40 I=1,N,2
      IP=I+1
      T=A(IP)
      A(IP)=A(I)-T
   40 A(I)=A(I)+T
      DO 50 L=2,M
      LE=2**L
      LE1=LE/2
      U=CMPLX(1.0,0.0)
      IF(IS) 41,42,42
   41 W=CMPLX(COS(PI/FLOAT(LE1)),-SIN(PI/FLOAT(LE1)))
      GO TO 45
   42 W=CMPLX(COS(PI/FLOAT(LE1)),SIN(PI/FLOAT(LE1)))
   45 DO 50 J=1,LE1
      DO 50 I=J,N,LE
      IP=I+LE1
      T=A(IP)*U
      A(IP)=A(I)-T
   50 A(I)=A(I)+T
   60 U=U*W
      RETURN
      END
```

```
      SUBROUTINE FFT2(X,NSTAGE,N,SIGN)
C
C     DEFINITION OF VARIABLES:
C      INPUT:
C         X(2,1024) : DATA INPUT IN COLUMN 1
C         NSTAGE = POWER OF TWO WHICH N IS
C         N = 2**NSTAGE
C         SIGN = -1, FORWARD TRANSFORM
C              = +1. INVERSE TRANSFORM
C      OUTPUT:
C         X(2,1024) : FORWARD-TRANSFORMED DATA OUTPUT IN COL 1
C                   : INVERSE-TRANSFORMED DATA OUTPUT IN COL 2
C
      COMPLEX X(2,N), W
      INTEGER R, SIGN
      N2=N/2
      FLTN=N
      PHI2N=6.2831853/FLTN
      DO 30 J=1,NSTAGE
      N2J=N/(2**J)
      NR=N2J
      NT=(2**J)/2
      DO 20 I=1,NT
      IN2J=(I-1)*N2J
      FLTN2J=IN2J
      XSIGN=SIGN
      TEMP=FLTN2J*PHI2N*XSIGN
      W=CMPLX(COS(TEMP),SIN(TEMP))
      DO 20 R=1,NR
      ISUB=R+IN2J
      ISUB1=R+IN2J*2
      ISUB2=ISUB1+N2J
      ISUB3=ISUB+N2
      X(2,ISUB)=X(1,ISUB1) + W*X(1,ISUB2)
      X(2,ISUB3)=X(1,ISUB1) - W*X(1,ISUB2)
   20 CONTINUE
      DO 30 R=1,N
   30 X(1,R)=X(2,R)
      IF(SIGN.LT.0.) RETURN
      DO 40 R=1,N
   40 X(2,R)=X(1,R)/FLTN
      RETURN
      END
```

```
      SUBROUTINE FFT3(X,Y,M,N,IS)
C
C          X = REAL COMPONENTS OF INPUT DATA
C          Y = IMAGINARY COMPONENTS OF INPUT DATA
C          N = 2**M = NUMBER OF DATA POINTS
C         IS = -1, FORWARD TRANSFORM
C            = +1, INVERSE TRANSFORM
C          MUST CALL BRITS(X,Y,M,N) OR BLBTS(X,N)
C          AND BLBTS(Y,N) TO REORDER
C
      DIMENSION X(N),Y(N)
      DO 10 LG=1,M
      LMX=2**(M-LG)
      LIX=2*LMX
      SCL=6.283185/LIX
      DO 10 LM=1,LMX
      ARG=(LM-1)*SCL
      C=COS(ARG)
      S=-FLOAT(IS)*SIN(ARG)
      DO 10 LI=LIX,N,LIX
      J1=LI-LIX+LM
      J2=J1+LMX
      T1=X(J1)-X(J2)
      T2=Y(J1)-Y(J2)
      X(J1)=X(J1)+X(J2)
      Y(J1)=Y(J1)+Y(J2)
      X(J2)=C*T1+S*T2
   10 Y(J2)=C*T2-S*T1
C
      CALL BRITS(X,Y,M,N)
C
      RETURN
      END
```

```
      SUBROUTINE FFT4A(X,M,N,IS)
C
C     BASED ON DECIMATION-IN-FREQUENCY ALGORITHM
C
C     X = COMPLEX INPUT DATA ARRAY
C         (RESULT IS RETURNED IN X)
C     N = NUMBER OF DATA SAMPLES
C     M = ORDER OF SYSTEM (N=2**M)
C     IS = (DIRECTION OF TRANSFORM)
C        = -1, FORWARD TRANSFORM
C        = +1, INVERSE TRANSFORM
C
      COMPLEX X(1024),U,W,T
      PI=3.14159265
      DO 20 L=1,M
      LE=2**(M+1-L)
      LE1=LE/2
      U=(1.0,0.0)
      IF(IS)5,5,6
    5 W=CMPLX(COS(PI/FLOAT(LE1)),-SIN(PI/FLOAT(LE1)))
      GO TO 7
    6 W=CMPLX(COS(PI/FLOAT(LE1)),SIN(PI/FLOAT(LE1)))
    7 DO 20 J=1,LE1
      DO 10 I=J,N,LE
      IP=I+LE1
      T=X(I)+X(IP)
      X(IP)=(X(I)-X(IP))*U
   10 X(I)=T
   20 U=U*W
      NV2=N/2
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF(I.GE.J) GO TO 25
      T=X(J)
      X(J)=X(I)
      X(I)=T
   25 K=NV2
   26 IF(K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 26
   30 J=J+K
      RETURN
      END
```

```
      SUBROUTINE FFT43(X,M,N,IS)
C
C     DECIMATION IN FREQUENCY
C     THIS FFT SUBROUTINE IS A MODIFIED VERSION
C     OF FFT44.  IT ELIMINATES THE LAST SET OF
C     COMPLEX MULTIPLIES IN THE DECIMATION-IN-
C     FREQUENCY ALGORITHM.
C
C     X = COMPLEX INPUT DATA ARRAY
C         (RESULT IS RETURNED IN X)
C     N = NUMBER OF DATA SAMPLES
C     M = ORDER OF SYSTEM (N=2**M)
C     IS = (DIRECTION OF TRANSFORM)
C        = -1, FORWARD TRANSFORM
C        = +1, INVERSE TRANSFORM
C
      COMPLEX X(1024),U,W,T
      PI=3.14592653979
      MM1=M-1
      DO 20 L=1,MM1
      LE=2**(M+1-L)
      LE1=LE/2
      U=(1.0,0.0)
      IF(IS) 5,6,5
    5 W=CMPLX(COS(PI/FLOAT(LE1)),-SIN(PI/FLOAT(LE1)))
      GO TO 7
    6 W=CMPLX(COS(PI/FLOAT(LE1)),SIN(PI/FLOAT(LE1)))
    7 DO 20 J=1,LE1
      DO 10 I=J,N,LE
      IP=I+LE1
      T=X(I)+X(IP)
      X(IP)=(X(I)-X(IP))*U
   10 X(I)=T
   20 U=U*W
      DO 21 I=1,N,2
      IP=I+1
      T=X(I)+X(IP)
      X(IP)=X(I)-X(IP)
   21 X(I)=T
      NV2=N/2
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF(I.GE.J) GO TO 25
      T=X(J)
      X(J)=X(I)
      X(I)=T
   25 K=NV2
   26 IF(K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 26
   30 J=J+K
      RETURN
      END
```

```
      SUBROUTINE FFT5(SIGN,DTIME,X,NPOW,NMAX)
C
C     COOLEY-TUKEY METHOD OF FOURIER TRANSFORM
C     INCLUDES SINE COSINE COMPUTATION AND
C     REARRANGES DATA ACCORDING TO REVERSE BIT
C     ADDRESSES
C
C     SIGN = FOURIER DIRECTION TRANSFORM FLAG
C          = -1, FOR FORWARD TRANSFORM
C          = +1, FOR INVERSE TRANSFORM
C     DTIME = DELTA TIME
C     X     = LOC. OF FOURIER TRANSFORM BLOCK
C     NPOW = POWER OF 2 TO OBTAIN NMAX
C     NMAX = LENGTH OF BLOCK X
C
      DIMENSION X(NMAX),CS(2),MSK(13)
      COMPLEX X,CXCS,HOLD,XA
      EQUIVALENCE (CXCS,CS)
      ZZ=6.283185306*SIGN/FLOAT(NMAX)
      MSK(1)=NMAX/2
      DO 10 I=2,NPOW
   10 MSK(I)=MSK(I-1)/2
      NN=NMAX
      MM=2
C
C     LOOP OVER NPOW LAYERS
C
      DO 50 LAYER=1,NPOW
      NN=NN/2
      NW=0
      DO 40 I=1,MM,2
      TT=NN*I
C
C         CXCS = CEXP(2*PI*NW*SIGN/NMAX)
C
      W=FLOAT(NW)*ZZ
      CS(1)=COS(W)
      CS(2)=SIN(W)
C
C     COMPUTE ELEMENTS FOR BOTH HALVES OF EACH BLOCK
C
      DO 20 J=1,NN
      II=II+1
      IJ=II-NN
      XA=CXCS*X(II)
      X(II)=X(IJ)-XA
   20 X(IJ)=X(IJ)+XA
```

```
C
C       PUMP UP SERIES BY 2
C
C       COMPUTE REVERSE ADDRESS
C
        DO 30 LOC=2,NPOW
        LL=NW-MSK(LOC)
        IF(LL) 32,34,30
   30   NW=LL
   32   NW=MSK(LOC)+NW
        GO TO 40
   34   NW=MSK(LOC+1)
   40   CONTINUE
   50   MM=MM*2
C
C       DO FINAL REARRANGEMENT
C       ALSO MULTIPLY BY DELTA TIME
C
        NW=0
        DO 90 I=1,NMAX
        NW1=NW+1
        HOLD=X(NW1)
        IF(NW1-I)55,52,60
   50   X(NW1)=X(I)*DTIME
   62   X(I)=HOLD*DTIME
C
C       PUMP UP SERIES BY 1
C       COMPUTE REVERSE ADDRESS
C
   55   DO 70 LOC=1,NPOW
        LL=NW-MSK(LOC)
        IF(LL) 72,74,70
   70   NW=LL
   72   NW=MSK(LOC)+NW
        GO TO 90
   74   NW=MSK(LOC+1)
   90   CONTINUE
        IF(SIGN)100,100,91
   91   PTS=NMAX
        DO 95 I=1,NMAX
   95   X(I)=X(I)/PTS
  100   RETURN
        END
```

```
      SUBROUTINE FFT6(L,X,Y)
      DIMENSION X(1), Y(1), IS(13), IU(13)
      EQUIVALENCE (I2S,IS(2)), (I3S,IS(3)), (I4S,IS(4)),
     *            (I5S,IS(5)), (I6S,IS(6)), (I7S,IS(7)),
     *            (I8S,IS(8)), (I9S,IS(9)), (I10S,IS(10)),
     *            (I11S,IS(11)), (I12S,IS(12)), (I13S,IS(13))
      EQUIVALENCE (I1L,IU(1)), (I2L,IU(2)), (I3L,IU(3)),
     *            (I4L,IU(4)), (I5L,IU(5)), (I6L,IU(6)),
     *            (I7L,IU(7)), (I8L,IU(8)), (I9L,IU(9)),
     *            (I10L,IU(10)), (I11L,IU(11)), (I12L,IU(12)),
     *            (I13L,IU(13))
      DATA TWOPI/6.283185/
C
C
C          FFT :  FAST FOURIER TRANSFORM
C
C              1 DIMENSIONAL, MIXED RADIX
C                 X = REAL COMPONENTS
C                 Y = IMAGINARY COMPONENTS
C                 2 ** L = NO. OF ELEMENTS
C              L < 0 COMPUTES INVERSE TRANSFORM
C              NO SCALING IS DONE
C              EQUIVALENCED VARIABLES ARE USED FOR SORTING
C              APPROXIMATE EXECUTION TIME
C                 TIME = C * L * 2**L   (SECONDS)
C                        C = 3.4E-5 FOR L > 0
C                        C = 3.7E-5 FOR L < 0
C
C
      L2N=IABS(L)
      N=2**L2N
      IF(L) 6,5,8
    6 DO 7 J=1,N
    7 Y(J)=-Y(J)
C
    8 IF (L2N-1) 15,15,9
    9 M=N
      DO 12 K=2,L2N,2
      M4=M
      M=M4/4
      F=TWOPI/M4
      DO 12 J=1,M
      THETA=F*(J-1.)
      C1=COS(THETA)
      S1=SIN(THETA)
      C2=C1*C1-S1*S1
      S2=2.*C1*S1
      C3=C2*C1-S2*S1
      S3=C2*S1+S2*C1
      JM4=J-M4
      DO 12 I=M4,N,M4
      J1=I+JM4
      J2=J1+M
      J3=J2+M
      J4=J3+M
```

```
      P1=X(J1)+X(J3)
      P2=X(J1)-X(J3)
      P3=Y(J1)+Y(J3)
      P4=Y(J1)-Y(J3)
      P5=X(J2)+X(J4)
      P6=X(J2)-X(J4)
      P7=Y(J2)+Y(J4)
      P8=Y(J2)-Y(J4)
      Y(J1)=P1+P5
      Y(J1)=P3+P7
      IF(J-1)10,11,10
   10 TMP1=S2+S8
      TMP2=S4-S6
      X(I3)=TMP1*S1+TMP2*S1
      Y(I3)=TMP2*S1-TMP1*S1
      TMP1=P1-P5
      TMP2=P3-P7
      X(J2)=TMP1*S2+TMP2*S2
      Y(I2)=TMP2*S2-TMP1*S2
      TMP1=S2-S8
      TMP2=S4+S6
      X(J4)=TMP1*S3+TMP2*S3
      Y(J4)=TMP2*S3-TMP1*S3
      GO TO 12
   11 Y(I3)=P2+P8
      Y(I3)=P4-P6
      X(I2)=P1-P5
      Y(I2)=P3-P7
      X(J4)=P2-P8
      Y(J4)=P4+P6
   12 CONTINUE
C
   15 IF(L2N-2*(L2N/2)) 16,16,16
   16 DO 17 I=1,N,2
      P1=X(I)+X(I+1)
      P2=X(I)-X(I+1)
      P3=Y(I)+Y(I+1)
      P4=Y(I)-Y(I+1)
      X(I)=P1
      Y(I)=P3
      X(I+1)=P2
   17 Y(I+1)=P4
   18 TMG=N/2
      TML=N
      J=12
      DO 20 JJ=1,11
      IS(J)=1
      IF(IS(J+1)-1.LE.0) GO TO 19
      IS(J)=IS(J+1)/2
   19 IU(J)=IS(J+1)
      J=J-1
   20 CONTINUE
```

```
C
      IAL=IRS
      JT=0
      DO 22 IA=1,IAL
      DO 22 IB=IA,IBL,IBS
      DO 22 IC=IB,ICL,ICS
      DO 22 ID=IC,IDL,IDS
      DO 22 IE=ID,IEL,IES
      DO 22 IEI=IE,IEL,IES
      DO 22 IG=IEI,IGL,IGS
      DO 22 IH=IG,IHL,IHS
      DO 22 II=IH,IIL,IIS
      DO 22 IJ=II,IJL,IJS
      DO 22 IK=IJ,IKL,IKS
      DO 22 IL=IK,ILL,ILS
      DO 22 IM=IL,IML,IMS
      JT=JT+1
      IF(JT-IM) 22,22,21
   21 T=Y(JT)
      Y(JT)=X(IM)
      X(IM)=T
      T=Y(JT)
      Y(JT)=Y(IM)
      Y(IM)=T
   22 CONTINUE
C
      IF(L) 35,35,37
   35 DO 36 J=1,N
   36 Y(J)=-Y(J)
   37 RETURN
      END
```

## Appendix D

### FWT/FHT Subroutines

The fast Walsh/Hadamard transforms listed in this appendix are internally documented. Both the input and transformed output arrays are real and in sequency order. The call statement must list all of the parameters required by the subroutine parameter list.

```
      SUBROUTINE FHT1(F,N,M,IX)
C
C        FHT1 FINDS THE HADAMARD TRANSFORM OR
C        INVERSE TRANSFORM OF AN INPUT VECTOR
C        (A SAMPLED REAL SIGNAL)
C
C     DESCRIPTION OF VARIABLES
C        F - INPUT VECTOR OF LENGTH N
C        N - NUMBER OF DATA
C        M - LOG BASE 2 OF N:
C            2**M = NO. OF COEFF. PER TIME INTERVAL
C       IX - FOR FORWARD TRANSFORM:  X = 1
C          - FOR INVERSE TRANSFORM:  X = J
C
C
      DIMENSION F(N)
      L=N
      K=1
      IF(IX.LT.1) GO TO 40
      DO 30 NM=1,M
      I=1
      L=L/2
      DO 20 NL=1,L
      DO 10 NK=1,K
      I=I+1
      J=I+K
      F(I)=(F(I)+F(J))/2.
   10 F(J)=F(I)-F(J)
   20 I=I+1
   30 K=K*2
      GO TO 80
   40 DO 70 NM=1,M
      I=1
      L=L/2
      DO 60 NL=1,L
      DO 50 NK=1,K
      I=I+1
      J=I+K
      F(I)=F(I)+F(J)
   50 F(J)=F(I)-F(J)-F(J)
   60 I=I+1
   70 K=K*2
   80 RETURN
      END
```

```
      SUBROUTINE FHT2(X,N,M,KOPT,NUM)
C
C       ONE DIMENSIONAL HADAMARD TRANSFORM
C
C       KOPT = 1 : SCRAMBLE AND INVERT
C            = 2 : TO TRANSFORM WITHOUT UNSCRAMBLING
C                  OR TO INVERT WITHOUT SCRAMBLING
C            = 3 : TRANSFORM AND UNSCRAMBLE
C        NUM   : NORMALIZED OR UNNORMALIZED OUTPUT
C              = 1 : NORMALIZED   (FORWARD)
C              = 0 : UNNORMALIZED (INVERSE)
C
C       VARIABLES
C         X - INPUT DATA ARRAY
C         M - NUMBER OF BITS
C         N - NUMBER OF DATA POINTS : N=2**M
C
C
      DIMENSION X(N)
      IF(KOPT.NE.1) GO TO 31
      MBIT=2
      KSIZE=4
      DO 1 KBIT=2,M
      KHALF=KSIZE/2
      KHALF1=KHALF+1
      DO 2 K=KHALF1,KSIZE,2
      DO 2 K1=K,N,KSIZE
      DUM=X(K1)
      X(K1)=X(K1+1)
      X(K1+1)=DUM
    2 CONTINUE
      JUMP=KSIZE/4
      KSPAN=KSIZE
      DO 11 NBIT=2,MBIT
      KST1=JUMP+1
      DO 12 KST=KST1,KSIZE,KSPAN
      KSINMX=KST+JUMP-1
      DO 12 KSIN=KST,KSINMX
      DO 12 KSING=KSIN,N,KSIZE
      DUM=X(KSING)
      X(KSING)=X(KSING+JUMP)
      X(KSING+JUMP)=DUM
   12 CONTINUE
      JUMP=JUMP/2
      KSPAN=KSPAN/2
   11 CONTINUE
      KSIZE=KSIZE+KSIZE
      MBIT=MBIT+1
    1 CONTINUE
   31 CONTINUE
      NSPAN=N
      NDIST=NSPAN/2
      DO 20 I=1,M
```

```
      DO 21 L=1,NDIST
      JM=L+N-NSPAN
      DO 21 J=L,JM,NSPAN
      DUM=X(J)
      X(J)=DUM+X(J+NDIST)
      X(J+NDIST)=DUM-X(J+NDIST)
   21 CONTINUE
      NSPAN=NDIST
      NDIST=NSPAN/2
   20 CONTINUE
      DO 22 I=1,N
      X(I)=X(I)/(FLOAT(N)**NIM1)
   22 CONTINUE
      IF(KOPT.NE.3) RETURN
      KSIZE=N
      MBIT=M
      DO 3 KBIT=2,M
      KHALF=KSIZE/2
      KHALF1=KHALF+1
      JUMP=1
      KSPAN=4
      DO 13 NBIT=2,MBIT
      KST1=JUMP+1
      DO 14 KST=KST1,KSIZE,KSPAN
      KSTNMX=KST+JUMP-1
      DO 14 KSIN=KST,KSTNMX
      DO 14 KSING=KSIN,N,KSIZE
      DUM=X(KSING)
      X(KSING)=X(KSING+JUMP)
      X(KSING+JUMP)=DUM
   14 CONTINUE
      JUMP=JUMP+JUMP
      KSPAN=KSPAN+KSPAN
   13 CONTINUE
      DO 4 K=KHALF1,KSIZE,2
      DO 4 K1=K,N,KSIZE
      DUM=X(K1)
      X(K1)=X(K1+1)
      X(K1+1)=DUM
    4 CONTINUE
      KSIZE=KHALF
      MBIT=MBIT-1
    3 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FWT1(F,V,N,IX)
C
C       THIS SUBROUTINE FINDS THE WALSH TRANSFORM
C       IN THE FOLLOWING WAY
C       1.  GENERATES A SET OF WALSH FUNCTIONS
C       2.  ARRANGES THEM IN A HADAMARD MATRIX
C       3.  USES IT TO FIND THE FORWARD OR
C           INVERSE WALSH TRANSFORM OF AN INPUT VECTOR
C        NOTE - MAXIMUM SIZE IS 128
C      DEFINITION OF VARIABLES
C        F - INPUT VECTOR
C        V - OUTPUT VECTOR
C        N - NUMBER OF DATA SAMPLES
C        M - ORDER OF SYSTEM: N=2**M
C       IX - FORWARD TRANSFORM:  IX = 1
C          - INVERSE TRANSFORM:  IX = 0
C
      DIMENSION F(N),V(N)
      COMMON WAL(128,128)
      INTEGER WAL
C
      CALL WALSH(N)
C
      IF(IX.EQ.0) GO TO 100
C
      DO 30 I=1,N
      V(I)=0.
      DO 25 J=1,N
      A=WAL(I,J)
   25 V(I)=(F(J)*A)+V(I)
   30 V(I)=V(I)/FLOAT(N)
      GO TO 110
C
C      INVERSE TRANSFORM
  100 DO 70 I=1,N
      F(I)=0.
      DO 70 J=1,N
      A=WAL(I,J)
   70 F(I)=(V(I)*A)+F(I)
  110 RETURN
      END
```

```
      SUBROUTINE WALSH(N)
C
C     WALSH GENERATES A SET OF WALSH FUNCTIONS
C     USING THE METHOD DESCRIBED BY PETERSON
C
      COMMON WAL(128,128)
      INTEGER WAL,R(7)
      LN=INT((ALOG(FLOAT(N))/ALOG(2.))+.1)
      DO 50 I=1,N
      K=I-1
      J=1
    5 J1=J
      J=J+1
      IF(MOD(K,2)) 20,20,10
   10 K=K-1
      R(J)=-1
      GO TO 30
   20 R(J)=1
   30 K=K/2
      IF(J.LE.1) GO TO 5
      R(J1)=R(J)*R(J1)
      IF(LN-J) 40,40,5
   40 WAL(I,1)=1
      DO 50 J=1,LN
      JY=LN-J+1
      K=2**(J-1)
      DO 50 JX=1,K
      JY=JX+K
   50 WAL(I,JY)=WAL(I,JX)*R(JY)
      RETURN
      END
```

```
      SUBROUTINE FASWAL(F,X,N,LN,IX,T,B)
C
C        FASWAL FINDS THE WALSH TRANSFORM OR INVERSE TRANSFORM
C        OF AN INPUT VECTOR USING A 'FAST TRANSFORM' ALGORITHM
C
C        DESCRIPTION OF VARIABLES
C            F - INPUT VECTOR OF LENGTH N
C            X - OUTPUT VECTOR OF LENGTH N
C            N - ORDER OF SYSTEM
C           LN - LOG BASE 2 OF N
C            T - INTERNAL WORKING VECTOR OF LENGTH LN
C            B - INTERNAL WORKING VECTOR OF LENGTH LN
C           IX - FOR TRANSFORM:  IX = 1
C              - FOR INVERSE TRANSFORM, IX = 0
C
C         FASWAL REQUIRES INTEGER FUNCTION SUBROUTINE CONVERT
C
C
      DIMENSION F(N), X(N), T(N)
      INTEGER IX,B(LN)
      INTEGER CONVRT
      N1=N/2
      DO 10 I=1,N
   10 T(I)=F(I)
      DO 30 J=1,LN
      DO 20 K=1,N1
      K1=2*K
      X(K)=T(K1-1)+T(K1)
      K2=K+N1
   20 X(K2)=T(K1-1)-T(K1)
      DO 30 I=1,N
   30 T(I)=X(I)
C
C        T(I) NOW CONTAINS RESULT IN REVERSE REFLECTED BINARY ORDER
C
      NIX=N**IX
      DO 40 J=1,N
      JX=J-1
   40 X(J)=T(CONVRT(JX,LN,2)+1)/NIX
      RETURN
      END
```

```
      SUBROUTINE FFTI1(IX,IY,M,N,ISIGN)
C
C     IX = INTEGER REAL PART OF THE INPUT SEQUENCE
C     IY = INTEGER IMAGINARY PART OF THE INPUT SEQUENCE
C     M = ORDER OF THE SYSTEM (N=2**M)
C     N = NUMBER OF INPUT SAMPLES
C     ISIGN = -1, FORWARD TRANSFORM
C           = +1, INVERSE TRANSFORM
C
      INTEGER FACTOR,IX(N),IY(N),T1,T2
      FACTOR=2**15-1
C
      DO 10 LO=1,M
      LMX=2**(M-LO)
      LIX=2*LMX
      SCL=6.283185/LIX
      DO 10 LM=1,LMX
      ARG=(LM-1)*SCL
      IC=COS(ARG)*FACTOR
      IS=-FLOAT(ISIGN)*SIN(ARG)*FACTOR
      DO 10 LI=LIX,N,LIX
      J1=LI-LIX+LM
      J2=J1+LMX
      T1=IX(J1)-IX(J2)
      T2=IY(J1)-IY(J2)
      IX(J1)=IX(J1)+IX(J2)
      IY(J1)=IY(J1)+IY(J2)
      IX(J2)=SHIFT(IC*T1+IS*T2,-15)
      IY(J2)=SHIFT(IC*T2-IS*T1,-15)
   10 CONTINUE
C
      CALL UNSCR1(IX,IY,N)
C
      RETURN
      END
```

## Appendix E

### Fixed-Point FFT Subroutines

The fixed-point FFT subroutines are called by Subroutine CIMAGE (Version 3). The complex input data are integer scaled values placed into two separate arrays declared in the calling program. All the subroutine listings contain documentation defining the variables employed.

FFTI4 uses the values generated by Subroutine COSINE to perform the butterfly computations. Subroutine COSINE is called once in the calling program and the trigunometric values are passed as an array in the call statement parameter list.

Subroutine FFTI5 uses the values generated by Subroutine COSINE. It is called once in the calling program and the array passed through the parameter list in the call statement.

Subroutines FFTI1, FFTI4, and FFTI5 call either Subroutine UNSCR or Subroutine UNSCR1 (Appendix F) tc reorder the scrambled integer Fourier coefficients.

```
*
      SUBROUTINE FFTI3(IX,IY,M,N,ISIGN)
C
C     THIS INTEGER FFT IS BASED ON FFT4.
C
C     THE VARIABLES USED IN THIS SUBROUTINE
C     ARE DEFINED THE SAME AS IN FFTI1.
C
      INTEGER IX(N),IY(N),T1,T2
      DATA PI,FACTOR/3.14159265,32767./
      DO 10 L=1,M
      LE=2**(M+1-L)
      LE1=LE/2
      DO 10 J=1,LE1
      IC=COS((J-1)*PI/LE1)*FACTOR
      IS=-FLOAT(ISIGN)*SIN((J-1)*PI/LE1)*FACTOR
      DO 10 I=J,N,LE
      IP=I+LE1
      T1=IX(I)-IX(IP)
      T2=IY(I)-IY(IP)
      IX(I)=IX(I)+IX(IP)
      IY(I)=IY(I)+IY(IP)
      IX(IP)=SHIFT(IC*T1+IS*T2,-15)
      IY(IP)=SHIFT(IC*T2-IS*T1,-15)
   10 CONTINUE
      NV2=N/2
      NM1=N-1
      J=1
      DO 20 I=1,NM1
      IF(I.GE.J)GO TO 25
      T1=IX(J)
      T2=IY(J)
      IX(J)=IX(I)
      IY(J)=IY(I)
      IX(I)=T1
      IY(I)=T2
   25 K=NV2
   26 IF(K.GE.J)GO TO 20
      J=J-K
      K=K/2
      GO TO 26
   20 J=J+K
      RETURN
      END
```

```
      SUBROUTINE FFTI2(IX,IY,M,N,ISIGN)
      INTEGER IX(N),IY(N),T1,T2
C
C     THIS INTEGER FFT IS BASED ON FFT1.
C
C     THE VARIABLES USED IN THIS SUBROUTINE
C     ARE DEFINED THE SAME AS IN FFTI1.
C
      DATA PI/3.14159265/
      FACTOR=2**15-1
      NV2=N/2
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF(I.GE.J)GO TO 10
      T1=IX(J)
      IX(J)=IX(I)
      IX(I)=T1
      T2=IY(J)
      IY(J)=IY(I)
      IY(I)=T2
   10 K=NV2
   20 IF(K.GE.J)GO TO 30
      J=J-K
      K=K/2
      GO TO 20
   30 J=J+K
      DO 80 L=1,M
      LE=2**L
      LE1=LE/2
      IC=COS(PI/FLOAT(LE1))*FACTOR
      IS=-FLOAT(ISIGN)*SIN(PI/FLOAT(LE1))*FACTOR
      DO 80 J=1,LE1
      DO 80 I=J,N,LE
      IP=I+LE1
      IX(I)=SHIFT(IX(I)-IX(IP)*IC+IY(IP)*IS,-15)
      IY(I)=SHIFT(IY(I)-IX(IP)*IS+IY(IP)*IC,-15)
      IX(IP)=SHIFT(IX(I)-IX(IP)*IC-IY(IP)*IC,-15)
      IY(IP)=SHIFT(IY(I)+IX(IP)*IS-IY(IP)*IC,-15)
   80 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE COSINE(IC,N)
C
C     THIS SUBROUTINE GENERATES A COSINE
C     TABLE FROM 0. TO PI/2.
C
      DIMENSION IC(N)
      DATA TWOPI/6.28318530/
      N4=SHIFT(N,-2)
      IC(N4+1)=0
      DO 10 I=1,N4
      THETA=(I-1)*TWOPI/FLOAT(N)
   10 IC(I)=COS(THETA)*32767.
      RETURN
      END
```

```
*
      SUBROUTINE FFT1B(IX,IY,IC,M,N)
C
      INTEGER IX(N),IY(N),IC(N),T1,T2,CSE(2)
C
C     VARIABLES DEFINED THE SAME AS IN FFT1A
C
      NN2=SHIFT(N,-1)
      NN4=SHIFT(N,-2)
      MM1=M-1
      DO 10 L=1,MM1
      LE=2**(M+1-L)
      LA=2**(L-1)
      LE1=SHIFT(LE,-1)
      DO 10 J=1,LE1
      LUA=(J-1)*LA+1
      CSE(1)=IC(LUA)
      CSE(2)=-IC(NN4+LUA)
      DO 10 I=J,N,LE
      IP=I+LE1
      T1=IX(I)-IX(IP)
      T2=IY(I)-IY(IP)
      IX(I)=IX(I)+IX(IP)
      IY(I)=IY(I)+IY(IP)
      IX(IP)=SHIFT(CSE(1)*T1+CSE(2)*T2,-15)
   10 IY(IP)=SHIFT(CSE(1)*T2-CSE(2)*T1,-15)
      DO 30 I=1,N,2
      IP=I+1
      T1=IX(I)+IX(IP)
      T2=IY(I)+IY(IP)
      IX(IP)=IX(I)-IX(IP)
      IY(IP)=IY(I)-IY(IP)
      IX(I)=T1
   30 IY(I)=T2
      CALL UNSCR1(IX,IY,N)
      RETURN
      END
*
      SUBROUTINE COSINE(IC,N)
C
C     THIS SUBROUTINE GENERATES A COSINE
C     TABLE FROM 0 TO TWO PI.
C
      DIMENSION IC(N)
      DATA TWOPI/6.283185307/
      DO 10 I=1,N
      THETA=(I-1)*TWOPI/N
   10 IC(I)=COS(THETA)*32767.
      RETURN
      END
```

## Appendix F

### Reordering Subroutines

The reordering subroutin ; listed in this appendix unscramble bit-reversed or scramble into bit-reversed order the output sequence or the input sequence, respectively. The exchange operations are performed in-place.

Subroutine RBITS is written to perform bit reversal of real arrays. It must be called twice-once for the real component and once for the imaginary component. It can easily be modified to handle complex FORTRAN arrays.

Subroutines UNSCR and UNSCR1 are written to unscramble integer Fourier coefficients. Subroutine UNSCR1 is modified from Subroutine RBITS. Both subroutines can easily be modified to unscramble real, complex, or two separate real arrays depending on the need.

```
      SUBROUTINE BITS(X,Y,N,M)
C
C     PERFORMS IN-PLACE BIT REVERSAL FOR N=2**M VALUES OF X(I)
C     WHERE M IS LESS THAN OR EQUAL TO 10
C     OUTPUT SEQUENCE IS Y(I)
C
      DIMENSION X(N),Y(N),L(10)
      EQUIVALENCE (L10,L(1)),(L9,L(2)),(L8,L(3)),(L7,L(4)),
     *            (L6,L(5)),(L5,L(6)),(L4,L(7)),(L3,L(8)),
     *            (L2,L(9)),(L1,L(10))
      DO 20 J=1,10
      L(J)=1
      IF(J-M) 10,10,20
   10 L(J)=2**(M+1-J)
   20 CONTINUE
      JN=1
      DO 50 J1=1,L1
      DO 50 J2=J1,L2,L1
      DO 50 J3=J2,L3,L2
      DO 50 J4=J3,L4,L3
      DO 50 J5=J4,L5,L4
      DO 50 J6=J5,L6,L5
      DO 50 J7=J6,L7,L6
      DO 50 J8=J7,L8,L7
      DO 50 J9=J8,L9,L8
      DO 50 JR=J9,L10,L9
      IF(JN-JR) 30,30,40
   30 R=Y(JN)
      Y(JN)=X(JR)
      X(JR)=R
      F1=Y(JN)
      Y(JN)=Y(JR)
      Y(JR)=F1
   40 JN=JN+1
   50 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE UNSCR(IX,IY,N)
      INTEGER IX(N),IY(N)
C
C     PERFORMS BIT-REVERSAL FOR INTEGER FFT
C
      NM2=SHIFT(N,-1)
      NM1=N-1
      J=1
      DO 20 I=1,NM1
      IF(I.GE.J) GO TO 25
      T1=IX(J)
      T2=IY(J)
      IX(J)=IX(I)
      IY(J)=IY(I)
      IX(I)=T1
      IY(I)=T2
   25 K=NM2
   26 IF(K.GE.J) GO TO 20
      J=J-K
      K=SHIFT(K,-1)
      GO TO 26
   20 J=J+K
      RETURN
      END
```

```
      SUBROUTINE UNSCR1(IX,IY,N)
      DIMENSION IX(N),IY(N),L(10)
      EQUIVALENCE (L10,L(1)),(L9,L(2)),(L8,L(3)),(L7,L(4)),
     *            (L6,L(5)),(L5,L(6)),(L4,L(7)),(L3,L(8)),
     *            (L2,L(9)),(L1,L(10))
C
C     PERFORM IN-PLACE BIT REVERSAL FOR
C     INTEGER FFT SUBROUTINES
C
      M=ALOG(FLOAT(N))/ALOG(2.)+.1
      DO 20 J=1,10
      L(J)=1
      IF(J-M) 30,30,20
   30 L(J)=2**(M+1-J)
   20 CONTINUE
      JN=1
      DO 50 J1=1,L1
      DO 50 J2=J1,L2,L1
      DO 50 J3=J2,L3,L2
      DO 50 J4=J3,L4,L3
      DO 50 J5=J4,L5,L4
      DO 50 J6=J5,L6,L5
      DO 50 J7=J6,L7,L6
      DO 50 J8=J7,L8,L7
      DO 50 J9=J8,L9,L8
      DO 50 JR=J9,L10,L9
      IF(JN-JR)35,35,40
   35 IR=IX(JN)
      IX(JN)=IX(JR)
      IX(JR)=IR
      IF=IY(JN)
      IY(JN)=IY(JR)
      IY(JR)=IF
   40 JN=JN+1
   50 CONTINUE
      RETURN
      END
```

## Appendix G

### Display Subroutines

Subroutine PLOTID creates a CALCOMP display of the radar image processed by Subroutine CIMAGE. It reads data from the temporary image file (NIMG) and compares each value with a threshold and prints a symbol if the value exceeds the threshold.

Subroutine BUFOUT creates a line printer display of the radar image generated by Subroutine CIMAGE. The procedure is the same as in Subroutine PLOTID, except the display is printed on a hard copy by an on-line printer.

```
      SUBROUTINE PLOTTO(NIMG,ND3,NTM,NBSZ)
      REAL VOLT(256)
C
C     PLOTTO CREATES A CALCOMP
C     DISPLAY OF RADAR IMAGE
C
      DATA KPLOT/30/
      N=ND3*NTM
   20 READ(5,*) THRES
      IF(EOF(5).NE.0)GO TO 30
C
C-----CONSTRUCT THE BORDER
C
      CALL PLOTS(KPLOT,1,4LPLOT)
      CALL PLOT(0.,-3.,-3)
      CALL PLOT(0.,1.5,-3)
      CALL PLOT(0.,8.,2)
      CALL PLOT(5.3,8.,2)
      CALL PLOT(5.3,0.,2)
      CALL PLOT(0.,0.,2)
      CALL PLOT(0.15,7.5,-3)
C
C-----PLOT DATA
C
      DX=5./NBSZ
      DY=-7./N
      DO 10 J=1,N
      READ(NIMG) (VOLT(I), I=1,NBSZ)
      Y=FLOAT(J-1)*DY
      DO 10 I=1,NBSZ
      IF(VOLT(I).LE.THRES)GO TO 10
      X=FLOAT(I-1)*DX
      CALL SYMBOL(X,Y,0.04,11,0.,-1)
   10 CONTINUE
C
C-----RESET ORIGIN FOR NEXT PLOT
C
      CALL PLOT(7.5,-7.5,-3)
      REWIND NIMG
      GO TO 20
   30 CALL PLOTF(N)
      RETURN
      END
```

```
      SUBROUTINE BUFOUT(NIMG,ND3,NTM,NBSZ)
      DIMENSION BORDER(130)
      REAL VOLT(256)
C
C     BUFOUT CREATES A PAGE PRINTER
C     DISPLAY OF RADAR IMAGE
C
      DATA PLUS/1H+/, DOT/1H*/, BLANK/1H /
   40 READ(5,*) THRES
      IF(EOF(5).NE.0) GO TO 50
C
C     CONSTRUCT BORDER MATRIX
C
      DO 10 I=1,130
      BORDER(I)=PLUS
   10 CONTINUE
      WRITE(7,12)
   12 FORMAT(///,T53,"IMAGE PLOT",//)
      WRITE(7,35) THRES
   35 FORMAT(///,1X,"THRESHOLD VALUE IS ",F10.1,///)
C
C     TOP BORDER
C
      WRITE(7,15) BORDER
C
C     CONSTRUCT THE PLOT
C
      N=ND3*NTM
      DO 20 J=1,N
      READ(NIMG) (VOLT(I), I=1,NBSZ)
      DO 30 I=1,NBSZ
      IF(VOLT(I).GE.THRES)GO TO 21
      VOLT(I)=BLANK
      GO TO 30
   21 VOLT(I)=DOT
   30 CONTINUE
      WRITE(7,25) BORDER(1),(VOLT(I),I=1,NBSZ),BORDER(130)
   25 FORMAT(1X,1A1,128A1,1A1)
   20 CONTINUE
C
C     BOTTOM BORDER
C
      WRITE(7,15) BORDER
   15 FORMAT(1X,130A1)
      REWIND NIMG
      GO TO 40
   50 RETURN
      END
```

## Appendix H

### Radar/Scatterer Parameters

The radar/scatterer parameters used in this thesis are listed in this appendix. Data Set 1 is listed on page and Data Set 2 is listed on the following page.

They are in the Format specified by Program TGTID.

## IMAGE PARAMETERS

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| RADAR PARAMETERS | = | 500. | 0. | 0. | .050000 | .990000E+10 | .200000E+07 | |
| ANGLE DATA | = | -.04 | 0.00 | 0.00 | .02 | 0.00 | 0.00 | |
| NO. OF BURSTS | = | 70 | | | | | | |
| NO. OF FREQUENCIES | = | 100 | | | | | | |
| NO. OF SCATTERERS | = | 4 | | | | | | |
| SCATTERER PARAMETERS | = | 0.000 | 3.000 | 0.000 | 5.000 | .500 | -.866 | 0.000 |
| | | 5.000 | 0.000 | 0.000 | 2.000 | 1.000 | 0.000 | 0.000 |
| | | 0.000 | 0.000 | 0.000 | 2.000 | 1.000 | 0.000 | 0.000 |
| | | 0.000 | -1.000 | 0.000 | 3.000 | .500 | .866 | 0.000 |

MAXIMUM POWER =
NO. OF WORDS/RECORD =
NO. OF RECORDS =

Data Set 1.

## IMAGE PARAMETERS

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| RADAR PARAMETERS | = | 500. | 0. | 0. | .050000 | .990000E+10 | .200000E+07 |
| ANGLE DATA | = | -.04 | 0.00 | 0.00 | .02 | 0.00 | 0.00 |
| NO. OF BURSTS | = | 70 | | | | | |
| NO. OF FREQUENCIES | = | 100 | | | | | |
| NO. OF SCATTERERS | = | 4 | | | | | |
| SCATTERER PARAMETERS | = | -10.000 | -5.000 | 0.000 | 3.000 | .500 | .866 | 0.000 |
| | | -5.000 | -3.000 | 0.000 | 2.000 | 1.000 | 0.000 | 0.000 |
| | | -10.000 | 0.000 | 0.000 | 3.000 | .500 | -.866 | 0.000 |
| | | -10.000 | -3.000 | 0.000 | 2.000 | 1.000 | 0.000 | 0.000 |

MAXIMUM POWER =
NO. OF WORDS/RECORD =
NO. OF RECORDS =

Data Set 2.

## VITA

Robert L. Herron was born on 19 December 1947 in Amsterdam, New York, the son of Mr. and Mrs. Louis C. Herron. His secondary schooling was taken at Canajoharie High School, Canajoharie, New York. He received the degree of Bachelor of Science in Electrical Engineering from Union College, Schenectady, New York in June 1970. He received a commission in that same month through AFROTC. He was initially assigned to Keesler Air Force Base, Mississippi, where he attended the Basic Communications-Electronics Officer Course. His first assignment was OIC Record Communications in the 2017 Communications Squadron, McGuire AFB, New Jersey. In January 1972, he was reassigned to the Second Mobile Communications Group, Sembach AB, Germany and later Lindsey AS, Wiesbaden, Germany, as OIC Deployment Planning. He was selected to attend the Air Force Institute of Technology, School of Engineering, in June 1974. He received the degree of Master of Science in Electrical Engineering in December 1977.

Permanent Address: R.D.#1
Canajoharie, New York 13317

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

| REPORT DOCUMENTATION PAGE | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|
| 1. REPORT NUMBER: AFIT/GE/EE/77-20 | 2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE *(and Subtitle)*: COMPARISON OF FAST FOURIER TRANSFORMS WITH OTHER TRANSFORMS IN SIGNAL PROCESSING FOR TACTICAL RADAR TARGET IDENTIFICATION | 5. TYPE OF REPORT & PERIOD COVERED: ~~MS Thesis~~ / 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s): Robert L. Herron, ~~Captain~~ USAF | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT/EN), Wright-Patterson AFB Ohio 45433 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (OCTM), Griffiss AFB New York 13441 | 12. REPORT DATE: December 1977 / 13. NUMBER OF PAGES: 165 |
| 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: Approved for public release; IAW AFR 190-17.

JERRAL F. GUESS, Captain, USAF
Director of Information

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

| | | | |
|---|---|---|---|
| Radar | Mathematics | Fourier transformations | Signals |
| Doppler radar | Walsh Functions | Frequency Shift | Images |
| Radar Signals | Fourier Series | Doppler Effect | Processing |
| Radar reflection | Transformations | Signatures | Spectra |
| Radar Cross Sections | Integral transforms | Radar Signatures | Radar images |

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The High Resolution Radar Branch of the Rome Air Development Center has developed a tactical target identification (TTI) pulsed-Doppler radar system which generates two-dimensional 'images' of aircraft. The signal processing technique utilizes the fast Fourier transform (FFT) to produce a slant-range versus cross-range display. If the TTI system is to be effectively employed in an aerial warfare environment then real-time processing is necessary. In an effort to speedup the signal processing several alternative transforms were studied as possible substitutes for the FFT. The Karhunen-Loeve, Cosine (Sine), Mellin, and Hankel

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

19. Spectrum signatures
Radar Pulses
Signal processing
Power Spectra

transforms were investigated and found to be infeasible for use in TTI imaging. The Walsh (Hadamard) transform was studied in detail and tested in a simulation program and found that it could not be utilized in the TTI signal processing.

Two methods of converting from the Walsh sequency domain to the Fourier frequency domain were studied. The first scheme, a recursive relationship between the arithmetic and logical autocorrelation functions as presented by Robinson was discovered to be incorrect. The second, a method of computing the Fourier coefficients from the Walsh coefficients of a function was demonstrated to be too time consuming to be implemented in TTI signal processing.

Several floating-point FFT implementations were tested using the simulation program. Also, several fixed-point FFT algorithms were derived and tested. All of these were evaluated on the basis of speed and memory requirements and one fixed-point FFT algorithm was shown to be fast enough and accurate enough for implementation on the TTI Minicomputer.