SPECTROGRAM GENERATION WITH A MINICOMPUTER AND A GRAPHICS TERMINAL

by

RONALD DALE SAUDER

B. S., Kansas State University, 1975

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1977

Approved by:

E E Hoft

Major Professor

Table of Contents

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

Table of Figures

Acknowledgments

Several persons have been of assistance to me in the development of this report. I would like to thank Dr. E. Haft for helping me get started on the project and offering assistance. Several persons deserve thanks for answering my questions about the NOVA, particularly Joe Fogler. I also appreciate David Hein's advice on using his routines to read from the disk and Dr. N. Ahmed's suggestions on details about spectrogram generation. Finally, a special thanks should go to those who helped maintain the NOVA, including Dave Soldan, Joe Fogler, and others.

Introduction

A useful tool in many signal processing applications is a spectrograph. This powerful instrument was first developed for analyzing speech by Koenig, Dunn and Lacey (ref.1) in 1946. Their original instrument modulated a recorded segment of speech several times with a controlled carrier frequency and fixed band-pass filter. Many different types of spectrograph machines have been developed since then with different features depending on the application. One special variation, for example, was a speech spectrograph developed by Huggins in 1952 which emphasized the three formants of speech, and suppressed unwanted spectral power.

With the evolution of digital computers, more and more spectrograms are generated digitally, rather than with analog filters. Particularly since the development of the Cooley-Tukey algorithm for the fast Fourier transform (FFT) in 1965, digital processing of speech and other signals has become a practical alternative to the analog spectrograph. Digital systems, particularly those built around minicomputers, offer more versatility and adaptability than analog instruments, and many have comparable speed. Some systems using fast mini-computers and approximation techniques have even approached real time processing. (ref.2)

There are several possible applications of a spectrogram display. It can be an invaluable tool in studying the make-up of speech in the development of speech recognition and speech synthesis systems. As a type of "visible speech", it may also be useful in teaching deaf persons how to speak better. (ref.6) Other signals besides speech may also be analyzed. Many complex signals can be much better understood by observing a spectrogram. For example, the effectiveness of an adaptive filter algorithm may be evaluated by analyzing how quickly undesirable periodic noise is filtered out.

General Description of Procedure

In the system described here a NOVA minicomputer was used to process digitized data and plot a spectrogram on a Tektronix graphics terminal. A block diagram of the process involved in generating the spectrograms is given in fig. 1. Any type of signal can be analyzed, although the examples presented here are all speech signals sampled at 8000 sps. After a signal has been digitized and stored on disk, a program written in Fortran processes the data, reading sampled values from the disk as they are needed. Blocks of sampled data are converted to the frequency domain with an FFT algorithm. The spectral power is computed from the resultant coefficients and quantized on a logarithmic scale. The quantization level of each spectral point determines

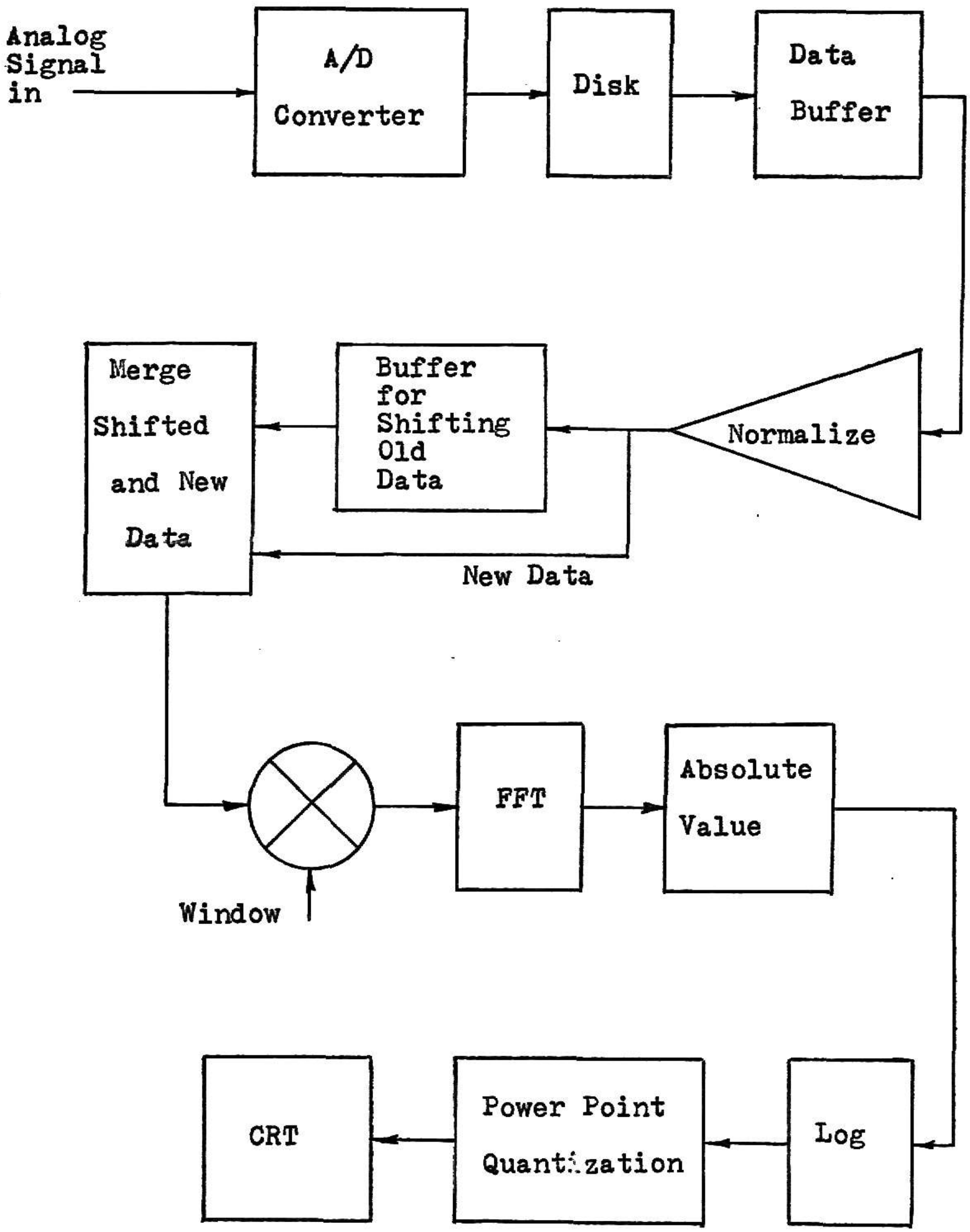


fig. 1 Block Diagram

the intensity of the corresponding plotted point.

Examples of narrow-band spectrograms and a wide-band spectrogram are presented. In the narrow-band case, data blocks of 512 samples are processed, compared to 128 in the wide-band case. By processing four times as many samples per block, the narrow-band algorithm yields four times the spectral points. However, one time slot spans 64 msec, compared to 16 msec in the wide-band case, and there is four times the time shift between data blocks in the narrow-band case. Thus, the trade-off is between four times the frequency resolution and four times the time resolution. Often a signal will be analyzed with two sizes of data blocks in order to give more complete information about it.

Data Acquisition and Preprocessing

The acquisition of a data file to be processed is accomplished by two assembly language subroutines ROPEN and DISKR. ROPEN uses three parameters, FNAME(1), ID, and IFLAG. FNAME is an integer vector which should contain the name of the requested file, packed two ASCII characters to a word and left justified. ID is the file type specification shich is defaulted to if none is given. It is set to 'SO' in this program. IFLAG is a variable which indicates the results of the search for the requested file name. If the file is found, and it has legal characters, a 1 is returned. Otherwise, the user is requested by the main program to designate another file name.

DISKR then reads in the binary data using two parameters, BUF(I) and N. BUF(I) is an integer vector into which the data is read sequentially, beginning at element I. N points are read in. The designated data file is assumed to contain 16 bit signed binary data. The fraction of the allowable range (-32768 to +32768) that is spanned by the data is not critical, since a user input displacement factor (DISP) can be adjusted to allow the full potential of the dynamic range of the spectrogram plot to be used. Thus, no restrictions are placed on the range of the data.

In the case of the narrow-band spectrogram, the main program initially reads in the first 512 data points for processing into integer data buffer BUF(I). After these points have been processed, the latter 448 points are shifted to the beginning of the data buffer, affecting a time shift as indicated in fig. 2. Retaining the latter 448 data points corresponds to an 87.5% overlap of the data blocks. Then, 64 new points are read with DISKR into the latter 64 buffer slots. Thus, the frequency content of the signal is computed as it changes in time, with an 87.5% overlap of time slots. This time shift process is identical in the wide-band case, except that only 128 points are processed in one block, and the time shift is 16 rather than 64 points.

Each new data block is normalized to a change between -1

Time Shift of Data Blocks

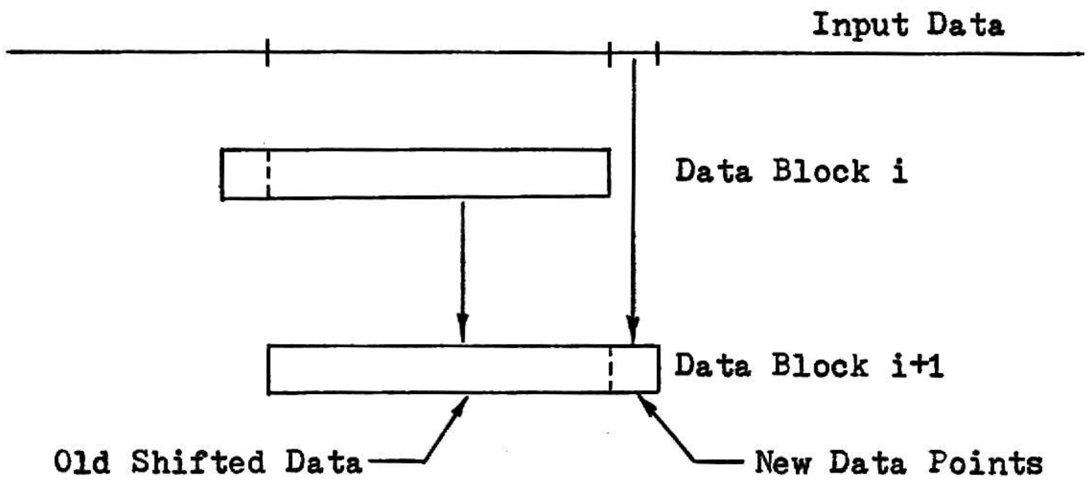
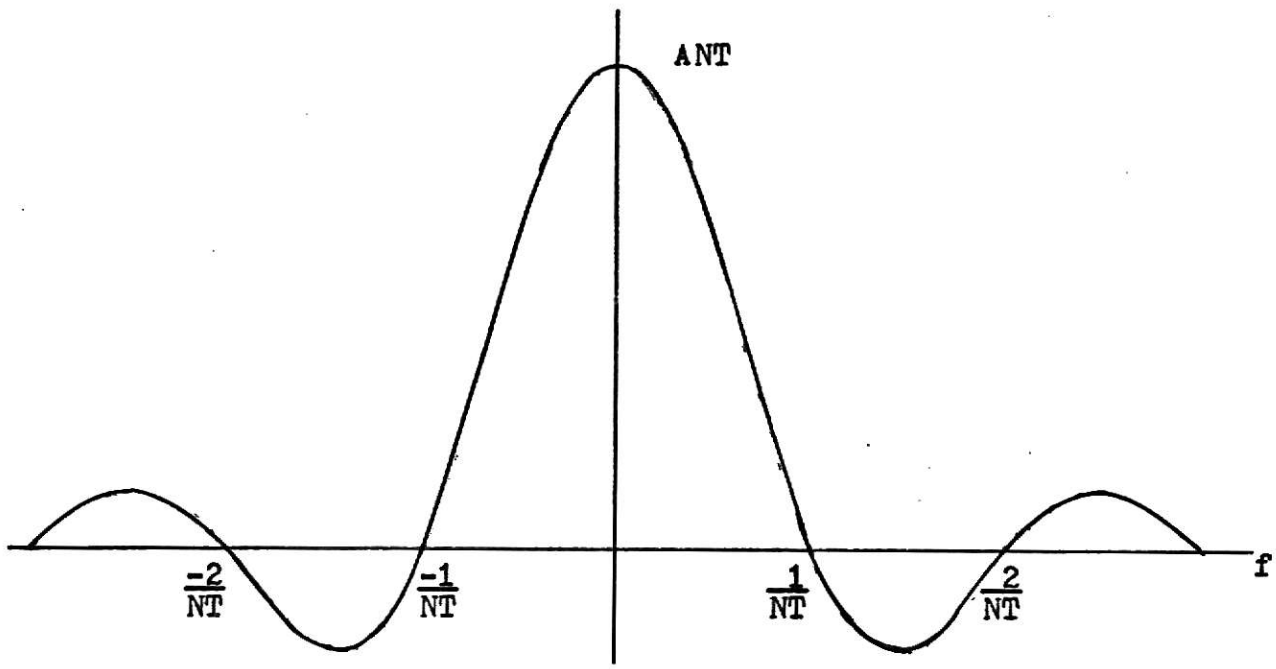


fig. 2



Filter Characteristic of a Rectangular Window

fig. 3

and +1 and put into a complex vector X(I) in preparation for the FFT algorithm. Although the data is real valued, of course, the FFT algorithm destroys the real data values and replaces them with the complex transform coefficients (C_k). Because of this, the normalized data is saved in a separate data vector V(I), so that the last 448 values are retained and can be shifted and reprocessed with the new 64 values as described. In other words, only the new values need be read in.

The Effect of the Time Window on Distortion

In processing one segment of N samples, the signal is effectively being windowed by a rectangular pulse of width NT, where T is the sample time. This pulse or window is then shifted in time in order to compute the variation of the power spectrum with time. The effect of this rectangular window is to spread out or "smear the component frequencies in the signal. This tends to distort the resulting spectrogram as can be seen in fig. 4, for instance, which shows a narrow-band spectrogram of about one second of speech using a 64 msec rectangular window.

This distortion can be explained with the modulation theorem. Essentially, each frequency component in the signal modulates the rectangular pulse window. By the modulation theorem, we know that a term with frequency f_c Hz shifts the frequency spectrum of the

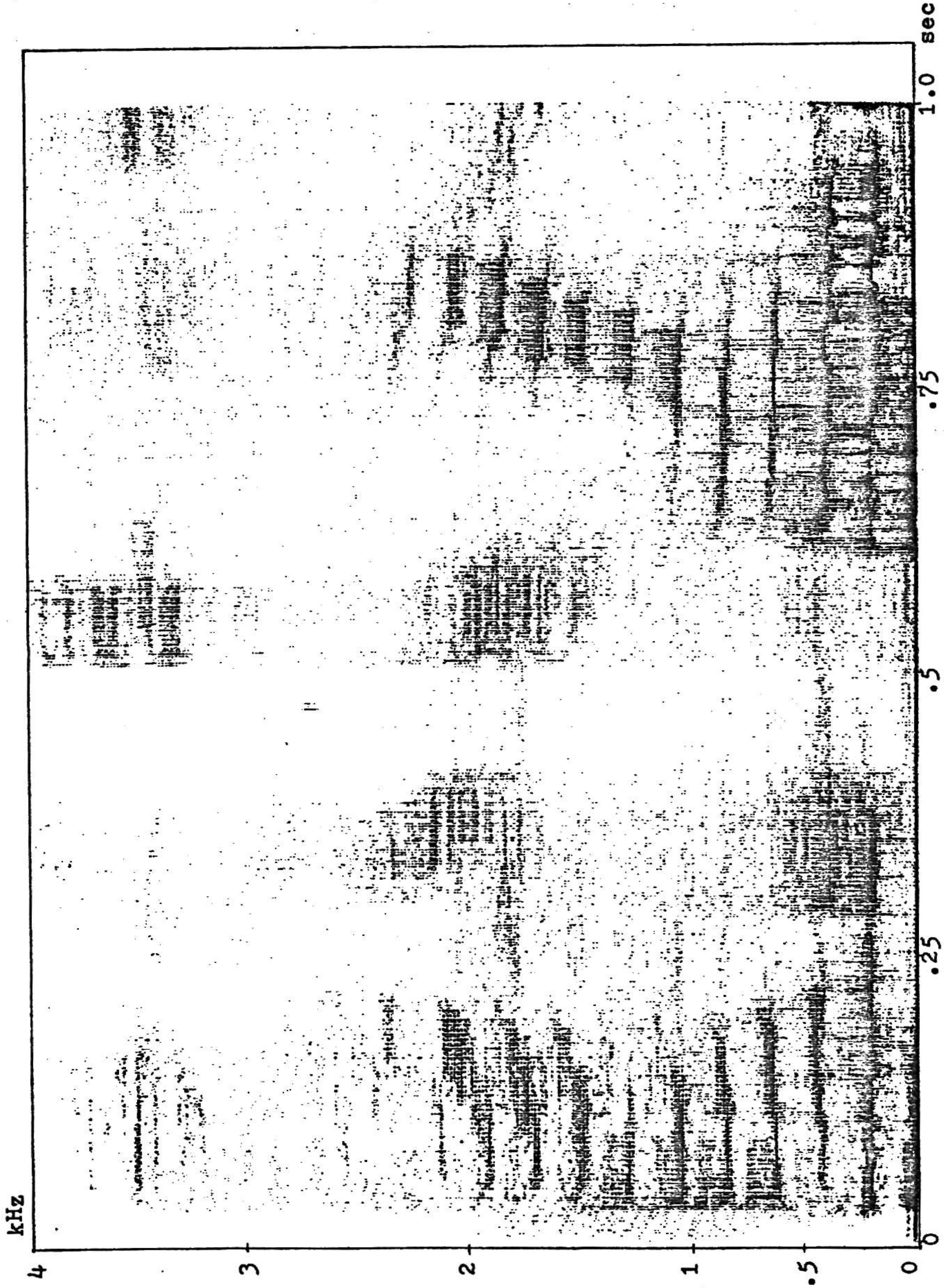


fig. 4 Narrow-band Speech Spectrogram
with Rectangular Window

pulse by f_c Hz. If this frequency term has amplitude A, and the rectangular window has unit height, then a one-sided spectrum of the shifted pulse will have maximum height ANT at f_c. The first nulls away from f_c will occur at $f_c + 1/NT$ and $f_c - 1/NT$, as shown in fig. 3. Thus, each frequency component is effectively distorted by the pulse power spectrum and a narrower pulse window causes more distortion.

The only way to have no distortion at all would be to sample the signal for a very long time, which of course would defeat the purpose of computing the time variation of the power spectrum. One way to lessen the amount of distortion without sacrificing time resolution is to use a different type window than the rectangular pulse. One type window often used is the Hanning window, defined as (ref. 9):

$$h(n) = \tfrac{1}{2}(1 - \cos(2\pi n/N)) \qquad\qquad 0 \le n \le N$$

The filter characteristic of this window is derived in the appendix. It is shown versus time and in the frequency domain in fig. 5. Note that the separation between the first nulls is greater than that between the nulls of the rectangular pulse. However, the spectrum quickly dies out beyond the first null, giving a superior response overall. This lessening of distortion is seen clearly in fig. 6, which is a narrow-band spectrogram of the same speech segment as fig. 4, only using a Hanning window. The improved sharpness of the frequency bands is quite noteable.

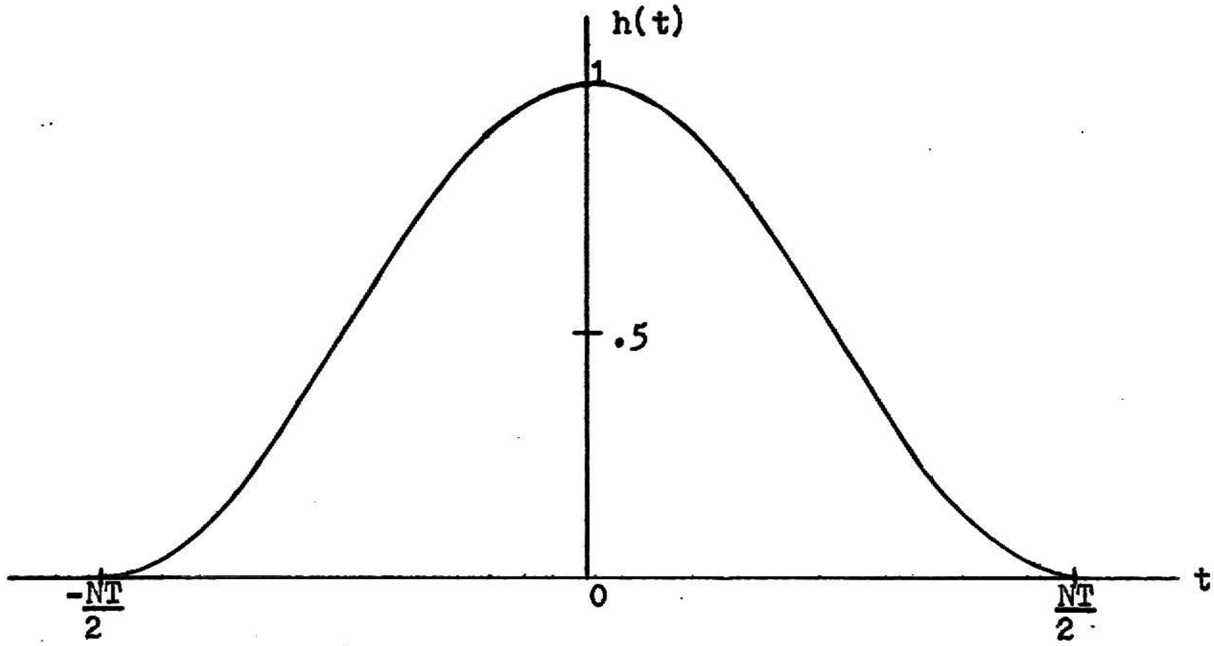


fig. 5a   Hanning Window

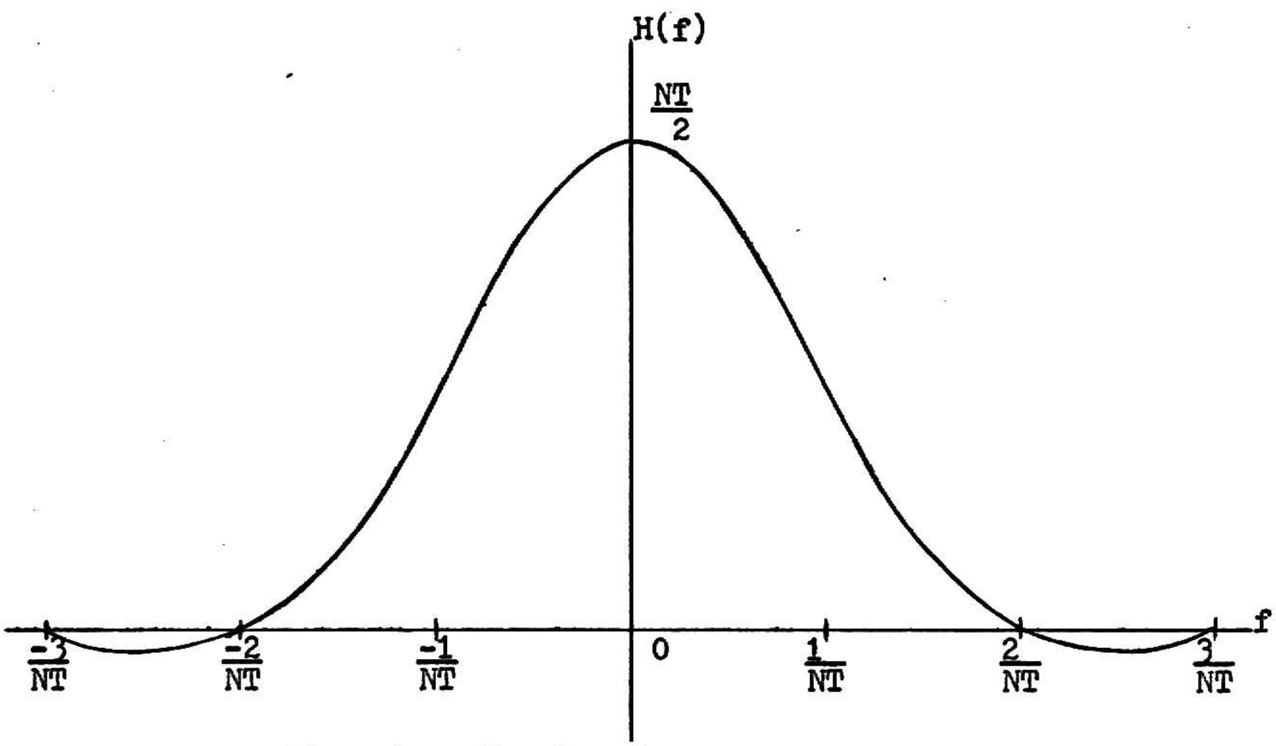


fig. 5b   Hanning Filter Characteristic

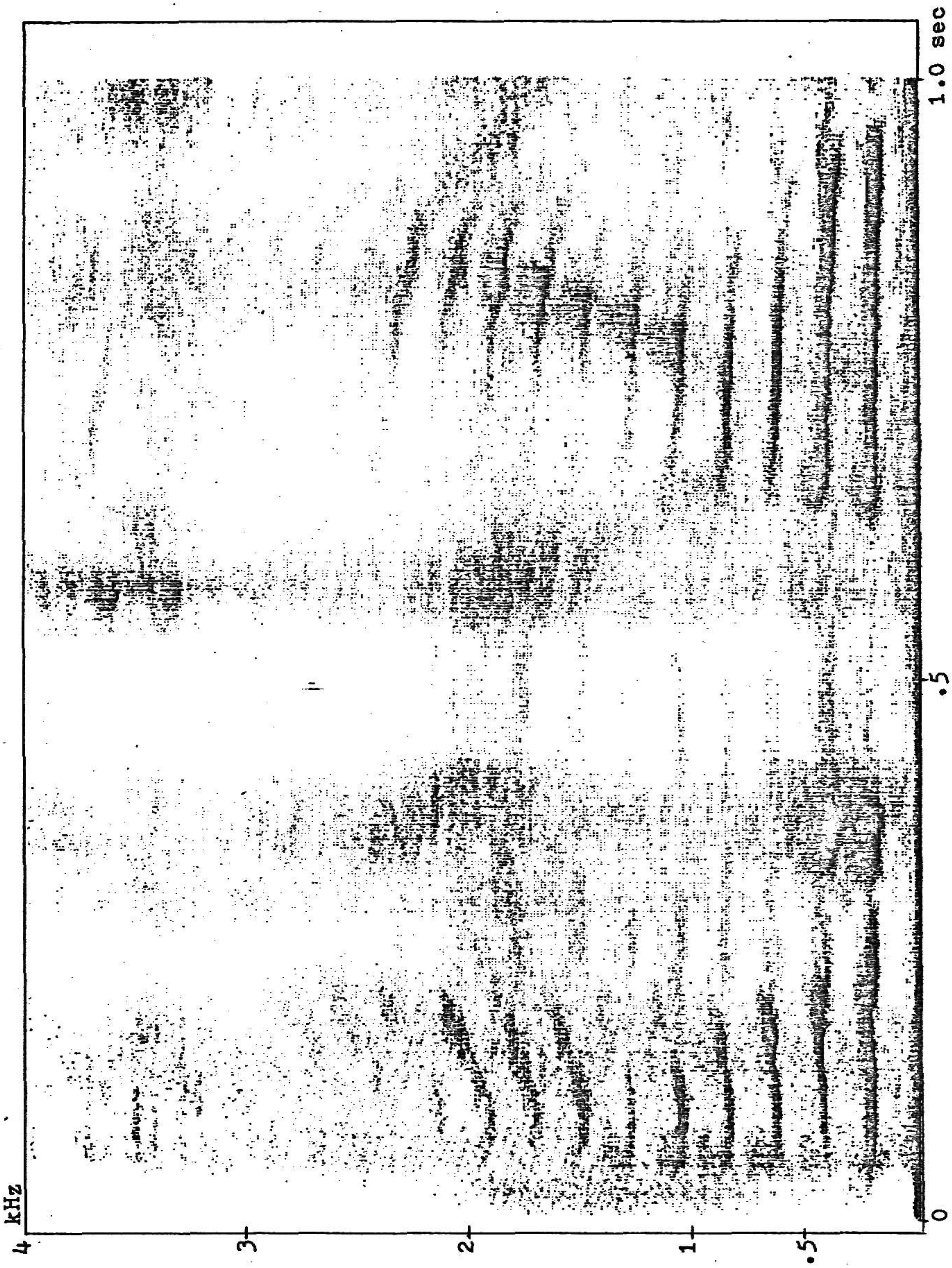


fig. 6 Narrow-band Speech Spectrogram with Hanning Window

Computation of Spectral Points

The fast Fourier transform (FFT) is an algorithm for efficiently computing the discrete Fourier transform (DFT), defined as:

$$C_x(k) = \sum_{n=0}^{N-1} x(nT) \exp(-j2\pi nk/N) \qquad k=0,1,2,...(N-1)$$

where $x(nT)$ is the value of the nth sample, N is the number of samples, T is the sample time, and $C_x(k)$ is the kth transform coefficient. The FFT algorithm was developed by J. W. Cooley and J. W. Tukey in 1965 (ref. 3), and it reduced the required multiplications for an N point DFT from about $2N^2$ to around $N \log_2 N$. The FFT routine used in this program is a FORTRAN version based on a description of the FFT using matrix products (ref. 4).

Once the DFT coefficients have been evaluated at a particular time slot, the resulting data must be processed to give useful information and a meaningful display. The desired information is the power associated with each spectral point. By Parseval's theorem, we know that:

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} x^2(nT) = \sum_{k=0}^{N-1} \left|C_x(k)\right|^2$$

where P_x is the total power in the signal $x(nT)$ during one data block or time slot. Thus $\left|C_x(k)\right|^2$ is the contribution to the signal power of the kth spectral coefficient. There are actually only $N/2 + 1$ independent spectral points ($k=0,1,2,\ldots,N/2$), because of the complex conjugate theorem (ref. 8):

$$C_x(N/2 + L) = C_x^*(N/2 - L)$$

where * indicates complex conjugate. This is easily proven with the definition of the DFT.

In the case of many types of analyzed signals, such as speech, for example, the power levels of the spectral points may vary several orders of magnitude. A linear scaling of point intensity versus power may therefore be inadequate to display the full range of power levels. In order to properly display the frequency components of low power level, a logarithmic scale is used, related to the spectral power in decibels. Power points range over 13 discrete levels, from level zero up to level 12. Which of the 13 levels is displayed is determined in the main program by a function given as:

$$\text{MAG} = \text{INT}(\text{WM}(\log\left|N \cdot C_x(k)\right|) + \text{DISP}),$$

where WM is a user input multiplier and DISP is a user input displacement factor. The INT function truncates a floating point number to an integer. The MAG function can be rearranged to a more convenient form, expressing the power of a spectral point in decibels. The power of point k in decibels is:

$$(P_k)_{dB} = 10 \log P_k = 10 \log \left|C_x(k)\right|^2$$
$$= 20 \log \left|C_x(k)\right|$$

The power level function can be expanded out to:

$$\text{MAG} = \text{INT}(\text{WM}(\log \left|C_x(k)\right|) + \text{WM}(\log N) + \text{DISP})$$

$$MAG = INT((WM/20)(20 \log |C_x(k)|) + WM(\log N) + DISP)$$

$$= INT((WM/20)(P_k)_{dB} + WM(\log N) + DISP) \qquad \text{(eqn. 1)}$$

This discrete spectral function is thus shown to be proportional to the spectral power in decibels. In order to check the sensitivity of this function to change in power level, we define the dB change in power between discrete power levels as $(\Delta P_k)_{dB}$. Since the WM(log N) and DISP terms are both constants, a change of one discrete level corresponds to the term $(WM/20)(\Delta P_k)_{dB}$ changing by 1. Thus:

$$(WM/20)(\Delta P_k)_{dB} = 1$$

and $$(\Delta P_k)_{dB} = 20/WM$$

The quantum step between power levels is seen to be inversely proportional to the multiplier WM. A larger WM therefore corresponds to a smaller step size and more sensitivity. This relationship between $(\Delta P_k)_{dB}$ and WM is shown graphically in fig. 8. Furthermore, since there are only twelve discrete power levels, the total dynamic range of a spectrogram is given by $13(\Delta P_k)_{dB}$, and a larger WM corresponds to a smaller range.

Once the WM multiplier has been specified, the DISP factor can be chosen so as to ensure maximum use of the available dynamic range without overranging. From equation 1, we see that DISP is given by:

$$DISP = MAG - WM(\log N) - WM/20\ (P_k)_{dB}$$

In order to prevent overranging, we must require that:

$$DISP = 13 - WM(\log N) - WM/20\ (P_k)_{max} \qquad \text{(eqn. 2)}$$

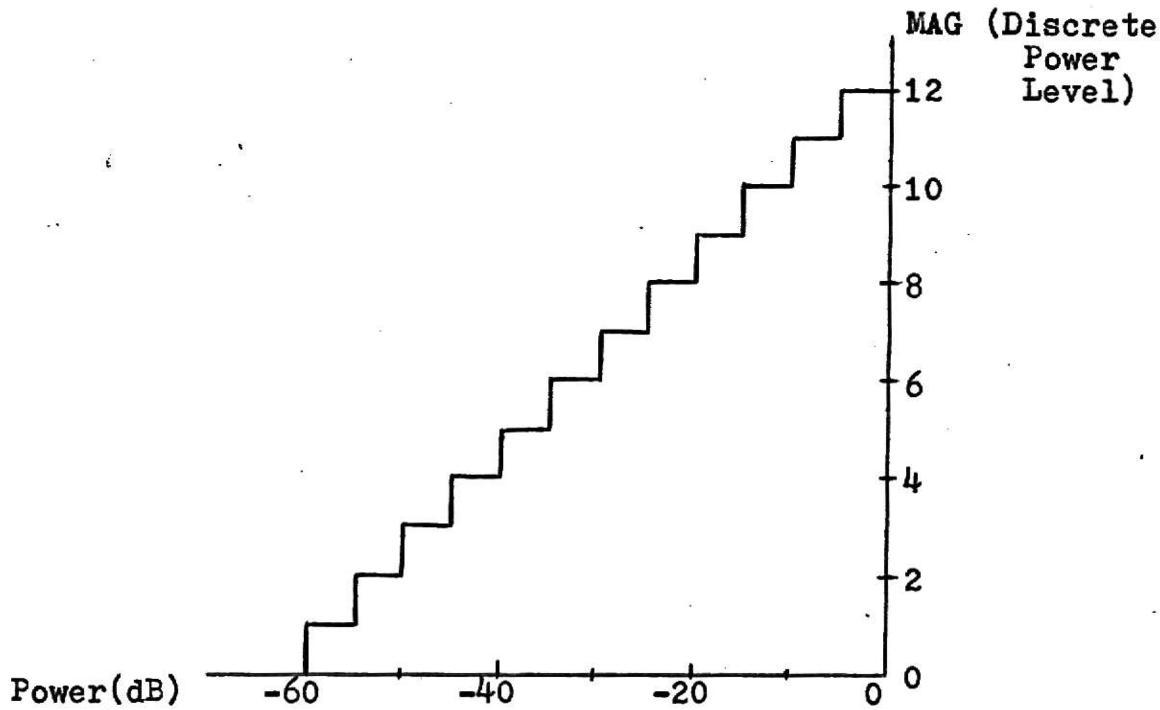


fig. 7 Quantization of Power Level

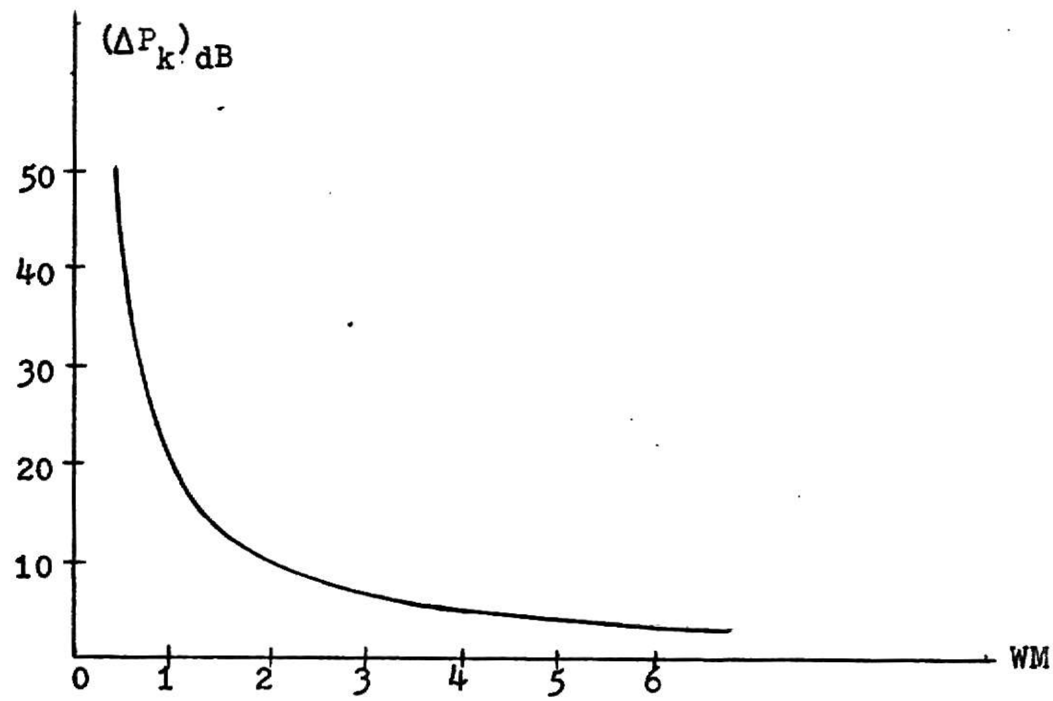


fig. 8 Step Size versus Multiplier (WM)

$(P_k)_{max}$ is the maximum power in decibels expected in any one spectral point. If this condition is satisfied, then MAG will never be truncated to any integer above 12, and there will be no overranging.

As an example case, consider a narrow-band spectrogram which utilizes a 512 point FFT and assume that the maximum expected amplitude of any one spectral point is 0.15. Further assume that we desire to resolve between spectral power points 5 dB apart, which will give us a maximum possible dynamic range of 65 dB. We know that WM is given by:

$$WM = 20/(\Delta P_k)_{dB} = 20/5 = 4$$

The maximum power in any one spectral point can be found from:

$$(P_k)_{max} = 20 \log \left| C_x(k) \right|_{max} = 20 \log(.15) = -16.48 \text{ dB}$$

All the variables in equation 2 are now known, so that the restriction on DISP to prevent overranging can be calculated. The result in this example is: DISP=5.45. DISP should be chosen close to the limit, so that most of the potential range is used.

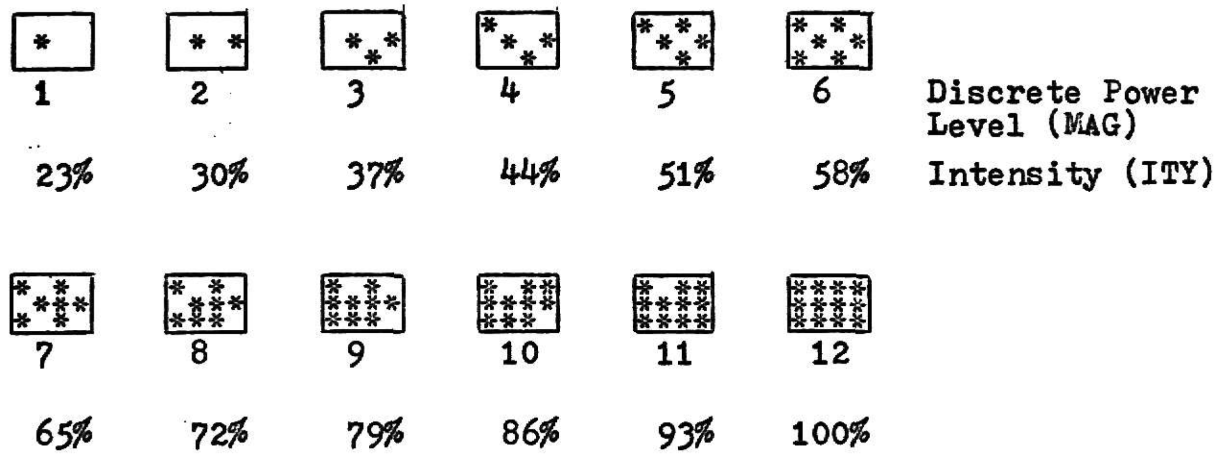
Once the power level of the spectral points has been determined, this information must be displayed in such a manner as to allow the user to easily see how the power density spectrum of the analyzed signal changes with time. Each time slot is represented by a vertical line made up of spectral points. In order to insure a smooth display in the narrow-band case, an extra time slot is added between each pair of original time slots. This extra spectrum is the average of the two spectrums it separates.

On the other hand, in the case of a wide-band spectrogram, while there is adequate time resolution, there is poor frequency resolution. Thus, to smooth out a wide-band display, an average point is placed between the spectral points within each time slot.

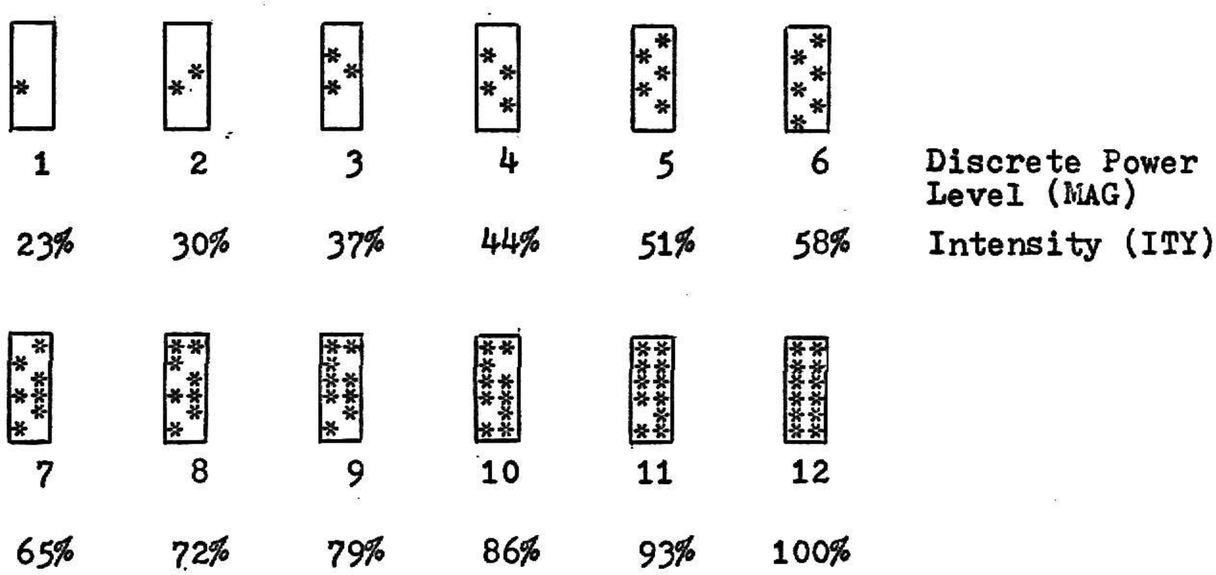
The spectral points themselves are plotted using a point plotting routine POINT(0, ITY, X, Y), which plots a single point with intensity ITY at the specified coordinate (X, Y). ITY is an integer which can range between 0 and 100. The ranges of the X and Y coordinates are 0 to 1023 and 0 to 767 respectively. Actually, each spectral point is an array of small points which form a 4 x 3 block in the narrow-band case, and a 2 x 6 block in the wide-band case. The number of points plotted in a given block corresponds to the power level of the spectral point. In fig. 9, it can be seen how the points are arranged in the array for each of the twelve possible power levels. In order to increase the contrast between power levels, the intensity of the points plotted in the array is a function of the number of points plotted. This intensity is given by:

$$ITY = 7(MAG) + 16$$

Fig. 9 also shows this variation in intensity with power level.



Narrow-band Spectrogram Power Point Arrays



Wide-band Spectrogram Power Point Arrays

fig. 9

Verification of Results

In order to verify that this method of generation of spectrograms is effective, several examples of speech signals have been presented. Since all the analyzed data was sampled at 8000 sps, the resulting spectrograms have a bandwidth of 4kHz according to the sampling theorem. A total of 8000 samples were processed in the examples shown in fig. 4 and fig. 6, so that the time spanned was one second. The apparent variation of frequency content with time in these figures is certainly reasonable.

It is evident that the spectrogram in fig. 10 is very similar to the one in fig. 6. These segments of speech were spoken by the same person, a female, who was saying the same phrase both times. The similarity between the two figures demonstrates the consistency of a person's voice, and proves the repeatability of results.

As further evidence towards the credibility of this algorithm, test plots were made using data generated by the computer. A DC term and several sinusoidal functions with various amplitudes were added to form a data function given by:

$$X(I) = 1. + .316 \sin(\pi I/8) + .1 \cos(\pi I/4) + .0316 \sin(\pi I/2) + .01 \cos(3\pi I/4)$$

Note that the frequency of the first sinusoidal term can be found by setting the argument equal to $2\pi fnT$. Thus:

$$\pi I/8 = 2\pi fnT$$

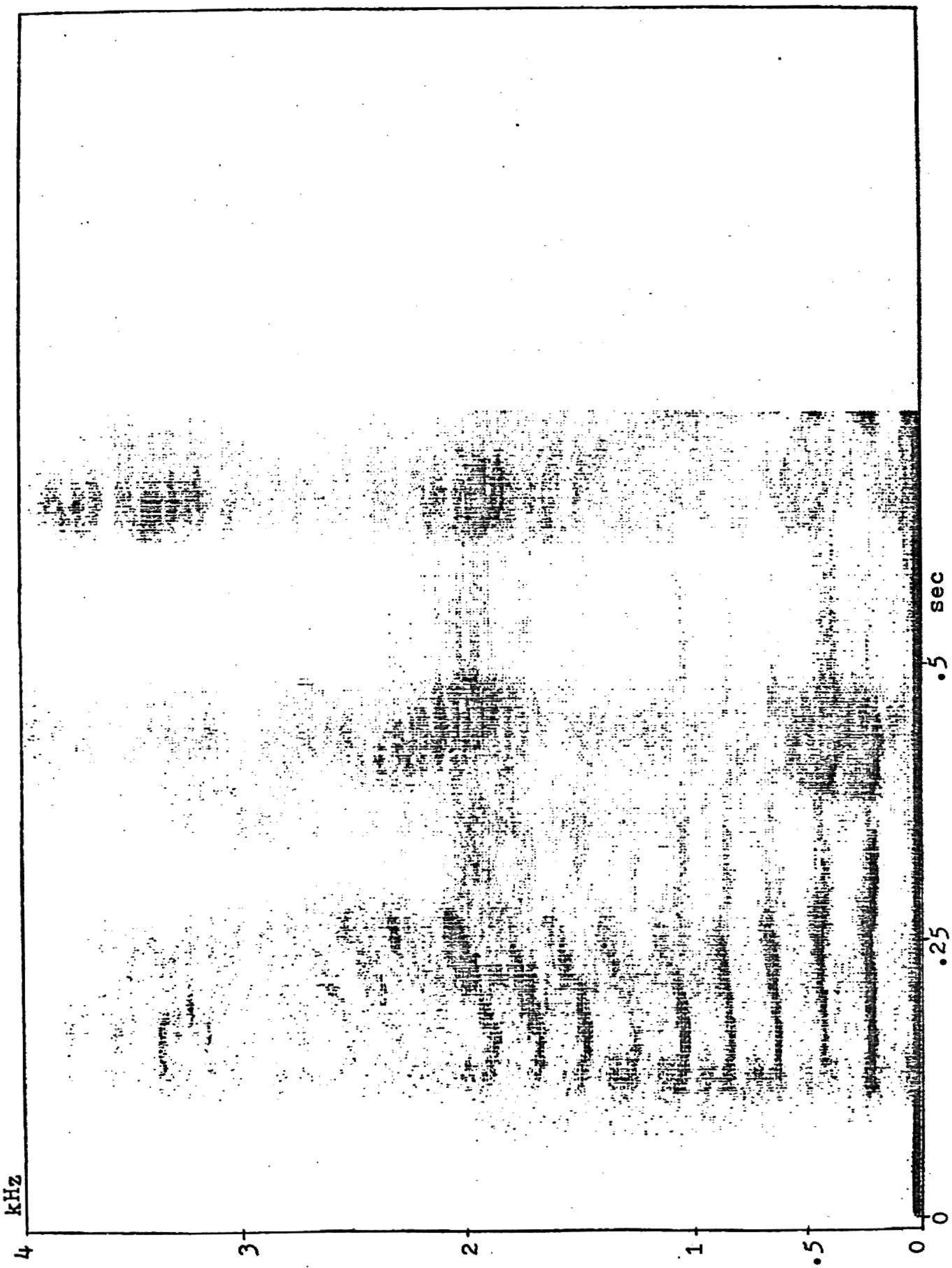


fig. 10 Narrow-band Speech Spectrogram with Hanning Window

Since the time variables n and I are actually the same, we have:

$$f = 1/16T$$

Since the sample time T is 1/8000 sec, the first sinusoidal term is seen to represent a 500 Hz sine function. The succeeding terms are therefore 1kHz, 2kHz, and 3kHz respectively. The amplitude of each sinusoidal term is also seen to be about 5 dB below the preceeding term. This corresponds to a difference in power of 10 dB, which is shown in the resultant test spectrogram, fig. 11. Thus, the variation of intensity with power level is also demonstrated. Also, a wide-band spectrogram is given in fig. 12 to demonstrate the credibility of the wide-band program.

Conclusions

This program for the generation of spectrograms could prove useful in many varied applications concerned with the time variation of spectral power density. One aspect of it which could be improved is its speed. Due to the time involved in computing the FFT, over an hour is required to analyze one second of speech. One possible method of speeding up the FFT is to compute all the complex multipliers and store them in an array at the start of the program. It would then be faster to find a value in the array than to calculate the value each time. One problem with this is the limit on the size of an array in FORTRAN. This is not a problem in the case of the wide-band spectrogram, however, which processes blocks of only 128 points. In order to

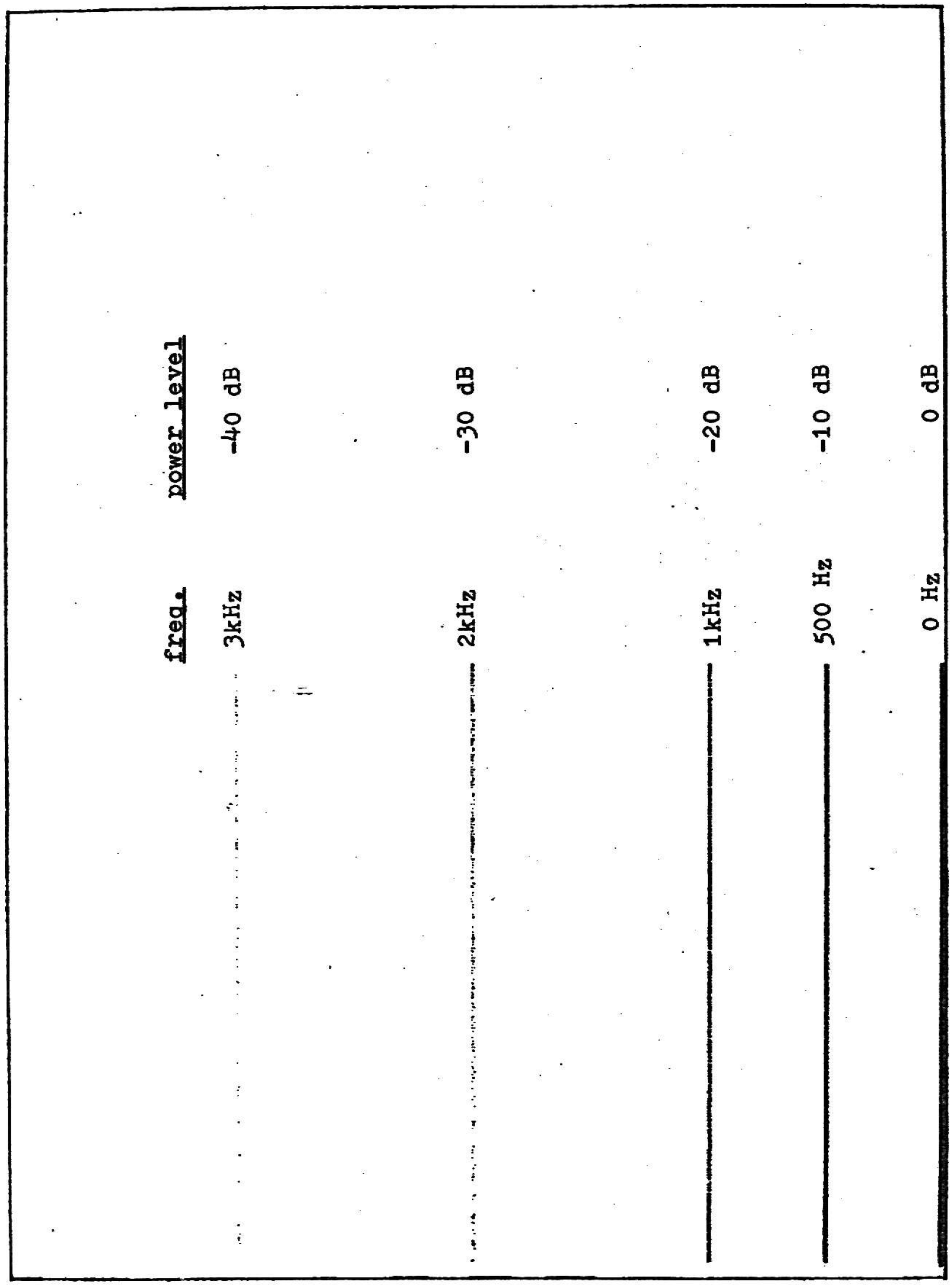


fig. 11 Test Spectrogram

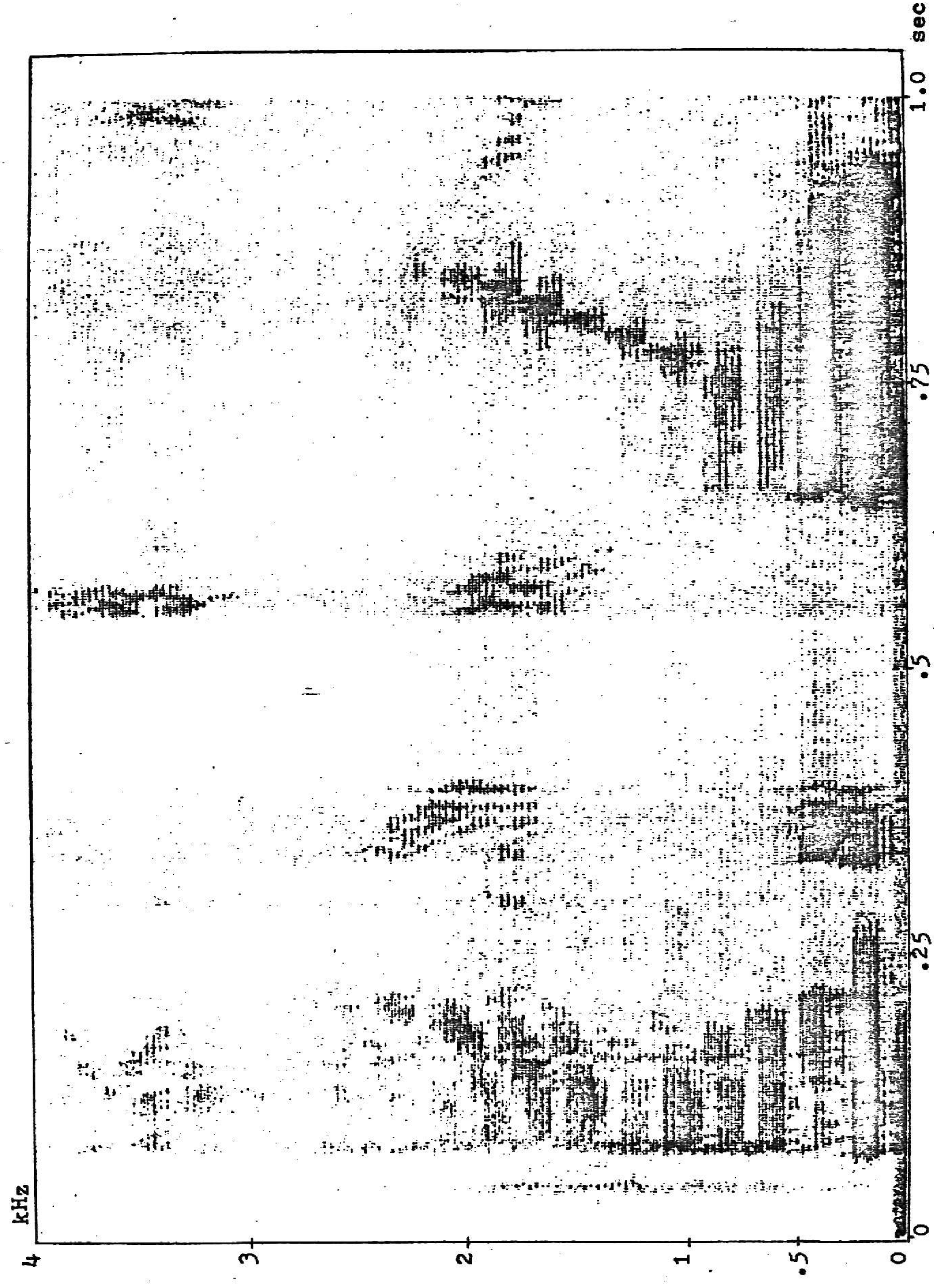


fig. 12 Wide-band Speech Spectrogram with Hanning Window

approach real time processing, the FFT would probably have to be computed with specialized hardware. Another alternative for improving speed is to use a fast Walsh transform, rather than the FFT (ref. 5). Since no multiplications and no complex numbers are involved with the Walsh transform, it can be computed much faster. In its present form, this routine is useful when an immediate analysis is not required. Although it requires a little time, it can still be a convenient instrument for anyone interested in signal analysis and processing.

Appendix

Hanning Window Filter Characteristic

In order to examine the filtering effect of a Hanning window, we need to find its Fourier transform. The Hanning window is given by:

$$h(n) = \tfrac{1}{2}(1 - \cos(2\pi n/N)) \qquad 0 \le n \le N$$

To simplify the result, a time window symmetric about the origin is chosen. Thus:

$$h'(n) = \tfrac{1}{2}(1 + \cos(2\pi n/N) \qquad -N/2 \le n \le N/2$$

This reflects a shift in time only and does not effect the frequency characteristic of the window. The continuous function corresponding to h'(n) can be found by replacing n with t/T. Thus we have:

$$h'(t) = \tfrac{1}{2}(1 + \cos(2\pi t/NT)) \qquad -NT/2 \le t \le NT/2$$

This time window is shown in fig. 5a. The Fourier transform of this function is given by:

$$H(f) = \tfrac{1}{2}\int_{-NT/2}^{NT/2} (1 + \cos(2\pi t/NT))\exp(-j2\pi ft)\ dt$$

$$= \frac{1}{-j4\pi f}\exp(-j2\pi ft)\Bigg|_{-NT/2}^{NT/2} + \tfrac{1}{2}\int_{-NT/2}^{NT/2}\cos(2\pi t/NT)\exp(-j2\pi ft)\ dt$$

$$= \frac{1}{2\pi f}\,\frac{\exp(j\pi fNT) - \exp(-j\pi fNT)}{2j} + \tfrac{1}{2}\int_{-NT/2}^{NT/2}\cos(bt)\exp(at)\ dt$$

where $a = -j2\pi f$ and $b = 2\pi/NT$

$$H(f) = \frac{1}{2\pi f}\sin(\pi fNT) + \tfrac{1}{2}\exp(at)\left.\frac{a\cos(bt) + b\sin(bt)}{(a^2 + b^2)}\right|_{-NT/2}^{NT/2}$$

The result from the latter integral is given in ref. 7 on page 85. Since $\sin(bt)\Big|_{t=NT/2} = \sin(bt)\Big|_{t=-NT/2} = 0,$

$$H(f) = \frac{1}{2\pi f}\sin(\pi fNT) + (a/2)\exp(at)\left.\frac{\cos(bt)}{(a^2 + b^2)}\right|_{-NT/2}^{NT/2}$$

Since $\cos(bt)\Big|_{t=NT/2} = \cos(bt)\Big|_{t=-NT/2} = -1,$

$$H(f) = \frac{1}{2\pi f}\sin(\pi fNT) + (a/2)\frac{\exp(-aNT/2) - \exp(aNT/2)}{(a^2 + b^2)}$$

$$= \frac{1}{2\pi f}\sin(\pi fNT) + \frac{2\pi f}{2j}\frac{\exp(-aNT/2) - \exp(aNT/2)}{(a^2 + b^2)}$$

Since $aNT/2 = -j\pi fNT$,

$$H(f) = \frac{1}{2\pi f}\sin(\pi fNT) + \frac{2\pi f}{(a^2 + b^2)}\sin(\pi fNT)$$

$$a^2 + b^2 = \frac{(2\pi)^2}{(NT)^2} - (2\pi)^2 f^2 = \frac{(2\pi)^2\,(1 - (fNT)^2)}{(NT)^2}$$

$$H(f) = \frac{NT}{2}\frac{\sin(\pi fNT)}{\pi fNT} - \frac{(fNT)^3}{2f((fNT)^2 - 1)}\frac{\sin(\pi fNT)}{\pi fNT}$$

Let $fNT = x$

$$H(f) = \frac{x}{2f} - \frac{x^3}{2f(x^2 - 1)}\frac{\sin(\pi x)}{\pi x} = \frac{x^3 - x - x^3}{2f(x^2 - 1)}\frac{\sin(\pi x)}{\pi x}$$

$$H(f) = \frac{x}{2f(1 - x^2)} \frac{\sin(\pi x)}{\pi x} = \frac{NT}{2(1 - (fNT)^2)} \frac{\sin(\pi fNT)}{\pi fNT}$$

This filter characteristic of the Hanning window is shown in fig. 5b. It is apparent that the Hanning window offers a superior spectral characteristic and does not "smear" the spectral distribution of the sampled function as much as the rectangular window.

```
C       NARROWBAND SPECTROGRAM PROGRAM UTILIZING THE FFT
C
C       ANALYZES UP TO 1 SECOND OF SPEECH AT 8000 SPS
C       OR ANY OTHER DISRETE SIGNAL WITH UP TO 8640 SAMPLES
C
C       VARIABLES:
C           FNAME = NAME OF DATA FILE ON DISK TO BE ANALYZED
C
C           UM = MULTIPLIER WHICH DETERMINES THE POWER RANGE
C                AS DESCRIBED IN THE ACCOMPANYING PAPER
C
C           DISP = DISPLACEMENT FACTOR WHICH ADJUSTS FOR
C                MAXIMUM USE OF POWER RANGE AS DESCRIBED
C                IN THE PAPER
C
        COMPLEX X(512),U,T,V(512)
        INTEGER FNAME(5),BUF(512),MA(256,2)
10      WRITE(10,200)
        ID='SO'
        READ(11,210) (FNAME(I),I=1,5)
C
C       ESTABLISHMENT OF DATA FILE
C
        CALL ROPEN(FNAME(1),ID,IFLAG)
        IF(IFLAG NE 1) GO TO 10
        CALL DISKR(BUF(1),512)
        PI=3 141592
        ANG=PI/256
15      ACCEPT 'NUMBER DATA POINTS (8640 MAX)=',NP
        LAST=(NP-448)/64
        IF(LAST GT 128 OR LAST LT 1)GO TO 15
        ACCEPT 'WINDOW? (1=YES, 0=NO)',IW
        ACCEPT 'LOG MULTIPLIER=',UM
        ACCEPT 'DISPLACEMENT=',DIS
C
C       DRAW OUTLINE AND PUT IN POINT PLOT MODE
C
        CALL INIT(12,0,0)
        CALL VECT(0,10,0,767)
        CALL VECT(0,10,1023,767)
        CALL VECT(0,10,1023,0)
        CALL VECT(0,10,0,0)
        CALL INIT(3,0,0)
        DO 20 J=1,256
20      MA(J,1)=0
C
C       NORMALIZATION OF DATA
C
        DO 35 I=1,512
        X(I)=FLOAT(BUF(I))/32768.
        V(I)=X(I)
35      CONTINUE
C
C       HANNING WINDOW IF REQUESTED
C
        IF(IW EQ 0)GO TO 38
        DO 36 I=1,512
36      X(I)=X(I)*(1.-COS(ANG*FLOAT(I-1)))/2.
38      N=512
        ITER=9
        DO 90 IX=1,LAST
C
C       COMPUTATION OF FFT
C
        NXP2=512
        DO 50 IT=1,ITER
C
C       COMPUTATION FOR EACH ITERATION
C       NXP: NUMBER OF POINTS IN A PARTITION
C       NXP2  NXP/2
C
        NXP=NXP2
        NXP2=NXP2/2
        WPWR=3 141592/FLOAT(NXP2)
        DO 40 M=1,NXP2
C
C       CALCULATE THE MULTIPLIER
C
        ARG=FLOAT(M-1)*WPWR
        W=CMPLX(COS(ARG),-SIN(ARG))
        DO 40 MXP=NXP,N,NXP
C
C       COMPUTATION FOR EACH PARTITION
C
        J1=MXP-NXP+M
        J2=J1+NXP2
        T=X(J1)-X(J2)
        X(J1)=X(J1)+X(J2)
40      X(J2)=T*W
50      CONTINUE
C
C       UNSRCAMBLE THE BIT-REVERSED DFT COEFS
C
        N2=N/2
        N1=N-1
        J=1
        DO 65 I=1,N1
        IF(I GE J)GO TO 55
        T=X(J)
        X(J)=X(I)
        X(I)=T
55      K=N2
60      IF(K GE J) GO TO 65
        J=J-K
        K=K/2
        GO TO 60
65      J=J+K
C
C       COMPUTATION OF POWER LEVEL AND PLOTTING
C       OF POWER POINTS
C       K1=1: PLOT THE AVERAGE OF PRESENT AND
C       PREVIOUS SEGMENTS
C       K1=2: PLOT PRESENT SEGMENT
C
        DO 84 K1=1,2
        DO 84 I=1 256
        IF(K1 EQ 1)GO TO 67
        MAG=MA(I,1)
        GO TO 68
67      MA(I,2)=MA(I,1)
        MA(I,1)=INT(UM*ALOG10(CABS(X(I)))+DIS)
        MAG=(MA(I,1)+MA(I,2))/2
68      IF(MAG LT 1)GO TO 84
        NX=IX*8+(K1-1)*4
        NY=(I-1)*3
        ITY=7*MAG+16
```

```
      GO TO(82,81,80,79,78,77,76,75,74,73,72,71)MAG
71    CALL POINT(0,ITY,NX+1,NY+2)
72    CALL POINT(0,ITY,NX+3,NY)
73    CALL POINT(0,ITY,NX+3,NY+2)
74    CALL POINT(0,ITY,NX,NY+1)
75    CALL POINT(0,ITY,NX+1,NY)
76    CALL POINT(0,ITY,NX+2,NY+1)
77    CALL POINT(0,ITY,NX,NY)
78    CALL POINT(0,ITY,NX+2,NY+2)
79    CALL POINT(0,ITY,NX,NY+2)
80    CALL POINT(0,ITY,NX+2,NY)
81    CALL POINT(0,ITY,NX+3,NY+1)
82    CALL POINT(0,ITY,NX+1,NY+1)
84    CONTINUE
C
C     TIME SHIFT OF DATA POINTS
C
      DO 85 I=1,448
      V(I)=V(I+64)
85    X(I)=V(I)
C
C     READING IN, NORMALIZATION, AND POSSIBLE
C     WINDOWING OF NEW DATA POINTS
C
      CALL DISKR(BUF(449),64)
      DO 87 I=449,512
      V(I)=FLOAT(BUF(I))/32768.
87    X(I)=V(I)
      IF(IU.EQ.0)GO TO 90
      DO 83 I=1,512
83    X(I)=V(I)*(1.-COS(ANG*FLOAT(I-1)))/2.
90    CONTINUE
200   FORMAT('CENTER FILE NAME')
210   FORMAT(5A2)
      END
```

```
C       WIDEBAND SPECTROGRAM PROGRAM UTILIZING THE FFT
C
C       ANALYZES UP TO 1 SECOND OF SPEECH AT 8000 SPS
C       OR ANY OTHER DISRETE SIGNAL WITH UP TO 8304 SAMPLES
C
C       VARIABLES
C           FNAME = NAME OF DATA FILE ON DISK TO BE ANALYZED
C
C           UM = MULTIPLIER WHICH DETERMINES THE POWER RANGE
C               AS DESCRIBED IN THE ACCOMPANYING PAPER
C
C           DISP = DISPLACEMENT FACTOR WHICH ADJUSTS FOR
C               MAXIMUM USE OF POWER RANGE AS DESCRIBED
C               IN THE PAPER
C
C
        COMPLEX X(128),U,T,V(128)
        INTEGER FNAME(5),BUF(128),MA(65,2)
10      WRITE(10,200)
        ID='SO'
        READ(11,210) (FNAME(I),I=1,5)
C
C       ESTABLISHMENT OF DATA FILE
C
        CALL ROPEN(FNAME(1),ID,IFLAG)
        IF(IFLAG NE 1) GO TO 10
        CALL DISKR(BUF(1),128)
        PI=3 141592
        ANG=PI/64
15      ACCEPT 'NUMBER DATA POINTS (8304 MAX)=',NP
        LAST=(NP-112)/16
        IF(LAST GT 512 OR LAST LT 1)GO TO 15
        ACCEPT 'WINDOW? (1=YES, 0=NO)',IU
        ACCEPT 'LOG MULTIPLIER=',UM
        ACCEPT 'DISPLACEMENT=',DIS
C
C       DRAW OUTLINE AND PUT IN POINT PLOT MODE
C
        CALL INIT(12,0,0)
        CALL VECT(0,10,0,767)
        CALL VECT(0,10,1023,767)
        CALL VECT(0,10,1023,0)
        CALL VECT(0,10,0,0)
        CALL INIT(3,0,0)
        DO 20 J=1,64
20      MA(J,1)=0
C
C       NORMALIZATION OF DATA
C
        DO 35 I=1,128
        X(I)=FLOAT(BUF(I))/32768.
        V(I)=X(I)
35      CONTINUE
C
C       HANNING WINDOW IF REQUESTED
C
        IF(IU EQ.0)GO TO 38
        DO 36 I=1,128
36      X(I)=X(I)*(1.-COS(ANG*FLOAT(I-1)))/2.
38      N=128
        ITER=7
        DO 90 IX=1,LAST
C
C       COMPUTATION OF FFT
C
        NXP2=128
        DO 50 IT=1,ITER
C
C       COMPUTATION FOR EACH ITERATION
C       NXP: NUMBER OF POINTS IN A PARTITION
C       NXP2: NXP/2
C
        NXP=NXP2
        NXP2=NXP2/2
        UPUR=3 141592/FLOAT(NXP2)
        DO 40 M=1,NXP2
C
C       CALCULATE THE MULTIPLIER
C
        ARG=FLOAT(M-1)*UPUR
        U=CMPLX(COS(ARG),-SIN(ARG))
        DO 40 MXP=NXP,N,NXP
C
C       COMPUTATION FOR EACH PARTITION
C
        J1=MXP-NXP+M
        J2=J1+NXP2
        T=X(J1)-X(J2)
        X(J1)=X(J1)+X(J2)
40      X(J2)=T*U
50      CONTINUE
C
C       UNSRCAMBLE THE BIT-REVERSED DFT COEFS.
C
        N2=N/2
        N1=N-1
        J=1
        DO 65 I=1,N1
        IF(I GE J)GO TO 55
        T=X(J)
        X(J)=X(I)
        X(I)=T
55      K=N2
60      IF(K GE J) GO TO 65
        J=J-K
        K=K/2
        GO TO 60
65      J=J+K
C
C       COMPUTATION OF POWER LEVEL AND PLOTTING
C       OF POWER POINTS
C       K1=1  PLOT THE AVERAGE OF PRESENT AND
C       PREVIOUS SEGMENTS
C       K1=2  PLOT PRESENT SEGMENT
C
        DO 84 K1=1,2
        DO 84 I=2,65
        IF(K1 EQ 1)GO TO 67
        MAG=MA(I,1)
        GO TO 68
67      MA(I,2)=MA(I,1)
        MA(I,1)=0
        IF(CABS(X(I)) LT. 001)GO TO 84
        MA(I,1)=INT(UM*ALOG10(CABS(X(I)))+DIS)
        MAG=(MA(I,1)+MA(I,2))/2
68      IF(MAG LT 1)GO TO 84
        NX=IX*2
```

```
      NY=6*(2*I+K1-5)
      ITY=7*MAG+16
      GO TO(82,81,80,79,78,77,76,75,74,73,72,71)MAG
71    CALL POINT(0,ITY,NX,NY+1)
72    CALL POINT(0,ITY,NX+1,NY+4)
73    CALL POINT(0,ITY,NX+1,NY)
74    CALL POINT(0,ITY,NX,NY+3)
75    CALL POINT(0,ITY,NX,NY+5)
76    CALL POINT(0,ITY,NX+1,NY+2)
77    CALL POINT(0,ITY,NX,NY)
78    CALL POINT(0,ITY,NX+1,NY+5)
79    CALL POINT(0,ITY,NX+1,NY+1)
80    CALL POINT(0,ITY,NX,NY+4)
81    CALL POINT(0,ITY,NX+1,NY+3)
82    CALL POINT(0,ITY,NX,NY+2)
84    CONTINUE
C
C     TIME SHIFT OF DATA POINTS
C
      DO 85 I=1,112
      U(I)=U(I+16)
85    X(I)=U(I)
C
C     READING IN, NORMALIZATION, AND POSSIBLE
C     WINDOWING OF NEW DATA POINTS
C
      CALL DISKR(BUF(113),16)
      DO 87 I=113,128
      U(I)=FLOAT(BUF(I))/32768.
87    X(I)=U(I)
      IF(IW.EQ.0)GO TO 90
      DO 88 I=1,128
88    X(I)=U(I)*(1.-COS(ANG*FLOAT(I-1)))/2.
90    CONTINUE
200   FORMAT('0ENTER FILE NAME')
210   FORMAT(5A2)
      END

O
```

```
P? DISKR.FS
          SUBROUTINE DISKR(IBUF,N)
A         .ENT     SECT.BLOCK
          INTEGER IBUF,N,I
          I=0
A         STA      3.AC3    .SAVE THE STACK POINTER
A         LDA      0.@T.+1.3
A         STA      0.CTR    ,SET-UP THE WORD COUNTER
A         LDA      0.T..3   ,GET THE BUFFER ADDRESS
A         STA      0.PTR1   ,SET-UP THE BUFFER POINTER
A         INTDS             .DO NO DISTURB
ALOOP:    JSR      GETU     ,GET A WORD
A         STA      0.@PTR1  ,STORE THE WORD IN THE BUFFER
A         ISZ      PTR1     ,BUMP THE BUFFER POINTER
A         DSZ      CTR      ,ARE WE DONE?
A         JMP      LOOP     ,NO. GET ANOTHER WORD
A         NIOC     DSK      .CLEAR THE DISK
A         INTEN             .INABLE THE INTERRUPT
A         LDA      3.AC3    ,GET THE STACK POINTER
          RETURN
AGETU:    STA      3.RET1   .SAVE THE RETURN ADDRESS
A         LDA      0.BLOCK  ,GET THE BUFFER POINTER
A         MOV      0.0.SZR  ,IS THE BUFFER EMPTY?
A         JMP      .+6      .NO. CONTINUE
A         LDA      0.SECT   .GET THE SECTOR ADDRESS
A         LDA      1.ABUF   ,GET THE DISK BUFFER ADDRESS
A         JSR      @DSKR    ,READ A SECTOR
A         STA      0.SECT   ,SAVE THE NEXT SECTOR ADDRESS
A         SUB      0.0      .BLOCK=0
A         LDA      2.ABUF   .GET THE BUFFER ADDRESS
A         ADD      0.2      .FORM THE MEMORY ADDRESS
A         INC      0.0      ,BUMP THE BUFFER POINTER
A         LDA      1..377   ;377
A         AND      1.0      .FORM MODULO 256 NUMBER
A         STA      0.BLOCK  ,SAVE THE BUFFER POINTER
A         LDA      0.0.2    ,GET THE WORD
A         JMP      @RET1    ,RETURN
AAC3:     0
ACTR      0
APTR1     0
ARET1:    0
ABLOCK:   0
ASECT:    0
AABUF:    BUF
A.377:    377
ADSKR:    57777-130
ABUF:     .BLK     400
          END
```

◇

DISKR, ROPEN, and BNAME were written by Dave Hein.

```
PR ROPEN
         SUBROUTINE ROPEN(FNAME.ID.FLAG)
         INTEGER FNAME.ID.FLAG
A        .EXTN    SECT.NAME.BLOCK
         FLAG=0
A        STA      3.AC3     ,SAVE THE STACK POINTER
A        INTDS              ,DO NOT DISTURB
A        LDA      0.@T.+1.3
A        LDA      1.T.,3    ,GET THE FILE NAME
A        M0VZL    1.1       ,FORM BYTE ADDRESS
A        JSR      @FNAME    ,GET THE FILE NAME
A        MOV      0.0.SNR   ,FILE FOUND?
A        JMP      RTURN     ,RETURN
A        COM#     0.0.SNR   ,ERROR IN FILE NAME?
A        JMP      RTURN     ,RETURN
AFOUND:  LDA      3.WORKF   ,GET THE ADDRESS OF THE WORKFILE TAPLE
A        LDA      1.3.3     ,GET THE SECTOR ADDRESS
A        STA      1.@ASECT        ,SET UP THE DISK ADDRESS POINTER
A        SUB      1.1       ,GENERATE A ZERO
A        STA      1.@ABLOC        ,INITIALIZE THE DISK POINTER
ARTURN:  LDA      3.AC3     ,GET THE  STACK POINTER
A        STA      0.@T.+2.3
A        NIOC     DSK       ,CLEAR THE DISK
A        INTEN              ,INABLE THE INTERRUPT
         RETURN
AASECT:  SECT
AABLOC:  BLOCK
AAC3:    0
AFNAME:  NAME
AWORKF:  57777-351
         END

<>
```

```
PR BNAME.SD
        .TITLE  NAME
        .EXTN   .LDB
        .ENT    NAME
        .DUSR   TOP-57777
        .NREL
NAME:   STA     3.SU3    ,SAVE THE RETURN ADDRESS
        STA     0.ID     ,SAVE THE DEFAULT ID
        STA     1.ADNAM  ,SAVE THE STRING ADDRESS
        JSR     RESET    ,RESET THE WORKFILE TABLE
        LDA     0.NG     ,-6
        LDA     1.WORKF
        JSR     GETN     ,GET THE FILE NAME
        JMP     BADC     ,BAD CHARACTER
        JMP     BADC     ,BAD CHARACTER
        JMP     +2       ,PERIOD
        JMP     FTCH     ,NULL
        ADCZL   0.0      ,-2
        LDA     1.ADRID  ,ID ADDRESS
        JSR     GETN     ,GET THE ID
        JMP     BADC     ,BAD CHARACTER
        JMP     FTCH     ,NO ID SPECIFIED
        JMP     BADC     ,PERIOD
FTCH:   LDA     1.@ADRID,GET THE ID
        SUBZR   0.0      ,GENERATE 100000
        LDA     2.ID     ,GET THE DEFAULT ID
        SUB#    0.1.SNR  ,WAS THERE AN ID?
        STA     2.@ADRID,NO. USE THE DEFAULT
        LDA     0.C40    ,NON RESIDENT DMS SECTOR
        STA     0.@LASTP
        JSR     @FETCH   ,IS THE FILE THERE?
        MOV     0.0      ,NO
        SUB     0.0.SKP  ,NO
        SUBZL   0.0      ,YES
        JMP     @SU3     ,RETURN
BADC:   ADC     0.0      ,GENERATE A -1
        JMP     @SU3     ,RETURN
RESET:  LDA     2.WORKF  ,ADDRESS OF THE WORKFILE TABLE
        LDA     1.-1.2   ,GET THE COUNTER (-8)
        SUBZR   0.0      ,RESET WORD
        STA     0.0.2    ,STORE THE WORD
        INC     2.2      ,BUMP THE POINTER
        INC     1.1.SZR  ,ARE WE DONE?
        JMP     -3       ,NO. DO THE NEXT WORD
        JMP     0.3      ,RETURN
SU3:    0                ,RETURN ADDRESS
RTURN   0                ,SECONDARY RETURN ADDRESS
NG:     -6
C40:    40
C56:    56
C177:   177
ID:     0
DMS     TOP-756
WORKF:  TOP-351
TIOUT:  TOP-56
ADRID:  TOP-342
FETCH:  TOP-317
LASTP:  TOP-320
ADNAM:  0
BCTR:   0
COUNT:  0
DEST:   0
LDB:    .LDB
GETN:   STA     0.BCTR   ,SAVE THE BYTE COUNT
        SUB     0.0      ,INITIALIZE BYTE COUNT
        STA     0.COUNT  ,TO ZERO
        MOVZL   1.1      ,GET THE BYTE ADDRESS OF THE
        STA     1.DEST   ,DESTINATION STRING
        STA     3.RTURN  ,SAVE THE RETURN ADDRESS
NEXTC:  LDA     0.ADNAM  ,GET THE NAME ADDRESS
        JSR     @LDB     ,GET A BYTE
        INC     0.0
        STA     0.ADNAM  ,BUMP THE STRING POINTER
        MOV     1.1.SNR  ,IS IT A NULL?
        JMP     NULLC    ,YES. CHECK IT OUT
        LDA     0.C177   ,RUB OUT
        AND     0.1      ,GET RID OF THE PARITY BIT
        SUB#    0.1.SNR  ,IS IT A RUB OUT?
        JMP     @RTURN   ,YES  ITS A BAD CHRACTER
        LDA     0.C56    ,PERIOD
        SUB#    0.1.SNR  ,IS IT A PERIOD?
        JMP     PERD     ,YES. RETURN
        LDA     0.C40    ,SPACE
        SUB#    1.0.SNR  ,IS IT A SPACE?
        JMP     NULLC    ,YES. CHECK IT OUT
        SUBZ#   1.0.SZC  ,IS IT A CONTROL CHARACTER?
        JMP     @RTURN   ,YES. ITS A BAD CHARACTER
        LDA     0.BCTR   ,GET THE BYTE COUNTER
        MOV     0.0.SNR  ,SAVE THE BYTE?
        JMP     NEXTC    ,NO. IGNORE IT
        LDA     3.DEST   ,GET THE DESTINATION ADDRESS
        MOVZR   3.3.SZC  ,GET THE WORD ADDRESS
        JMP     ODDCH    ,ITS AN ODD CHARACTER
        MOVS    1.1      ,SWAP BYTES
        STA     1.0.3    ,STORE IT
        MOVOL   3.3      ,BUMP THE BYTE ADDESS
        JMP     LOPIT
ODDCH:  LDA     0.0.3    ,GET THE PREVIOUS CHARACTER
        ADDS    0.1      ,COMBINE THE BYTES
        STA     1.0.3    ,STORE IT
        INCZL   3.3      ,BUMP THE BYTE POINTER
LOPIT   STA     3.DEST   ,SAVE THE BYTE POINTER
        ISZ     COUNT    ,BUMP THE BYTE COUNT
        ISZ     BCTR     ,ARE WE DONE?
        JMP     NEXTC    ,NO. GET ANOTHER CHARACTER
        JMP     NEXTC    ,YES. IGNORE THE REST OF THE CHARACTERS
NULLC:  LDA     0.COUNT  ,GET THE BYTE COUNT
        LDA     3.RTURN  ,GET THE RETURN ADDRESS
        MOV     0.0.SZR  ,ANY BYTES READ IN?
        JMP     3.3      ,YES. TAKE THE NORMAL RETURN
        JMP     1.3      ,NO. TAKE ERROR RETURN
PERD:   LDA     3.RTURN  ,GET THE RETURN ADDRESS
        JMP     2.3      ,ERROR RETURN
        END

<>
```

```
<>PR PAGE.FS
C       PAGE    (CALL 2)
C
C
C       programmed by JOE FOGLER
C
C       APRIL  1977
C
C
C       PAGE.BELL, & COPY CONTROL
C
C       1ST PARAMETER
C
C       1       BELL
C       2       PAGE
C       3       PAGE THEN BELL
C       4       COPY
C       5       COPY THEN BELL
C       6       COPY THEN PAGE THEN BELL
C
C
        SUBROUTINE      PAGE(NUM)
        INTEGER NUM,FLAG
        COMMON /PL/ FLAG
C       DEFAULT=1       (BELL)
        IF (NUM.LT.1.OR.NUM.GT.6) NUM=1
C       OUTPUT CONTROL CODES
        IF (NUM.EQ.1) GO TO 300
        IF (NUM.LT.4) GO TO 100
        CALL PCHAR(27)
        CALL PCHAR(23)
C       DELAY 20 SECONDS FOR COPY
        CALL DELAY(200)
100     IF (NUM.EQ.4.OR.NUM.EQ.5) GO TO 200
        FLAG=0
        CALL PCHAR(27)
        CALL PCHAR(12)
200     IF (NUM.EQ.2.OR.NUM.EQ.4) GO TO 400
300     CALL PCHAR(7)
400     RETURN
        END

<>
```

```
<>PR INIT.FS
C       INIT    (CALL 3)
C
C
C       programmed by JOE FOGLER
C
C       APRIL 1977
C
C
C       INITIALIZE MODE AND POSITION
C
C       1ST PARAMETER
C
C       0X      NO-OP
C       1X      RESET PAGE
C
C       X1      ALPHA MODE
C       X2      GRAPHIC MODE
C       X3      SPECIAL POINT PLOT MODE
C       X4      INCREMENTAL MODE
C       X5      POINT PLOT MODE
C
C       2ND PARAMETER
C
C       0-1023  X POSITION
C
C       3RD PARAMETER
C
C       0-767   Y POSITION
C
C
        SUBROUTINE INIT(NUM,X,Y)
        INTEGER NUM,SCRN,MODE,FLAG,X,Y
        COMMON /PL/ FLAG
        FLAG=0
C       EXTRACT TENS AND ONES DIGITS
        SCRN=NUM/10
        MODE=NUM-10*SCRN
        IF (SCRN.NE.1) GO TO 90
        CALL PCHAR(27)
        CALL PCHAR(12)
90      CALL PCHAR(13)
        CALL PCHAR(29)
        CALL SEND(0,X,Y)
        IF (MODE.LT.1.OR.MODE.GT.5) MODE=1
        GO TO (100,600,300,400,500) MODE
100     CALL PCHAR(31)
        GO TO 600
300     CALL PCHAR(27)
        CALL PCHAR(28)
        GO TO 600
400     CALL PCHAR(30)
        GO TO 600
500     CALL PCHAR(28)
600     RETURN
        END
```

```
<>PR VECT.FS
C       VECT    (CALL 5)
C
C
C       programmed by JOE FOGLER
C
C       APRIL  1977
C
C
C       VECTOR PLOT
C
C       1ST PARAMETER
C
C       0       LOW RESOLUTION
C       1       HIGH RESOLUTION
C
C       2ND PARAMETER
C
C       1X      NORMAL MODE
C       2X      DEFOCUSED MODE
C       3X      WRITE THRU MODE
C
C       X0      NORMAL LINE
C       X1      DOTTED LINE
C       X2      DOT-DASHED LINE
C       X3      DOT-DASHED LINE
C       X4      SHORT-DASHED LINE
C       X5      LONG-DASHED LINE
C
C       3RD PARAMETER
C
C       X       POSITION
C
C       4TH PARAMETER
C
C       Y       POSITION
C
        SUBROUTINE VECT(RESO,NUM,X,Y)
        INTEGER RESO,NUM,MODE,LTYP,X,Y,CMD
        CMD=96
C       EXTRACT TENS AND ONES DIGITS
        MODE=NUM/10
        LTYP=NUM-10*MODE
        IF (MODE.LT.1.OR.MODE.GT.3) MODE=1
        IF (LTYP.LT.0.OR.LTYP.GT.4) LTYP=0
        CMD=CMD+LTYP+2*(MODE-1)
        CALL PCHAR(27)
        CALL PCHAR(CMD)
        CALL SEND(RESO,X,Y)
```

```
<>PR POINT.FS
C       POINT   (CALL 6)
C
C
C       programmed by JOE FOGLER
C
C       AFRIL  1977
C
C
C
C       GREY SCALED POINT PLOT
C
C       1ST PARAMETER
C
C       RESO    0       LOW RESOLUTION
C               1       HIGH RESOLUTION
C
C       2ND PARAMETER
C
C       INTN    0-100   INTENSITY
C
C       3RD PARAMETER
C
C       X       0-1023  (LOW RESOLUTION)
C               0-4095  (HIGH RESOLUTION)
C
C       Y       0-767   (LOW RESOLUTON)
C               0-3071  (HIGH RESOLUTON)
C
C
        SUBROUTINE POINT(RESO,INTN,X,Y)
        INTEGER RESO,INTN,X,Y
        INTEGER BRIT
        IF (RESO.NE.1) RESO=0
        IF (INTN.LT.0) INTN=0
        IF (INTN.GT.100) INTN=100
        BRIT=64+31*INTN/100
        CALL PCHAR(BRIT)
        CALL SEND(RESO,X,Y)
        RETURN
        END

<>
```

```
<>PR PCHAR.FS
;**********************************************
;*****        PUT CHARACTER ROUTINE      *****
;**********************************************
;
;
;programmed by JOE FOGLER
;
;APRIL  1977
;
;
;
        .TITL   PCHAR
        .ENT    PCHAR
        .DUSR    SYSTM=6017
        .DUSR    .PCHAR=20B7
        .NREL
        .TXTM   1
        .EXTU
        .EXTN   .I
.F1:
.F2:    0
        .CSIZ   0
        FS.
PCHAR:
        JSR     @.CPYL
        JMP     @.+1
        L1.
L1.:
        LDA     0,@T.+0.3       ;CHAR
        SYSTM                   ;OUTPUT CHARACTER
        .PCHAR                  ;TO THE TTY
        JMP     .+1
        JSR     @ FRET
        JSR     @.FRET
                FS.=1
        SFS.=0
        T.=-167
        V.=200+T.
        TS.=T.+0
        FTS.=T.+0
        VS.=V.+0
        FVS.=V.+0
        .END

<>
```

```
<>PR GCHAR.FS
;**********************************************
;*****        GET CHARACTER ROUTINE      *****
;**********************************************
;
;
;programmed by JOE FOGLER
;
;APRIL  1977
;
;
        .TITL   GCHAR
        .ENT    GCHAR
        .DUSR    SYSTM=6017
        .DUSR   .GCHAR=17B7
        .NREL
        .TXTM   1
        .EXTU
        .EXTN   .I
.F1:
.F2:    0
        .CSIZ   0
        FS.
GCHAR:
        JSR     @.CPYL
        JMP     @.+1
        L1.
L1.:
        STA     3,SV3           ;SAVE POINTER
        SYSTM                   ;GET CHARACTER FROM TTY
        .GCHAR
        JMP     .+1
        LDA     3,SV3           ;RECALL POINTER
        STA     0,@T.+0.3       ;STORE CHARACTER
        JSR     @ FRET
        JSR     @ FRET
                FS.=1
        SFS.=0
        T.=-167
        V.=200+T.
        TS.=T.+0
        FTS.=T.+0
        VS.=V.+0
        FVS.=V.+0
SV3:    0
        .END

<>
```

```
<>PR SEND.FS
;**************************************************************
;*****            FORTRAN/ASSEMBLY LINKAGE               *****
;**************************************************************
;
;
;programmed by JOE FOGLER
;
;APRIL 1977
;
;
        .NREL
        .TITL   SEND
        .ENT    SEND
        .NREL
        .TXTM   1
        .EXTU
        .EXTN   .I
.F1:
.F2:    0
        .CSIZ   0
        FS.
SEND:
        JSR     @ CPYL
        JMP     @.+1
        L1.
L1.:
;**************************************************************
;
        LDA     -2,@T.+0,3              ;GET RESOLUTION
        LDA     0,@T.+1,3               ;GET X VALUE
        LDA     1,@T.+2,3               ;GET Y VALUE
        JSR     OUT                     ;OUTPUT HIY,LOY,ETC..
;
;**************************************************************
        JSR     @ FRET
        JSR     @ FRET
                FS.=3
        SFS =0
        T.=-167
        V.=200+T.
        TS.=T.+2
        FTS.=T.+0
        VS.=V.+2
        FVS.=V +0

;**************************************************************
;*****     ROUTINE TO FORM HIY,LOY,HIX,LOX, etc          *****
;**************************************************************
;
        DUSR    SYSTM=6017
        .DUSR   .PCHAR=20R7
;INPUTS:        AC0     X-VALUE
;               AC1     Y-VALUE
;               AC2     RESOLUTION
;

OUT:    STA     3,SV3           ;SAVE RETURN ADDRESS
        MOVL#   0,0,SZC         ;IF X VALUE < 0
        SUB     0,0             ;DEFAULT TO ZERO
        MOVL#   1,1,SZC         ;IF Y VALUE < 0
        SUB     1,1             ;DEFAULT TO ZERO
        NEG     2,3             ;MAKE NEGATIVE
        COM     3,3,SZR         ;SKIP IF IT WAS A 1
        STA     2,RESO          ;SAVE RESOLUTION
        LDA     3,C7777         ;(MAX ALLOWABLE X)
        MOV     2,2,SNR         ;SKIP FOR HIGH RESOLUTION
        ADDZL   0,0,SNC         ;X=X*4
        SUBZ#   0,3,SNC         ;IF X > 7777
        MOV     3,0             ;DEFAULT = 7777
        LDA     3,C5777         ;(MAX ALLOWABLE Y)
        MOV     2,2,SNR         ;SKIP FOR HIGH RESOLUTION
        ADDZL   1,1,SNC         ;Y=Y*4
        SUBZ#   1,3,SNC         ;IF Y > 5777
        MOV     3,1             ;DEFAULT = 5777
        MOV     2,2,SNR         ;BYPASS IF
        JMP     FORM            ;IN LOW RESOLUTION
;EXTRACT THE EXTRA BYTE
        LDA     2,C3            ;2 BIT MASK
        MOV     2,3             ;COPY INTO AC3
        AND     0,2             ;2 LSB'S OF X
        AND     1,3             ;2 LSB'S OF Y
        ADDZL   3,3             ;SHIFT LEFT 2 BITS
        ADD     3,2             ;FORM 4 X & Y BITS
        LDA     3,C140          ;ADD THE
        ADD     3,2             ;TAG BITS
        STA     2,EXTB          ;SAVE THE EXTRA BYTE
FORM:   STA     0,SUX           ;SAVE X
        STA     1,SUY           ;SAVE Y
        LDA     3,C37           ;5 BIT MASK
        LDA     2,C40           ;TAG BITS
        MOVZL   1,0             ;ALIGN HIY IN
        MOVS    0,0             ;RIGHT BYTE
        AND     3,0             ;MASK FOR 5 BITS
        ADD     2,0             ;ADD TAG BITS
         SYSTM                  ;OUTPUT HIY
        .PCHAR                  ;TO THE TELETYPE
        JMP     .+1             ;ERRTN
        DSZ     RESO            ;IF IN LOW
        JMP     .+5             ;RESOLUTION, BYPASS
        LDA     0,EXTB          ;RECALL EXTRA BYTE
        .SYSTM                  ;OTPUT EXTRA BYTE
         PCHAR                  ;TO THE TELETYPE
        JMP     .+1             ;ERRTN
        LDA     0,SUY           ;RECALL Y
        LDA     3,C37           ;5 BIT MASK
        LDA     2,C140          ;TAG BITS
        MOVZR   0,0             ;ALIGN LOY IN
        MOVZR   0,0             ;RIGHT BYTE
        AND     3,0             ;MASK FOR 5 BITS
        ADD     2,0             ;ADD TAG BITS
         SYSTM                  ;OUTPUT LOY
         PCHAR                  ;TO THE TELETYPE
        JMP     .+1             ;ERRTN
        LDA     0,SUX           ;RECALL X
```

```
        LDA     3,C37           ;5 BIT MASK
        LDA     2,C40           ;TAG BITS
        MOVZL   0,0             ;ALIGN HIX IN
        MOVS    0,0             ;RIGHT BYTE
        AND     3,0             ;MASK FOR 5 BITS
        ADD     2,0             ;ADD TAG BITS
        .SYSTM                  ;OUTPUT HIX TO
        .PCHAR                  ;THE TELETYPE
        JMP     .+1             ;ERRTN
        LDA     0,SVX           ;RECALL X
        LDA     3,C37           ;5 BIT MASK
        LDA     2,C100          ;TAG BITS
        MOVZR   0,0             ;ALIGN LOX IN
        MOVZR   0,0             ;RIGHT BYTE
        AND     3,0             ;MASK FOR 5 BITS
        ADD     2,0             ;ADD TAG BITS
        .SYSTM                  ;OUTPUT HIX TO
        .PCHAR                  ;TO THE TELETYPE
        JMP     .+1             ;ERRTN
        JMP     @SV3            ;RETURN
SV3:    0
RESO    0
SVX:    0
SVY:    0
EXTB:   0
C5777:  5777
C7777:  7777
C3:     3
C37:    37
C40:    40
C100:   100
C140:   140
        .END
```

OSEND.FS , continued

```
<>PR RECU.FS
;*****************************************************
;*****          FORTRAN/ASSEMBLY LINKAGE         *****
;*****************************************************
;
;programmed by JOE FOGLER
;
;APRIL 1977
;
        .NREL
        .COMM   PL      1       ;COMMON /PL/ FLAG
        .TITL   RECU
        .ENT    RECU
        NREL
        TXTM    1
        .EXTU
        .EXTN   .I
.F1:
.F2:    0
        .CSIZ   0
        FS
RECU:
        JSR     @ CPYL
        JSR     @.FINI
        .F1
        .F2
        JMP     @.+1
        L1.
L1.:
        JSR@    LD0
        .GADD   PL.0            ;FLAG
        STA     3,SV3.
        JSR     RECU.
        LDA     3,SV3.
        STA     2,@T.+0.3
        STA     0,@T.+1.3
        STA     1,@T.+2.3
        JSR     @ FRET
        JSR     @.FRET
                FS.=3
        SFS.=0
        T.=-167
        V.=200+T.
        TS.=T.+2
        FTS =T.+0
        VS.=V.+2
        FVS =V.+0
SV3.:   0
;
;*****************************************************
;*****          GET STATUS ROUTINE              *****
;*****************************************************
;
        DUSR    SYSTM=6017
        .DUSR   GCHAR=17B7
;
;INPUTS.        AC0     FLAG
;
;                       0       EXPECT 5 BYTES
;                       1       EXPECT 4 BYTES
;
;OUTPUTS                AC0     X
;                       AC1     Y
;                       AC2     STATUS BYTE OR KEYSTROKE
;
RECU.:  STA     3,SV3           ;SAVE RETURN ADDRESS
        ADC     3,3             ;-1
        ADD     3,0,SNR         ;IF FLAG=1,
        JMP     BYPA.           ;SKIP FIRST CHARACTER
        .SYSTM                  ;GET FIRST CHARACTER
        .GCHAR                  ;FROM TELETYPE
        JMP     .+1
BYPA.:  STA     0,BYTE          ;SAVE IT
        .SYSTM                  ;GET HIX FROM THE TTY
        .GCHAR
        JMP     .+1
        LDA     1,C37           ;5 BIT MASK
        ANDZL   1,0             ;REMOVE TAG BITS
        ADDZL   0,0             ;SHIFT LEFT 5
        ADDZL   0,0
        STA     0,X             ;SAVE IT
        .SYSTM                  ;GET LOX FROM THE TTY
        .GCHAR
        JMP     .+1
        LDA     1,C37           ;5 BIT MASK
        AND     1,0             ;REMOVE TAG BITS
        LDA     1,X             ;RECALL HIX
        ADD     1,0
        STA     0,X             ;HAVE X ADDPESS
        .SYSTM                  ;GET HIY FROM THE TTY
        .GCHAR
        JMP     .+1
        LDA     1,C37           ;5 BIT MASK
        ANDZL   1,0             ;REMOVE TAG BITS
        ADDZL   0,0             ;SHIFT LEFT 5
        ADDZL   0,0
        STA     0,Y             ;SAVE IT
        .SYSTM                  ;GET LOY FROM THE TTY
        .GCHAR
        JMP     .+1
        LDA     1,C37           ;5 BIT MASK
        AND     1,0             ;PEMOVE TAG BITS
        LDA     1,Y             ;RECALL HIY
        ADD     0,1             ;FORM Y IN AC1
        LDA     0,X             ;GET X IN AC0
        LDA     2,BYTE          ;GET FIRST CHARACTER
        JMP     @SV3            ;RETURN
SV3:    0
BYTE:   0
X:      0
Y:      0
C37:    37
        .END
<>
```

References

1. Koenig, W., Dunn, H., and Lacey, L., "The Sound Spectrograph", J. Acoust. Soc. Am., vol. 18, pp.19-49, 1946.

2. Morris, L. R., "Fast Speech Spectrogram Reduction and Display on Minicomputer/Graphics Processors", IEEE Trans. Acoust. Speech, and Sig. Proc., pp.297-300, June 1975.

3. Cooley, J. W., and Tukey, J. W., "An Algorithm for the Machine Computation of Complex Fourier Series", Math. Comput., vol. 19, pp.297-301, April 1965.

4. Kahaner, D. K., "Matrix Description of the Fast Fourier Transform", IEEE Trans Audio and Electroacoustics pp.442-452, August 1970.

5. Ahmed, N., Natarajan, T., and Rainbolt, H. R., "On Generating Walsh Spectrograms", IEEE Trans. Electromagn. Compat. pp.198-200, November 1976.

6. Nolan, M. Helena, "Some Consideratins of Deaf Speech", Master's Report, Kansas State Univ., 1974.

7. Spiegel, M. R., Mathematical Handbook of Formulas and Tables, New York: McGraw-Hill, 1968.

8. Ahmed, N. and Rao, K. R., Orthogonal Transforms for Digital Signal Processing, Berlin or New York: Springer-Verlag, 1975.

9. Oppenheim, A. V., "Speech Spectrograms Using the Fast Fourier Transform", IEEE Spectrum, vol. 7, pp.57-62, August, 1970.

10. Flanagan, J. L., Speech Analysis, Synthesis and Perception, New York: Springer-Verlag, 1972.

SPECTROGRAM GENERATION WITH A MINICOMPUTER AND A GRAPHICS TERMINAL

by

RONALD DALE SAUDER

B. S., Kansas State University, 1975

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1977

Abstract

This paper presents a FORTRAN program for the generation of spectrograms. The program was developed on a NOVA mini-computer with a Tektronix graphics terminal and copier. Only speech data was processed, although any type data could be used. Since the program works with digitized data on disk, any signal which is to be processed must first be digitized and stored on disk. An A/D converter interfaced with the NOVA makes this a relatively easy task.

In processing the data, several factors are considered. The effect of the sampling window is explained in detail. It is demonstrated how a rectangular sampling window can distort the power spectrum, while a Hanning cosine window lessens this distortion. The trade-offs between wide-band and narrow-band spectrograms are discussed in terms of frequency and time resolution. The dynamic range of a spectrogram and its sensitivity to power level changes are also considered.

A technique for displaying the spectral points with good contrast between power levels is presented. Clusters or arrays of points are plotted with variable intensity, which produces displays having excellent contrast, although much of it is lost by the copier.

Several example spectrograms were made, demonstrating the aspects that were discussed. A test display is given which demonstrates the contrast between power levels and gives the algorithm presented here credibility. Suggestions are made to improve the speed of this algorithm, which now requires over one hour to process one second of speech. Program listings and several descriptive figures are included.