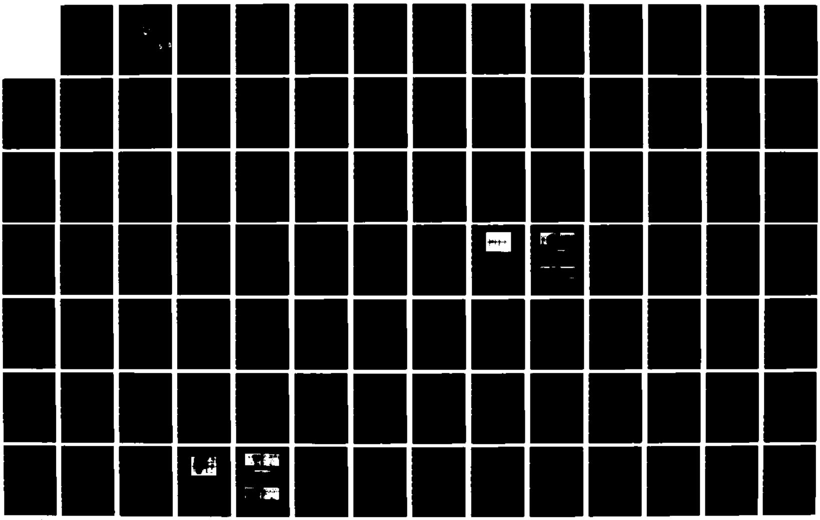
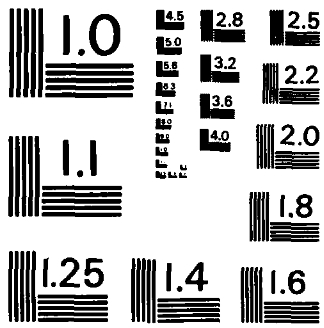
AD-A158 960 A MATCHED FILTER ALGORITHM FOR ACOUSTIC SIGNAL DETECTION(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D W JORDAN JUN 85 1/3

UNCLASSIFIED F/G 17/1 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

A MATCHED FILTER ALGORITHM FOR
ACOUSTIC SIGNAL DETECTION

by

Dorsett Weston Jordan

June 1985

Thesis Advisor: R. Panholzer

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
A Matched Filter Algorithm for Acoustic Signal Detection

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis;
June 1985

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Dorsett Weston Jordan

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943-5100

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943-5100

12. REPORT DATE
June 1985

13. NUMBER OF PAGES
221

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 is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Analog Filter; Digital Filter; Bilinear Transform; Space Shuttle; Auxiliary Power Unit; INTEL 2920 Analog Signal Processor; Adaptive LInear Combiner; Adpative Transversal Filter

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This thesis is a presentation of several alternative acoustic filter designs which allow Space Shuttle payload experiment initiation prior to launch. This initiation is accomplished independently of any spacecraft services by means of a matched band-pass filter tuned to the acoustic signal characteristic of the Auxiliary Power Unit (APU) which is brought up to operating RPMs approximately five minutes prior to launch.

These alternative designs include an analog filter built around operational amplifiers, a digital IIR design implemented with an INTEL 2920 Signal Processor, and an Adaptive FIR Weiner design. Working prototypes of the first two filters are developed and a discussion of the advantage of the 2920 digital design is presented.

19

A Matched Filter Algorithm for Acoustic Signal Detection

by

Dorsett Weston Jordan
Lieutenant, United States Navy
B.S., University of Colorado, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June, 1985

Author: D. W. Jordan

Approved by: Rudolf Panholzer, Thesis Advisor

Sydney Parker, Second Reader

Harriett Rigas, Chairman, Department of Electrical and Computer Engineering

John Dyer, Dean of Science and Engineering

ABSTRACT

This thesis is a presentation of several alternative acoustic filter designs which allow Space Shuttle payload experiment initiation prior to launch. This initiation is accomplished independently of any spacecraft services by means of a matched band-pass filter tuned to the acoustic signal characteristic of the Auxiliary Power Unit (APU) which is brought up to operating RPM's approximately five minutes prior to launch.

These alternative designs include an analog filter built around operational amplifiers, a digital IIR design implemented with an INTEL 2920 Signal Processor, and an Adaptive FIR Weiner design. Working prototypes of the first two filters are developed and a discussion of the advantage of the 2920 digital design is presented.

TABLE OF CONTENTS

ACKNOWLEDGMENT

Many persons aided me in the development of this thesis and I would be remiss in not acknowledging the help and support of those whose contributions were invaluable.

Vicente Garcia offered early insight and suggested avenues to a solution. Bob Yamamuro of Aerospace Corporation assisted me in the initial data collection and unselfishly spent several evenings of his own time in support of my efforts. Austin Boyd gave assistance with crucial data analysis during a difficult time in his life. Professor Rudy Panholzer was a talented teacher and supportive advisor throughout my stay at NPS. Dr. Bharat Madan was indispensable to me as an expert in digital filters and all aspects of the 2920 design implementation. He gave me many hours of wise counsel. David Rigmaiden was and is a superb and supremely patient lab technician. Thanks to all the NPS Get Away Special team members whose motivation and spirit kept me honest. And finally thanks and love to Linda Duncan-Hille who buoyed my spirits at all the right moments with her abundant joie de vivre.

I am especially grateful for the help that each and every one of you gave me during this research.

I. INTRODUCTION

A. BACKGROUND

In April of 1981 a new era of space exploration was opened for mankind when the Space Shuttle successfully returned to Earth after its first orbital mission. Notwithstanding the significant capabilities which this new and radically different concept in space transportation has afforded traditionally large and well funded government and industrial agencies, the Space Shuttle has also ushered in the age of the small experimenter with limited access to funds who has an idea which needs to be tested in space and who now has a means of seeing that test realized.

The Get Away Special (GAS) Program was developed by the National Aeronautics and Space Administration (NASA) to afford qualifying teams of experimenters the opportunity to launch and orbit GAS payloads on a not-to-interfere space available basis during regularly scheduled Shuttle missions.

The main purpose for Shuttle missions is the conveyance into space and deployment of large, sophisticated instruments into Earth orbit. Because many primary mission payloads do not occupy the entirety of the Shuttle payload bay there is frequently opportunity for the additional deployment of GAS payloads during Shuttle missions [Ref. 1: pp. 8-9].

One significant requirement demanded of GAS experimenters is that each payload be self-contained within its standard size (5.0 cubic foot) NASA provided canister. Thus it is expected that each GAS experiment include provisions within its own confines for all electrical power, heating elements, data control and storage, and so on. In short, GAS payloads may not draw upon any Shuttle services for their normal operation beyond an on-off switch control which may be thrown locally by an astronaut should the Shuttle timeline of events allow for such intervention.

This requirement for a completely self-contained experiment necessitates a well-planned execution of tasks performed under automated control. Although the concession for astronaut involvement in experiment initiation is provided, during some critical periods it is not feasible for astronauts to tend to GAS payload requirements, e. g., during launch and landing phases. At these times it then becomes necessary to design for complete automation of GAS experimental control, to provide most significantly for independent initiation of experimentation.

B. STATEMENT OF GOAL

It is the purpose of this thesis to consider a design algorithm to accomplish an automated initiation of GAS experimentation based solely upon the passive detection of a well-defined event in the evolution of the Shuttle timeline of operations. Specifically it is desired that an experiment be undertaken to measure the acoustical and vibrational environment present in the Shuttle payload bay from prior to Space Shuttle Main Engine (SSME) ignition (at approximately T-6 seconds) until the Orbiter has exited the Earth's atmosphere (at approximately T+2 minutes).

Previous GAS missions have been deployed to accomplish this same mission goal. In these previous missions the solution to the problem of independent experiment initiation was addressed by way of a simple sound pressure level (SPL) sensor used to detect the unmistakable roar of SSME's as they ignite at T-6 seconds. By this method the sizable impulse thus generated can easily be used to signal experiment initiation. In this way power is applied to data collection and storage elements whose purpose is to record the information relevant to the mission of the experiment.

The obvious shortcoming in this experimental scheme is that it uses as its prime mover the very noise and vibration which it presumes to measure. Though the delay caused by impulse generation and transmission via the SPL sensor may be minimal, it nonetheless requires a finite amount of time for tape transport transients to settle and begin the recording of meaningful data. This delay has been reported to be as much as several seconds and so can be seen to be a

Therefore our task is to develop a matched filter which will detect the presence of a spectral peak at 600 Hz and/or at coincident harmonic multiples of 600 Hz. We will attempt to realize the needed performance emphasizing only the most evident 600 Hz peak. Based upon the results of this design, will then know if we must consider the additional impact of the less evident harmonic elements.

Our desire is to implement the scheme which provides the simplest, smallest, most reliable, and least power consuming design which will ensure APU detection prior to launch. Therefore detection of all higher order harmonic components is less important than the development of a filter which accomplishes the primary goal. We shall examine both analog and digital methods for implementing our matched filter and compare the effectiveness of the various designs before selecting the device which will orbit the Earth.

but for our purposes this is an inconsequential shortcoming.)

An examination of the PSD's of the remaining sets of data not corrupted by the lack of TSC reveals the expected behavior. In all cases the noise background remains essentially unchanged from the first of the plot pair to the second except for the effect of the impressed APU signature clearly evident in the second. In every plot performed after APU start-up the characteristic 600 Hz peak is evident to some degree and in many instances we also see the harmonic components at 1200 and 1800 Hz. This is especially true for those plots reflecting the environment surrounding microphone 9403 which is located only a bulkhead removed from the APU's themselves. However, of some concern is the fact that the 600 Hz peak is least evident for microphone 9405 which is at the forward bulkhead furthest removed from the APU's and where it is expected that the GAS canister will be placed for the upcoming mission.

The salient points are therefore as follows:

1. The APU does provide a specific signature which becomes clearly evident in the audio spectrum of the Shuttle payload bay pre-launch acoustic environment. There are well-defined spectral peaks at a fundamental frequency of 600 Hz and at integral harmonics thereof.

2. The magnitude of the APU signature is variable in the payload bay depending upon the location of the sensor used to detect it. A sensor placed closer to the after bulkhead will be more apt to respond to the APU signature in a manner which will facilitate matched filter performance, but the signature is evident throughout the payload bay.

it is known that measures were taken to minimize the error introduced in the dubbing process. On most copies a Tape Speed Compensation (TSC) process was employed which ensures that the tape transport travels at the same speed as when originally recorded. That this process is indispensable is realized upon examination of the PSD's obtained for microphones 9405 and 9219 on STS-3. In these two cases alone TSC was not employed due to operator error at a previous generation. The noise floor thus generated is seen to be an order of magnitude greater than in other plots and is severe enough to mask the desired information. In all other tape copies TSC was employed.

(Another generation of tape copies was recorded for use at the Naval Postgraduate School in the development of the experiment described earlier. This recording process was accomplished using a Hewlett-Packard Model 3964A Instrumentation Data Recorder in an FM mode to allow recording of analog data from 0 to 8 kHz. The HP recorder also employs a tape speed servo control which also ensures tape speed accuracy. PSD plots were obtained in all cases for the dubbed tape to verify the accuracy of this latest dubbing routine. In each case the PSD of the dubbed data was confirmed to be a good reproduction of the "original" except for the region below 200 Hz. In this region the reproduction was not as good as in the region above 200 Hz

Appendix B is a summary of PSD plots of the acoustic environment present in the Shuttle payload bay at three different locations for the three missions listed in the previous section. As noted in Figure 2.2 the locations of the three microphones are dispersed throughout the payload bay to reveal variations in the acoustic signature from one location to another. Microphone V08Y9405A (which shall be referred to as 9405 for simplicity) was positioned near the forward bulkhead of the payload bay, furthest removed from the APU. Microphone V08Y9219A (9219) was located low and amidships, and microphone V08Y9403A (9403) was located at the after bulkhead, closest of the three to the APU. All three were identically configured to respond to a dynamic range of 110-157 dB.

The payload bay acoustic data is presented in pairs of plots with the first of the pair showing the signature before the APU is turned on and the second showing the signature approximately thirty seconds later after APU start-up. This data was obtained from an analog magnetic tape copy of the original data recording which was made available from the DATE group at Aerospace Corporation headquarters in Los Angeles.

There is doubtless some extraneous noise introduced in the process by which the original tape was copied to yield the tape available at Aerospace. Furthermore, the generation of the Aerospace copies are not known. However

Table 2.1

DEVELOPMENT FLIGHT INSTRUMENTATION (DFI)
ACOUSTIC MEASUREMENT INFORMATION

Microphone Number	Range (dB)	Cargo Bay Location	Orbiter Station Location X	Y	Z
VO8Y9219A	110-157	Internal	863	-100	381
VO8Y9220A	110-157	Internal	1190	0	427
VO8Y9401A	130-177	External	639	3.5	500
VO8Y9402A	130-177	External	1281	4.2	500
VO8Y9403A	110-157	Internal	1306	12.0	400
VO8Y9404A	130-177	External	1296	0	300
VO8Y9405A	110-157	Internal	640	4	423

Note: The frequency response of all microphones is wide-band, 20 Hz to 8 kHz.

D. STS PAYLOAD BAY PRE-LAUNCH ACOUSTIC ENVIRONMENT

As indicated in the previous section we are principally interested in the response of three internal microphones to the Shuttle payload bay environment. The intent is to examine the environment prior to APU start-up to determine the noise background present before the APU acoustic signature is impressed upon this background. Then microphone responses will be examined after APU start-up to reveal the APU acoustic signature over the noise background. Based upon our previous examination of the APU vibrational signature evident in equipment testing we expect to see that spectral peaks at the fundamental and harmonic frequencies of 600 Hz will become clear at the time of APU start-up.

at launch. Rounding out the array of acoustic sensors were three additional microphones mounted externally. All microphones were manufactured by Gulton Industries with only minor alterations differentiating the three external sensors from the four internal ones. The relative location of each of these seven acoustic sensors is shown in Figure 2-2.

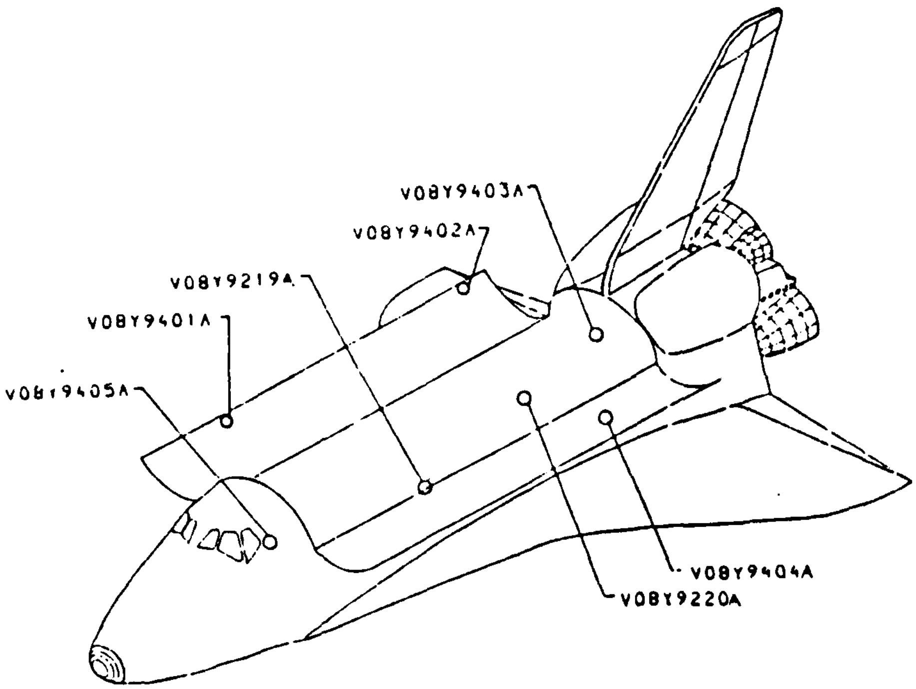


Figure 2.2 DFI Acoustic Measurement Locations

Sensor location designators, dynamic range and frequency response of each microphone are shown in Table 2-1 on the following page [Refs. 3 and 4].

C. STS PAYLOAD BAY ACOUSTIC ENVIRONMENT MEASUREMENT INSTRUMENTATION

Data instrumentation recorders were flown on all early Shuttle flights to measure the acoustic environment present during the launch and landing phases of these missions. These experiments were conducted under the auspices of the NASA DATE (Dynamic, Acoustic and Thermal Environments) Working Group which has as its mission the development of improved methods for predicting all aspects of STS (Shuttle Transportation System) payload environments. Pursuant to this study we shall make use of data collected during each of the following three Shuttle missions:

-- STS-2 (conducted November 12-17, 1981)

-- STS-3 (conducted March 22-27, 1982)

-- STS-4 (conducted June 27 - July 4, 1982)

In each of these missions it is only the data which was recorded just prior to the launch which is of any significance for our purposes here.

Of particular interest to us is the acoustic data recorded at three specific locations in the Shuttle payload bay on each of the three flights listed above. On each flight sixteen selected sensors comprised the STS Development Flight Instrumentation (DFI) system. Of these sixteen sensors, four internal microphones were used to measure the acoustic environment present in the payload bay of higher spectral components although an expected 2400 Hz peak is present in several plots despite the attenuation.

The significance of these plots is in the consistency of the component spectral elements despite APU loading. Although the relative and absolute magnitudes of the fundamental 600 Hz peak and its harmonic constituents vary somewhat over the range of loading displayed the spectral location of each remains fixed. This shift in magnitudes is of some concern to mechanical engineers as it has been shown that failures in some rotating machinery have occurred when vibration signatures have deviated in this manner. But our concern is with the consistency of the location of the spectral peaks as this confirms the RPM stability of the APU over expected levels of loading.

We thus have a basis for further investigation of the acoustic environment in the Shuttle payload bay. We know that there is a specific vibrational signature which accompanies the normal functioning of the APU and we may expect that this vibration will translate to an acoustic signature in the Shuttle pre-launch environment. We now proceed to investigate this proposition with an examination of the acoustic environment present in the Shuttle payload bay prior to launch.

APU output shaft horsepower is delivered to the hydraulic pump at a nominal 3800 RPM's through a two stage reduction gear. Although balanced to within exceedingly fine tolerances dictated by extremely fast rotational RPM's the APU is nevertheless characterized by a specific vibrational signature. We shall examine this signature in some detail because a careful understanding of its nature is crucial to the development of a filter dedicated to its detection.

In Appendix A there is included a graphical summary of the results of hotfire testing of one APU installed in the Shuttle Orbiter. These are Power Spectral Density (PSD) plots of accelerometers mounted along the x-axis (D0280A) and z-axis (D0281A) of this particular APU for various levels of loading. Also shown for reference is a PSD plot of the background noise prior to APU ignition. Relative to the plots of APU vibration we see that in each case the background noise is no less than two orders of magnitude lower than the PSD peaks of data obtained from the loaded APU.

The results of this test reveal a very particular vibrational signature for the APU. It should be noted that there are consistently repeating peaks at 600 Hz and two harmonics above this value at 1200 Hz and 1800 Hz. The bandwidth of the filter employed in this investigation had a 3 dB rolloff at 2300 Hz. This obviated a close examination

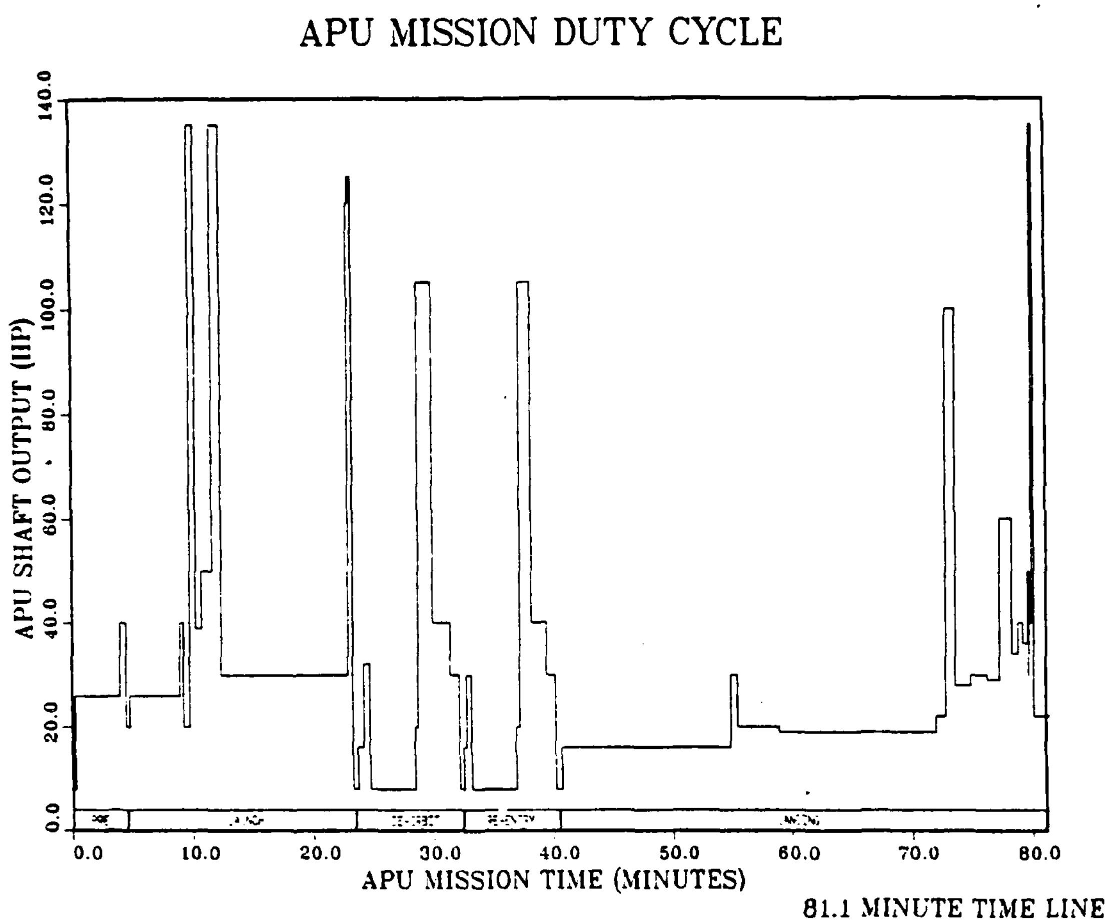


Figure 2.1 APU Mission Duty Cycle (Baseline 81.1 Minute Time Line)

In the pre-launch phase APU loading varies minimally from 8.0 to 40.0 horsepower according to hydraulic requirements during the phase. Because the APU is designed to operate at 72,000 RPM's (plus or minus eight percent) over its entire range of output shaft horsepower, its steady state and dynamic vibrational characteristics vary little during the pre-launch phase. This is due to the minimal hydraulic loading which characterizes this phase.

II. THE AUXILIARY POWER UNIT (APU) AND THE SHUTTLE PAYLOAD BAY PRE-LAUNCH ACOUSTIC ENVIRONMENT

A. APU DESCRIPTION

The Auxiliary Power Unit (APU) was developed by the Sundstrand Corporation under contract from Rockwell International, the prime contractor for the Space Shuttle. Each Shuttle Orbiter is equipped with three complete APU's and associated hydraulic systems. Each APU and its hydrazine fuel system is independent of the other two during normal operations. However, there are cross-ties between hydraulic systems which allow any two APU's to pick up the load from a third should it fail during operation.

B. APU MISSION DUTY CYCLE

Figure 2.1 is a representative diagram of a typical APU Mission Duty Cycle for an entire Shuttle mission [Ref. 2]. It is expected that a minimum of two restarts from a cold condition will be typical in a mission. The baseline duty cycle calls for 81.1 minutes of APU operation at various power levels from 8.0 horsepower to its maximum rated 135.0 horsepower. This includes launch, de-orbit, re-entry and landing phases and so includes operation at all altitudes corresponding to extremes of airfoil atmospheric resistance and Orbiter speed.

Several different schemes for accomplishing a matched band-pass filter design will be discussed. Nominally this will include an analog design built around operational amplifiers and a digital design of Infinite Impulse Response (IIR) implemented with an INTEL 2920 Signal Processor. This latter configuration will be derived from a cascaded IIR design which results from the Bilinear transformation of the analog filter. The difference equation representation of the digital filter transfer function will form the basis for the 2920 design. In addition I will discuss further design alternatives including an Adaptive Finite Impulse Response (FIR) Weiner filter and an idea for a design centered about a speech processing algorithm. Advantages and disadvantages of each approach will be discussed.

Ultimately one design will be chosen for integration within the GAS experiment just described and scheduled for launch in an upcoming Shuttle mission.

T-5 minutes. These units are essentially jet engines which provide for hydraulic power of Shuttle airfoil control surfaces during the atmospheric phases of launch and landing. They are designed to operate at very high (72,000) RPM's, but also generate a very specific acoustic signature in the audio range of the spectrum during normal operation. If it is possible to detect this APU acoustic signature during the pre-launch sequence of events and to discriminate this signature from among the various other acoustic events which may also occur during the pre-launch phase, then it may be possible to signal experiment initiation by detection of this event.

The emphasis of this thesis will be to describe the nature of the APU acoustic signature and to develop a matched filter which is tuned to its characteristics. It should be emphasized from the beginning that this it is not my intention to develop a classical "matched filter" which rigorously conforms to that definition. I do not have the uncontaminated APU acoustic signature data at my disposal which would allow that sort of an analysis. Rather it is my intention to examine the APU signature in the Shuttle cargo bay environment and to develop a filter which is "matched" to that contaminated signature. It is expected that an extremely narrow (high Q) band-pass filter will accomplish this goal.

significant gap in any serious analysis of the acoustical and vibrational transients which must accompany ignition and launch.

One means of lessening the impact of transient delay is to substitute a solid state data recorder for the traditional magnetic tape variety of instrumentation data recorders. This idea is being investigated by another team of researchers at the Naval Postgraduate School for incorporation into a deployable GAS mission canister. Despite the advantage which a solid state recorder will afford toward minimizing the transient delay prior to meaningful data collection, it can never completely eliminate the transient effect which accompanies any scheme which is trying to measure the same signal which it also uses for experiment initiation. There is a causal imperative here which is inescapable.

It is the purpose of this thesis then to develop a means of experiment initiation which will allow data collection to commence well before SSME ignition. In this manner we will be allowed a full measure of the acoustical and vibrational environment which is present in the Shuttle payload bay during launch.

C. DISCUSSION OF THE GENERAL SOLUTION ALGORITHM

The timeline of Shuttle events prior to launch includes the turn-on of Auxiliary Power Units (APU) at approximately

III. ANALOG FILTER DESIGN THEORY AND IMPLEMENTATION

The traditional method of filter implementation in electronic circuitry was for a long time characterized by an implementation of passive and discrete resistive, inductive and capacitive components tuned to respond to the desired frequency components of the input signal. In the earlier years of circuit design this procedure involved a lengthy process of theoretical development and precise component selection. This was often a tedious process involving component trial and substitution.

Analog filter design took a great leap forward with the advent of integrated operational amplifiers (op-amps) and later, integrated circuit (IC) technology. Now for instance, using an integrated circuit such as the National Semiconductor AF100 Universal Active Filter as a design basis, it is possible for a circuit designer to construct a precise analog filter circuit with a surprising economy of effort. We will use the Biquad Elliptic Filter design which forms the basis of the AF100 to implement the analog designs we shall develop herein.

A. PRACTICAL DEVELOPMENT OF AN ANALOG BAND-PASS FILTER

We will begin our development of a tuned filter design by examining an analog IC implementation of a band-pass filter with a center frequency of 600 Hz. (We could expand this band coverage to include the two additional center frequencies of 1200 Hz and 1800 Hz if it proves that such a design modification is necessary.) In this development we shall choose to build an Elliptic (or Cauer) filter which exhibits a much steeper roll-off outside the passband over a Butterworth or Chebyshev design of equivalent order. The disadvantage of ripple in the passband, which characterizes Elliptic filters, will cause minimal impact and will not be a factor in the realization of our goal. In fact we can allow the ripple in the passband to be relatively high because our intent is not to pass a faithful representation of the APU signature but only to detect its presence. Thus our goal is to construct a band-pass filter with a high quality factor and narrow passband. This corresponds to a steep roll-off out of the passband.

The method which will be employed to realize this band-pass filter will be to describe the characteristics of the desired analog band-pass design and, using a low-pass-to-band-pass transformation, solve for the form of the corresponding low-pass prototype using transform relations. This will allow us to determine the necessary order of the low-pass design which can subsequently be transformed into the required band-pass filter.

If we again examine the PSD plots of the APU noise above the background for the 600 Hz component we can generalize a desired filter transfer function to approximate this response. Let us choose the 3 dB points of the band-pass filter (centered at 600 Hz) to be at 575 and 625 Hz. We will allow the pass-band ripple width (PRW) to be as much as 2 dB within the pass-band. Let us furthermore require the stop-band attenuation on either side of the pass-band to be down at least 30 dB at 500 and 700 Hz. This represents a very steep roll-off characteristic and suggests the use of an Elliptic filter design for that reason. In fact, to achieve this degree of roll-off in a low-pass design would require a model prototype of order 6. The equivalent Chebyshev design would require a minimum order of 14, while the Butterworth low-pass filter equivalent order would be at least 63. Clearly the Elliptic design is our only viable alternative.

The low-pass to band-pass transformation for analog filters results in a transfer function which raises the order of the low-pass equivalent by a factor of two. Therefore if we design a band-pass filter it will necessarily consist of a number of second-order stages in the final implementation.

1. Low-Pass to Band-Pass Frequency Transformation

In order to develop the transfer function of an appropriate analog band-pass filter we must begin with the transfer function of the corresponding analog low-pass filter which may be transformed into the desired band-pass filter by a frequency transformation. However, the characteristics of our final filter are known in band-pass form. Thus we must deduce the analog low-pass design from the band-pass characteristics and then apply the analog transformation to the low-pass prototype to realize our goal. This development is a combination of procedures described in Chen [Ref. 5] and Johnson [Ref. 6].

We wish to design an analog band-pass filter having the following characteristics

f'_{p2} = 575 Hz

f'_{q2} = 500 Hz

f'_{p1} = 625 Hz

f'_{q1} = 700 Hz

PRW = 2.0 dB (allowable ripple in the passband)

MSL = 30 dB (minimum attenuation in stop-band)

In this analog development the prime frequencies refer to the band-pass function and the unprimed frequencies to the low-pass function. Figure 3.1 is a graphical depiction of the relationship between the transfer function transform pair. The negative axis frequencies arise from the mathematics of this development. However, we shall only be functionally concerned with the positive axis transformation. We are also considering the low-pass function to be normalized ($\omega = 1$).

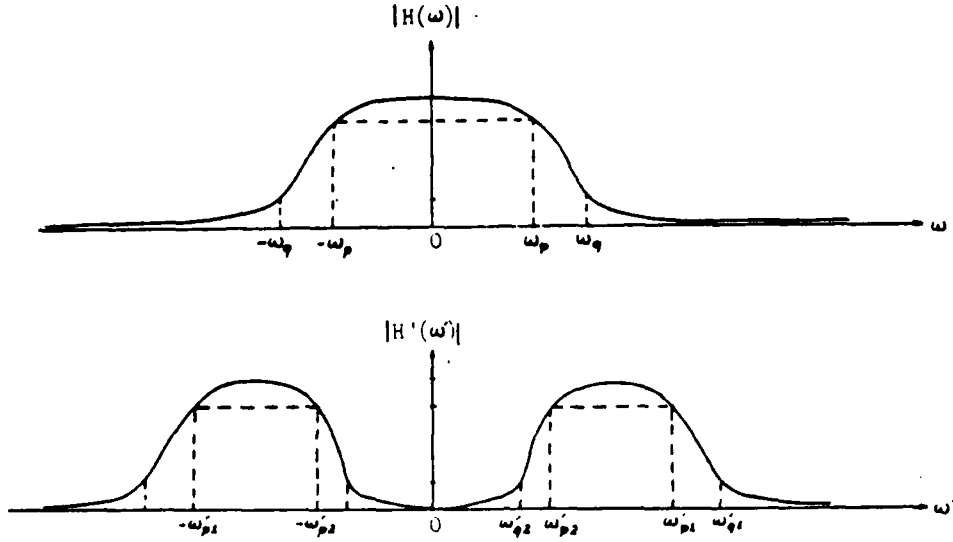


Figure 3.1 A Low-Pass to Band-Pass Frequency Transformation in the Analog Domain

A suitable transformation must therefore accomplish the transform relations detailed in Table 3.1 [Ref. 5].

Table 3.1

LOW-PASS TO BAND-PASS TRANSFORM PAIRS

Low-Pass Function	Band-Pass Function
$\omega = \infty$	$\omega' = \infty$
$\omega = 1$	$\omega' = \omega'_{p1}$
$\omega = -1$	$\omega' = \omega'_{p2}$
$\omega = -\infty$	$\omega' = 0$

As developed in Chen [Ref. 5: p. 235], the analog frequency transformation which will accomplish a low-pass to band-pass frequency transformation [H(s) ==> H(s')] is given by the following relation.

$$s = \frac{s'^2 + \omega'_{p1}\omega'_{p2}}{s'(\omega'_{p1} + \omega'_{p2})}$$

or if we substitute the above values

$$s = \frac{s'^2 + 4\pi^2 \bullet (3.59375 \times 10^5)}{s' \bullet 2\pi \bullet 50}$$

or, using $s = j\omega$

$$\omega = - \frac{4\pi^2 \bullet (3.59375 \times 10^5) - \omega'^2}{\omega' \bullet 2\pi \bullet 50}$$

Making the following substitutions into the preceding equation yields

$$\omega'_{q1} = 2\pi \bullet 500$$

$$\Longrightarrow \omega_{q1} = -4.3750000$$

and

$$\omega'_{q2} = 2\pi \bullet 700$$

$$\Longrightarrow \omega_{q2} = -3.7321429$$

Therefore we are left with the need to design an analog low-pass filter with

ω_c = 1 rad/sec, and

ω_q = 3.7321429 rad/sec

This leaves us with a normalized low-pass transition width (TW_{lp}) of

$$TW_{lp} = \omega_q - \omega_c$$
$$= 2.7321429$$

We now have enough information to enter the tables in Johnson [Ref. 6] to obtain the data shown in Table 3.2.

Table 3.2

ELLIPTIC LOW-PASS FILTER DATA
(for the band-pass filter low-pass prototype)

A	B	C	WZ	WM	KM
21.16400	0.787152	0.842554	4.600435	0.715610	1.258925

2. 600 Hz Elliptic Band-Pass Filter Design

In the case of elliptic band-pass filters the transfer function may be factored into the product of second-order functions. The two factors arising from each second-order low-pass stage have the forms [Ref 6: p. 100]

$$\left[\frac{V_2}{V_1}\right]_1 = \frac{K_1\sqrt{C/A}(s^2 + A_1\omega_0^2)}{s^2 + (D\omega_0/E)s + D^2\omega_0^2} \qquad 3.1$$

and

$$\left[\frac{V_2}{V_1}\right]_2 = \frac{K_2\sqrt{C/A}\,(s^2 + \omega_0^2/A_1)}{s^2 + (\omega_0/DE)s + \omega_0^2/D^2)} \qquad 3.2$$

where

$$E = \frac{1}{B}\sqrt{\frac{C + 4Q^2 + \sqrt{(C + 4Q^2)^2 - (2BQ)^2}}{2}}$$

$$D = \frac{1}{2}\left[\frac{BE}{Q} + \sqrt{\frac{(BE)^2}{(Q)^2} - 4}\right]$$

and

$$A_1 = 1 + \frac{1}{2Q^2}(A + \sqrt{A^2 + 4AQ^2})$$

and $Q = f_0/BW = 600/50 = 12$. The coefficients A, B and C are those of the normalized low-pass function given in Table 3.2 above, and K_1 and K_2 are related to the stage gain K by $K = K_1K_2$.

Equations 3.1 and 3.2 above are of the general form

$$\frac{V_2}{V_1} = \frac{\rho(s^2 + \alpha\omega_0^2)}{s^2 + \beta\omega_0 s + \gamma\omega_0^2} \qquad 3.3$$

which is identical to the form of the low-pass transfer function, except for the replacement of ω_0 by the corresponding low-pass term ω_c.

Our analog band-pass filter will have two stages of the form given by Eq. 3.3. Comparing Eq. 3.3 with Eq. 3.1 and Eq. 3.2 reveals the following transfer function coefficients of the band-pass filter stages [Ref 6: p. 118]:

1) First stage

$\rho = K_1 \sqrt{C/A}$

$\alpha = A_1$

$\beta = D/E$

$\gamma = D^2$

2) Second stage

$\rho = K_2 \sqrt{C/A}$

$\alpha = 1/A_1$

$$\beta = 1/DE$$

$$\gamma = 1/D^2$$

In the FORTRAN program ABPDBP (included as Appendix C) many of the calculations which are indicated in this thesis development will be performed. In Section 1 of this program we begin with desired filter parameters and tabulated values which correspond to the filter we wish to build. We then calculate several derived parameters from these initial values. Next we perform further calculations (which shall be developed shortly) which yield values for filter resistors and capacitors.

In Section 2 of ABPDBP we use the two sets of filter parameters which correspond to each of the filter stages indicated by Eq 3.3 to arrive at the overall transfer function. ABPDBP describes the corrections which must be made to provide for pre-warping of frequencies preparatory to a digital transformation (which shall be discussed in Chapter 4) but these changes can be ignored in this analog discussion. Thus we will use f_0 = 600 Hz (which implies the use of WO, not WODIG) in the program calculations. The fourth order analog filter transfer function which results from this development is given in Equation 3.4.

$$H(s) = \frac{a_0 s^4 + a_1 s^3 + a_2 s^2 + a_3 s + a_4}{b_0 s^4 + b_1 s^3 + b_2 s^2 + b_3 s + b_4} \qquad 3.4$$

The values of the coefficients in this analog transfer function representation are given in Table 3.3 which follows.

Table 3.3

ANALOG ELLIPTIC BPF FOURTH ORDER COEFFICIENTS
(Normalized coefficients with gain 0.03981)

Coefficient	Value
a_0	1.0000
a_1	0.0
a_2	2.9587×10^7
a_3	0.0
a_4	1.8991×10^{14}
b_0	1.0000
b_1	2.4351×10^2
b_2	2.7642×10^7
b_3	3.3558×10^9
b_4	1.8991×10^{14}

In Section 2A of the program ABPDBP the poles and zeros of the analog transfer function are then calculated to demonstrate the stability of the filter design. The values for the poles and zeros may be observed in the output of ABPDBP and are reproduced graphically in Figure 3.2 on the following page. The poles of the filter lie within the left half of the s-plane and this confirms the stability of our design.

3. Analog Band-Pass Filter Simulation

Now that we have developed the transfer function which describes the desired analog band-pass filter we can use this function to simulate the active operation of the

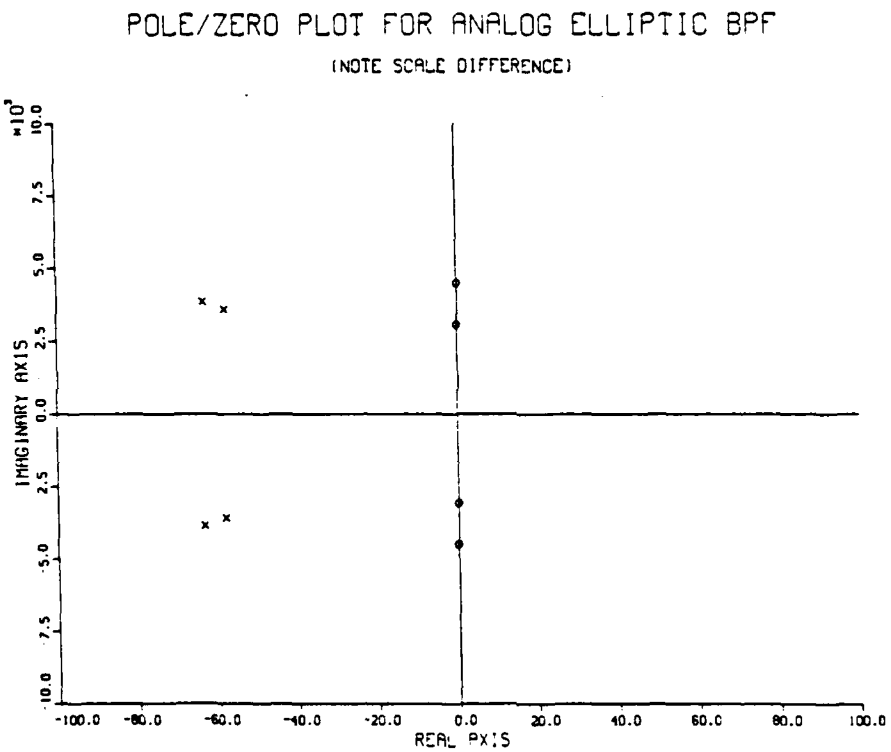


Figure 3.2 Pole/Zero Plot for the Analog Elliptic Band-Pass Filter

filter. The FORTRAN program ABPFR (which is included as Appendix D to this thesis) is used to examine this particular band-pass filter simulation. Figures 3.3, 3.4

and 3.5 which follow are the results of this computer simulation of the filter response for the device we have just designed. The range of frequencies of the simulated computer input is DC to 1 kHz. The simulated amplitude is constant over the range of input frequencies.

In Figure 3.3 we see the amplitude response which is near zero at all but the passband frequencies around 600 Hz. Between 500 and 700 Hz we confirm the desired filter response. The center frequency is located at 600 Hz and

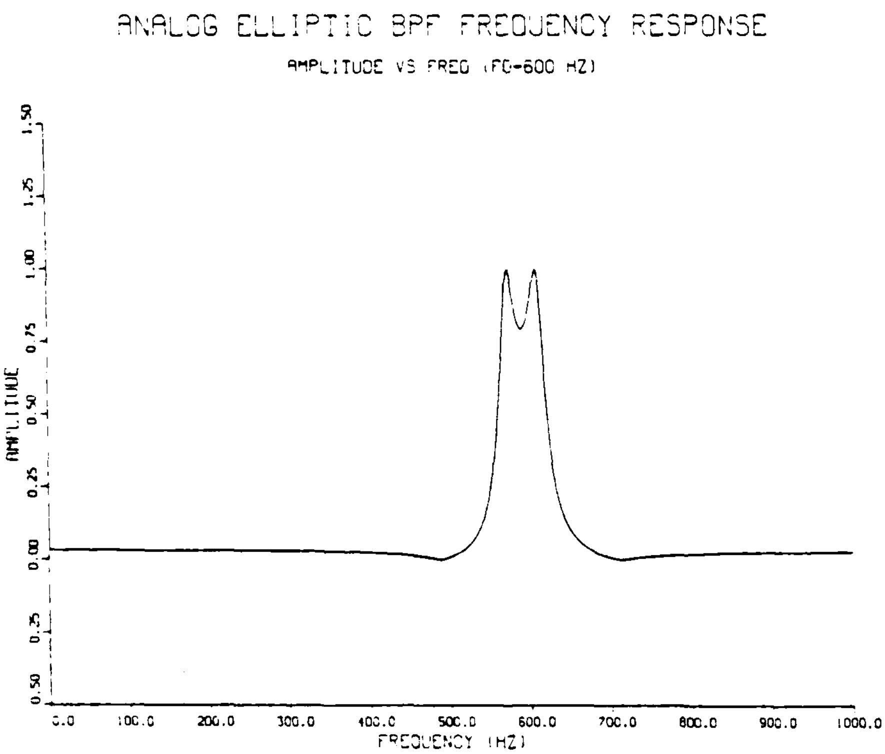


Figure 3.3 Analog Elliptic BPF Frequency Response (Computer Simulated Amplitude Response)

there is a significant minimum at the center frequency due to the effect of the passband ripple of 2.0 dB. Furthermore we observe a half power point (3.0 dB down point) at about 575 Hz and 625 Hz as specified in our design.

In Figure 3.4 we again observe a computer simulation of the amplitude response of the filter, this time measured in decibels. The marked presence of notches at about 500 Hz and 700 Hz is obvious, and the 30 dB minimum loss in the stopband is also confirmed. We also have graphical confirmation of the 2.0 dB ripple width in the passband.

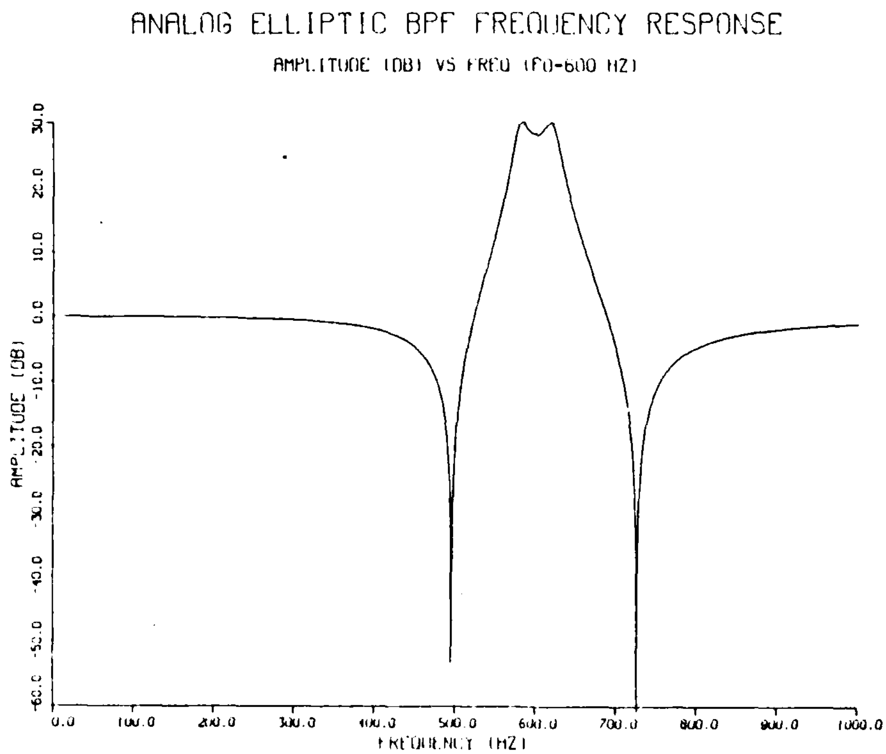


Figure 3.4 Analog Elliptic BPF Frequency Response
(Computer Simulated Amplitude Response in dB)

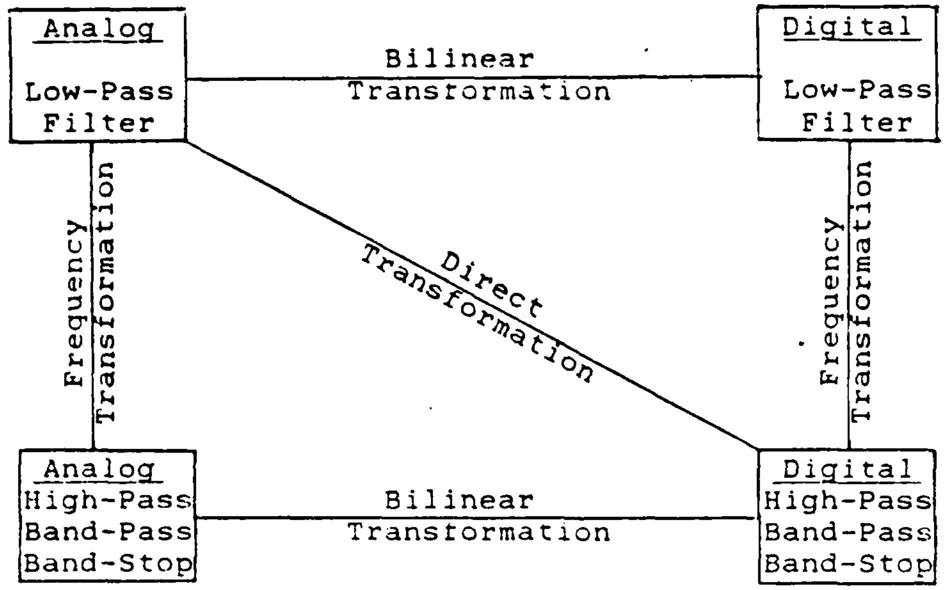


Figure 4.1 Analog and Digital Frequency Transformations

It should be reiterated that our goal in this section will be to develop a digital filter of Infinite Impulse Response (IIR) characteristics. This means that our filter will use the results of previous outputs to realize a later output. Although Finite Impulse Response (FIR) digital filters offer several qualitative advantages over IIR designs in the areas of phase linearity, stability, and an inherent protection against round-off error, they also require a larger number of delay elements to realize a design with a steep filter roll-off. This will be of concern to us when we realize an implementation in hardware with devices limited to a relative few number of filter transfer function poles and zeros.

case known analog filter characteristics in the frequency domain (the Laplacian "s" domain) are converted to similar characteristics in the digital "z" domain. Each of these techniques introduces a non-linearity into the resulting amplitude and phase characteristics of the original analog filter. If necessary to preserve the phase, equalizers may be employed to return the phase characteristic to a nearly linear behavior over the region of interest in the digital domain. In our case any phase distortion can be ignored because we are only interested in frequency detection and not accurate reproduction.

Generally speaking, when beginning with an analog low-pass design, we may proceed in a number of ways to arrive at a corresponding digital band-pass filter realization. For instance, we may first transform the low-pass filter to an analog band-pass design (as we did in Chapter 3 for the low-pass to band-pass transformation) and then employ an analog to digital transformation to yield the digital filter. Alternatively we may choose to employ the analog to digital transform on the low-pass filter and then apply a digital low-pass to digital band-pass transformation to realize our goal. Finally, it is also possible to combine these two-step routines into a single-step analog low-pass to digital band-pass direct transformation. These options are shown in Figure 4.1 [Ref. 5: p. 269].

IV. DIGITAL FILTER DESIGN

When designing an IIR digital filter for a specific application it is common practice to first develop an analog filter with appropriate characteristics as we did in the preceding chapter. Once the analog design is attained it is then possible to transform this analog filter into a digital filter with the desired passband characteristics.

There are several reasons why it is desirable to use this approach [Ref. 7: p. 5-7]. Of primary importance is the fact that the art of analog filter design is highly advanced. Consequently there are many techniques available for implementing specific designs. Because useful results can be achieved, following established analog design procedures presents advantages in the amount of effort which must be spent in the design phase.

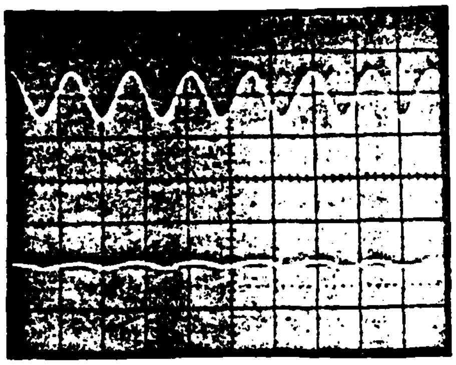
Additionally, many useful analog design methods have relatively simple closed-form design formulas. This greatly facilitates the implementation of the corresponding digital filters.

Finally, in many applications it is of interest to use a digital filter to simulate the performance of an analog linear time-invariant filter.

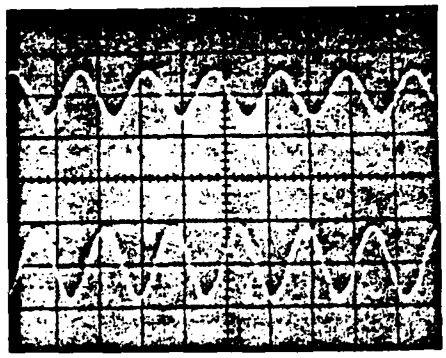
There are many alternative methods for accomplishing a transformation of fixed filter characteristics. In each

In Figure 3.10 we examine this response more specifically for discrete frequencies in the range of 500 Hz to 700 Hz. Instead of applying a ramped sinusoid we input five discrete sinusoids while maintaining a constant amplitude. Thus we again observe the very narrow bandpass filter response at least within the limits presented here. We also confirm the rapid shift in phase of 180 degrees from the lower to the upper bound in agreement with theory. While this examination by itself does not confirm the desired filter response, it does so when considered with the results of the previous figure.

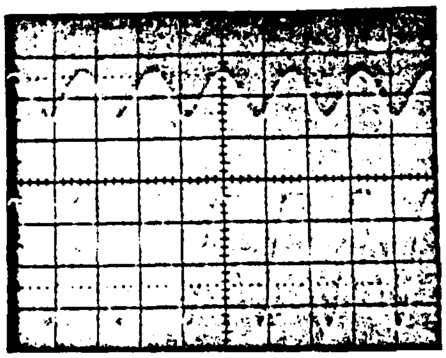
In Chapter 4 we will use the results of this analog filter implementation to develop an equivalent digital realization. To do this we will use common transformation techniques to arrive at a z-domain transfer function which we will then reduce to a difference equation. This format will then allow us to realize a digital elliptic filter by use of the INTEL 2920 Signal Processor. This hardware realization of the digital filter will be accomplished in Chapter 5.



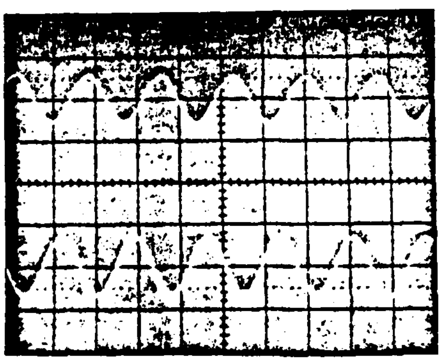
a) 500 Hz



b) 575 Hz



c) 600 Hz



a) 625 Hz

b) 700 Hz

Figure 3.10 Analog Elliptic BPF Frequency Response
Upper trace (Input): 50 mV/div scale
Lower trace (Output): 1.0 V/div scale

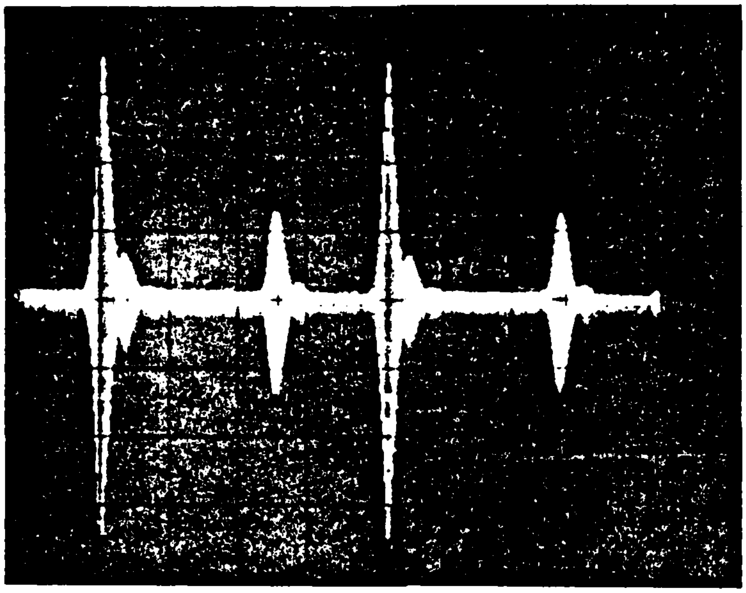


Figure 3.9 Analog Elliptic BPF Frequency Response (Photograph of Actual Filter Amplitude Response to a Ramped Sinusoidal Input)

This gives rise to the appearance of a double response which is noted in the figure. Actually we are observing a multiple reponse over successive up and (faster) down ramps of the input sinusoid. Thus we are able to observe graphical confirmation of the filter amplitude response predicted in the foregoing discussion.

As expected we observe a very narrow filter bandpass response (with frequency limits we will look at more closely in the following paragraph). The curious extended response ("hump") at the upper end of the passband is due to the inexact placement of poles and zeroes accomplished by tuning of the filter response in the aforementioned manner.

The two 74161 counter stages which follow the multivibrator are designed to count up to 255 occasions of the threshold being exceeded in a .5 second period before the decision is made that a valid 600 Hz signal was detected. This is an arbitrary figure. The .5 second period is established by the 555 timer which is also fed by the one-shot multivibrator. If the counter stages do not sum to 255 within a .5 second period then the 555 resets the counter stages to zero and counting begins anew with the comparator. If the counter does reach 255 within the .5 second period then a latch is set for the remainder of the .5 second period. This TTL level signal is the one which provides the microprocessor interrupt indicating that the APU signal has been detected.

C. ANALOG BAND-PASS FILTER IMPLEMENTATION RESULTS

Figure 3.9 on the following page is a photograph of the actual frequency response of the analog elliptic band-pass filter we have developed. A sweep generator was used to input a ramped sinusoidal input comprised of a linear continuum of frequencies (generated by application of a skewed triangular input to a voltage controlled frequency oscillator) in the range of approximately 100-1000 Hz. Because the ramp generator does not exhibit an instantaneous return, the return also generates a down-frequency response albeit at a rate greater than that of the up-frequency ramp.

which follow the amplifier. If the amplitude of the amplified filter output goes above the threshold set at the reference input of the LM311-based comparator then a pulse is developed for the duration that the input signal exceeds the threshold level. A negative-edge triggered 74121-based one-shot multivibrator follows the comparator. It is designed to send a one millisecond pulse to the counter stages which follow any time the comparator detects an input signal which exceeds the threshold level.

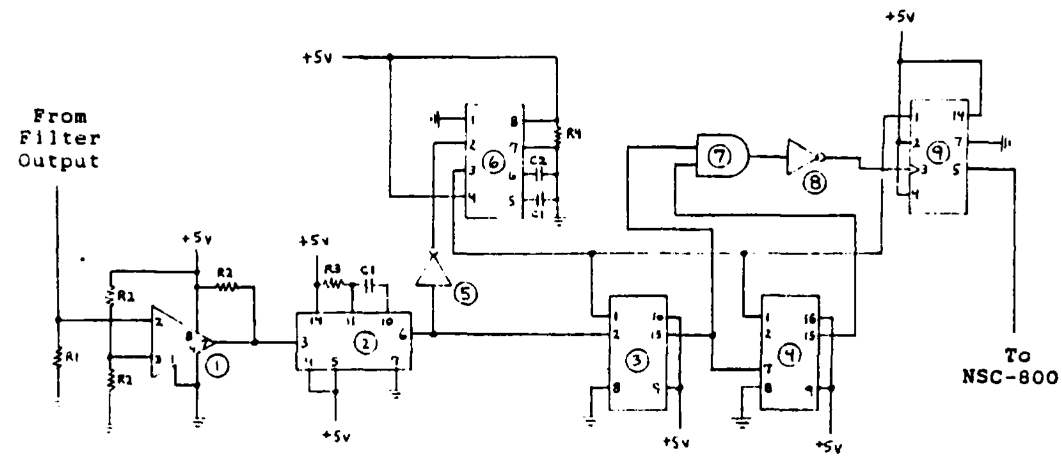
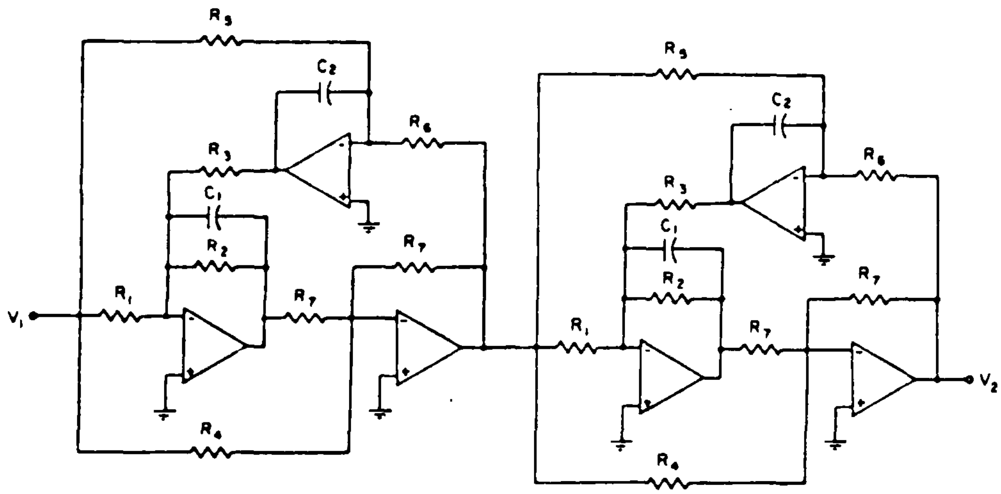


Figure 3.8 Follow-Up Pulse-Shaping Logic Circuitry

Microcircuits	Components
1: LM311 Op-Amp	R1: 100 kΩ
2: 74121 1 msec One-Shot	R2: 20 kΩ
3: 74161 4 Stage Counter	R3: 15 kΩ
4: 74161 4 Stage Counter	R4: 470 kΩ
5: 7404 Inverter	C1: 0.1 μF
6: 555 0.5 sec Timer	C2: 1.0 μF
7: 7432 AND Gate	
8: 7404 Inverter	
9: 7474 D-Type Flip-Flop	



First Stage Second Stage

Figure 3.7 Biquad Elliptic Band-Pass Filter Circuit

sinusoidal nature and inadequate to drive a microprocessor interrupt designed to accommodate TTL logic levels. Thus we must include further circuitry into our design which will send a TTL compatible logical signal to the microprocessor when the APU signal is detected. The circuit which accomplishes this is shown in Figure 3.8.

The output from the filter stages is first sent to a linear amplifier constructed around a 741 op-amp. Because the APU signal is low voltage out of the microphone detector it is necessary to amplify the filter output prior to logical evaluation.

The decision of whether or not a 600 Hz component is present is the function of comparator and counter elements possible to these values and then tuning the circuit for the desired performance. Tuning is accomplished in each stage by adjusting R_4 to set the notch frequency f_z, R_3 to set the center frequency f_0, R_2 to set Q, and R_1 or R_5 to set the gain.

Table 3.4

BIQUAD ELLIPTIC BAND-PASS FILTER COMPONENT VALUES

1st Stage		2nd Stage	
Component	Value	Component	Value
C_1	.00996	C_1	.01030
C_2	.00995	C_2	.01034
R_7	26.7	R_7	26.7
R_6	25.7	R_6	26.5
R_5	10.5	R_5	20.2
R_4	14.8	R_4	14.8
R_3	25.7	R_3	26.6
R_2	785.	R_2	813.
R_1	437.	R_1	452.

Note: Capacitor values are μF, resistor values are kΩ.

The resulting schematic for the fourth-order Biquad elliptic band-pass filter is shown in Figure 3.7.

2. Follow-Up Logic Circuitry

When the band-pass filter is implemented the effect is to produce a response which narrowly limits the passband to within a few tens of hertz about the center frequency of 600 Hz. Still, the output of this filter will be of

Similarly, the second stage values are given by

$$R_1 = \frac{DE\sqrt{A/C}}{K\omega_0 C_1}$$

$$R_2 = KR_1\sqrt{C/A}$$

$$R_3 = \frac{D}{\omega_0 C_1}$$

$$R_4 = \frac{A}{C}\sqrt{R_7/K}$$

$$R_5 = \frac{A_1\sqrt{A/C}}{KD\omega_0 C_2}$$

$$R_6 = \frac{C_1 R_3}{C_2}$$

In Section 1 of the program ABPDBP (introduced as Appendix C) we calculate the resistor values for the Biquad band-pass circuit just discussed. The resulting component values which were thus derived are shown in the appendix and are also included here as Table 3.4. These are computed values for resistance and capacitance. In fact the circuit is constructed by selecting standard values as close as coefficients and the derived complements we have already evaluated. We begin our development by choosing a standard value for C_1 (given roughly by $10/f_o$ μF) and then proceeding to calculate elemental values. In the equations which follow the values for C_2 and R_7 are arbitrary within limits, and are chosen to minimize the spread of resistance values. We pick $C_2=C_1$ and $R_7 \approx 1/(\omega_o C_1)$. A_1, E and D are as given previously.

The first stage values are thus [Ref. 6: p. 126]:

$$R_1 = \frac{E\sqrt{A/C}}{KD\omega_o C_1}$$

$$R_2 = KR_1\sqrt{C/A}$$

$$R_3 = \frac{1}{D\omega_o C_1}$$

$$R_4 = \frac{A}{C}\sqrt{R_7/K}$$

$$R_5 = \frac{D\sqrt{A/C}}{KA_1\omega_o C_2}$$

$$R_6 = \frac{C_1 R_3}{C_2}$$

B. HARDWARE IMPLEMENTATION

1. Biquad Analog Band-Pass Elliptic Filter

There are many ways to perform a hardware implementation of the analog band-pass transfer function we have just developed. One relatively easy method employs the use of op-amps as the active filter component. We shall use a Biquad op-amp filter implementation which exhibits good stability and ease of tuning. Additionally, implementation is made simpler by the use of a 74124 quad op-amp microchip which allows a single chip per second-order stage. The generalized circuit diagram for a second-order stage of a Biquad filter is shown in Figure 3.6 [Ref. 6: p. 127].

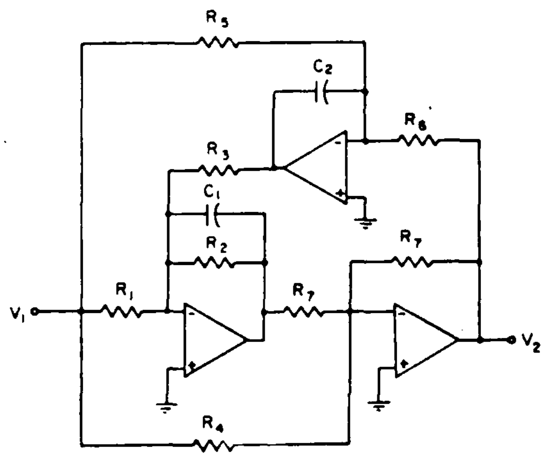


Figure 3.6 Biquad Elliptic Filter Circuit

Component resistor and capacitor values for the Biquad filter depend upon the low-pass normalized

Finally, in Figure 3.5 we view the computer simulation of the analog filter phase response. Although our application is not phase dependent (due to the fact that it is the presence alone of the 600 Hz element which is of concern to our circuit--not its accurate transmission), we do confirm the significant effect upon phase for our elliptic filter in the passband between 500 Hz and 700 Hz. Between these two frequencies we observe a 360 degree shift in phase.

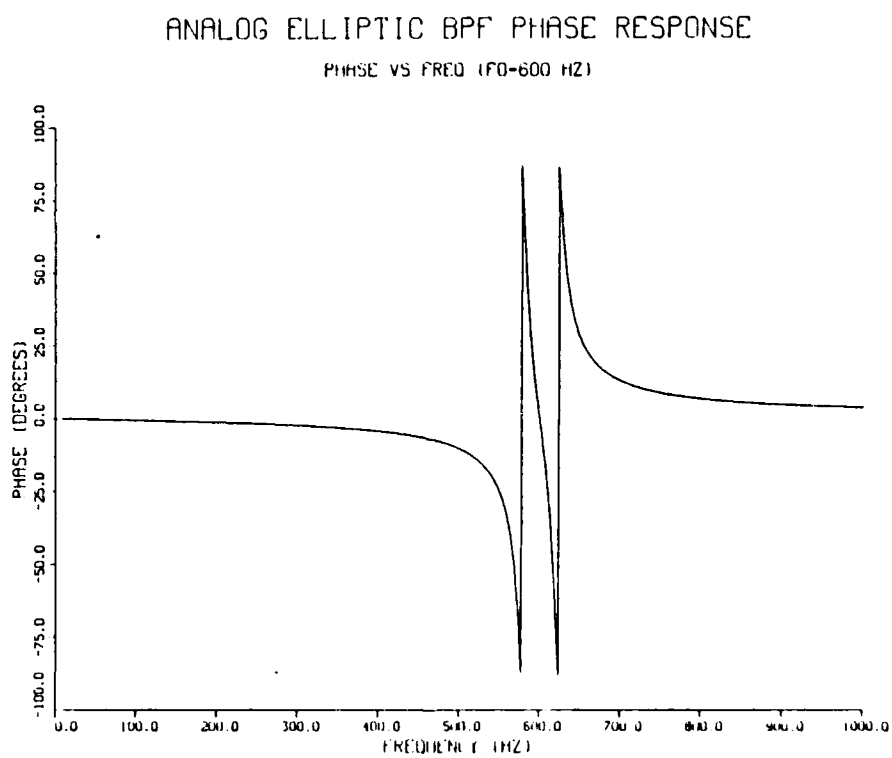


Figure 3.5 Analog Elliptic BPF Frequency Response
(Computer Simulated Phase Response)

A. ANALOG BAND-PASS TO DIGITAL BAND-PASS FILTER DESIGN

An analog-to-digital bilinear transformation makes it possible to apply a relation which transforms an analog band-pass filter into a desired band-pass digital design.

We will realize the 600 Hz digital bandpass filter by applying the bilinear transformation to the transfer function of the analog band-pass filter we developed in Chapter 3. Once implemented we will examine the performance of this filter in view of our goal of APU start-up identification. If necessary we will determine which refinements and modifications to our design may be necessary to realize our goal.

We recall from Chapter 3 that the transfer function of the analog elliptic band-pass filter was given by the product of the two second-order functions given by Eqs. 3.1 and 3.2. This product has been computed (as shown in Appendix C) and is found to be

$$\frac{V_2}{V_2} = \frac{\rho(s^4 + Fs^3 + Gs^2 + Hs + J)}{s^4 + Ms^3 + Ns^2 + Ps + Q}$$

where

$\rho = 0.039811$

$F = 0$

$G = 29.587 \times 10^6$

$H = 0$

$J = 189.91 \times 10^{12}$

$M = 0.24351 \times 10^{3}$

$N = 27.642 \times 10^{6}$

$P = 3.3558 \times 10^{9}$

$Q = 189.91 \times 10^{12}$

The poles of the analog band-pass filter transfer function were shown (graphically in Figure 3.5) to be:

$-63.567 \pm 3.8421j \times 10^{3}$

$-58.188 \pm 3.5859j \times 10^{3}$

The analog filter is therefore stable.

To transform the analog band-pass transfer function into a digital version we will use the Bilinear Transformation [Ref. 8: pp. 219-224] which is characterized by the following relation

$$s = \frac{2}{T}\frac{z - 1}{z + 1} \qquad 4.1$$

This transformation will map stable analog poles which are in the left-half of the s-plane into the interior of the unit circle in the z-plane. Thus stability is preserved in all cases.

If we then make the substitution $s = j\overline{\omega}$ and $z = e^{j\omega T}$ into Eq. 4.1 and simplify, we can establish the relationship between the frequencies in the analog and digital cases. (In this and the following discussion we shall denote frequencies in the analog case with an overbar ($\overline{\omega}$), and those in the digital case without one (ω).)

The resulting relation is

$$\overline{\omega} = \frac{2}{T} \tan \frac{\omega T}{2} \qquad 4.2$$

where T is the sample period given by $1/f_s$. In this case $f_s = (6.666 \times 10^6)/(4 \times 192)$, which we will show shortly. This results in $T = 1.15212 \times 10^{-4}$.

This relationship between analog and digital frequencies is shown in Figure 4.2 and reveals that the Bilinear Transform does not provide a linear mapping from one function to another. The frequency range from 0 to ∞ in the continuous case is warped into the frequency range from 0 to π/T in the digital case.

Therefore, if we have an analog filter with transfer function $\overline{H}(s)$, we may then perform the following substitution dictated by Eq. 4.1

$$H(z) = \overline{H}(s)\big|_{s=(2/T)[(z-1)/(z+1)]} \qquad 4.3$$

Another way of expressing this same relation is

$$H(e^{j\omega T}) = \bar{H}(j\bar{\omega})\,|_{\bar{\omega} = (2/T)\tan \omega T/2}$$

Using this relation the characteristics of H(z) can be obtained graphically from those of $\bar{H}$(s) as shown in Figure 4.2 [Ref. 5: p. 262].

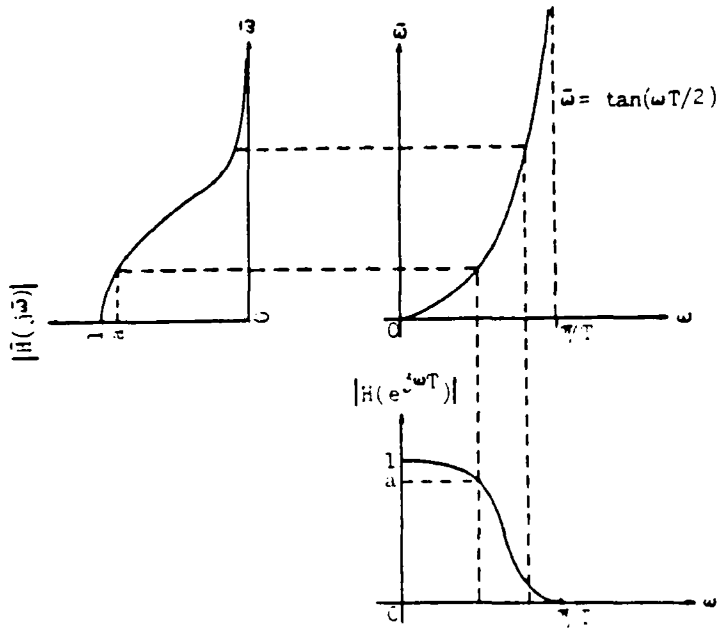


Figure 4.2 The Bilinear Transformation
(showing analog and digital transfer functions
and the non-linear warping of frequencies.)

We see from the figure that there is no aliasing problem associated with the transform because the frequency is limited to less than π/T (8680π) in the digital case. However, because of the frequency warping we have to make a proper transform of frequency according to Eq. 4.2

before application of the Bilinear Transform. Consequently, for a transformation of the analog band-pass filter derived as Eq. 3.3, we must substitute f_0 = 590.825 Hz for f_0 = 600 Hz before application of the Bilinear Transform to ensure a proper transformation to the digital domain. Once this is accomplished all we need do is apply the Bilinear Transform to the resulting "pre-warped" analog band-pass filter transfer function.

The FORTRAN based computer program previously introduced in Appendix C also provides for this development and implements Eqs. 4.2 and 4.3 to derive the following digital transfer function $H(z^{-1})$ for the desired digital band-pass filter

$$H(z^{-1}) = \frac{\rho(1 + Fz^{-1} + Gz^{-2} + Hz^{-3} + Jz^{-4})}{1 + Mz^{-1} + Nz^{-2} + Pz^{-3} + Qz^{-4}} \qquad 4.4$$

where the constants are as follows

ρ = 0.039516

F = -3.6279

G = 5.2861

H = -3.6279

J = 1.0000

M = -3.6251

N = 5.2586

$P = -3.5768$

$Q = 0.97353$

The poles of the transfer function given by Eq. 4.4 are

$0.90781 \pm 0.40813j$ (Magnitude = .99533)

$0.90475 \pm 0.40493j$ (Magnitude = .99123)

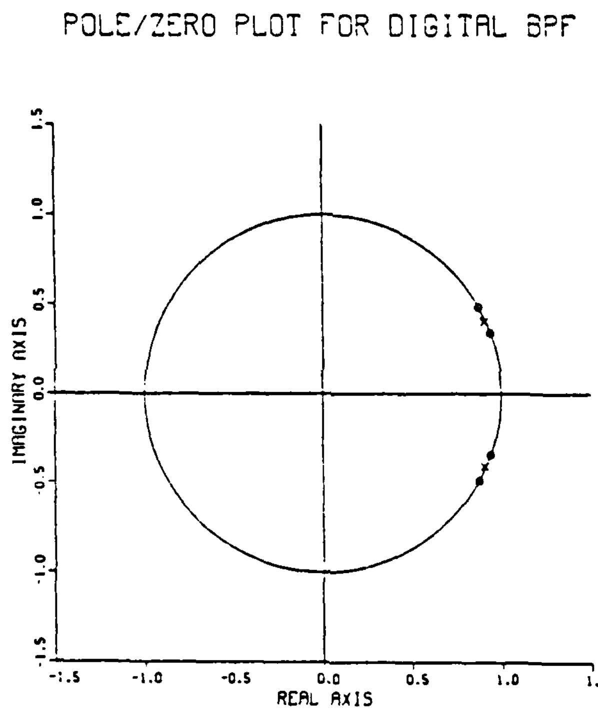


Figure 4.3 Pole/Zero Plot for the Digital Elliptic Band-Pass Filter (poles appear singular, but are in fact double and nearly coincident)

and thus we confirm the mapping of stable poles in the analog domain into stable digital poles located inside the unit circle. This digital pole/zero plot is shown graphically in Figure 4.3 on the previous page.

B. DIGITAL BAND-PASS FILTER SIMULATION

Equation 4.4 represents the digital filter transfer function equivalent to the analog filter transfer function we presented in Chapter 3. In a manner completely analogous to that development we are now able to demonstrate a computer graphical simulation of the digital filter frequency and phase response and compare these to the previous results. The FORTRAN program used to present this graphical output is included in Appendix E under the title DBPFR.

In Figure 4.4 we see the digital filter frequency response and observe that it is nearly identical to the analog response in consonance with our design goal. The minor differences are remarkable and explicable. The center frequency of the digital filter is diminished to the pre-warped center frequency of approximately 591 Hz. Additionally, the two peaks of the amplitude response located at about 585 Hz and 595 Hz are not of equal magnitude. This is due to the difference of pole proximity to the unit circle. Although the poles appear coincidental in the graphical presentation in Figure 4.3, they are actually distinct; the pole nearer to the real axis is some

.004 units closer to the unit circle which accounts for the amplitude disparity between the two poles.

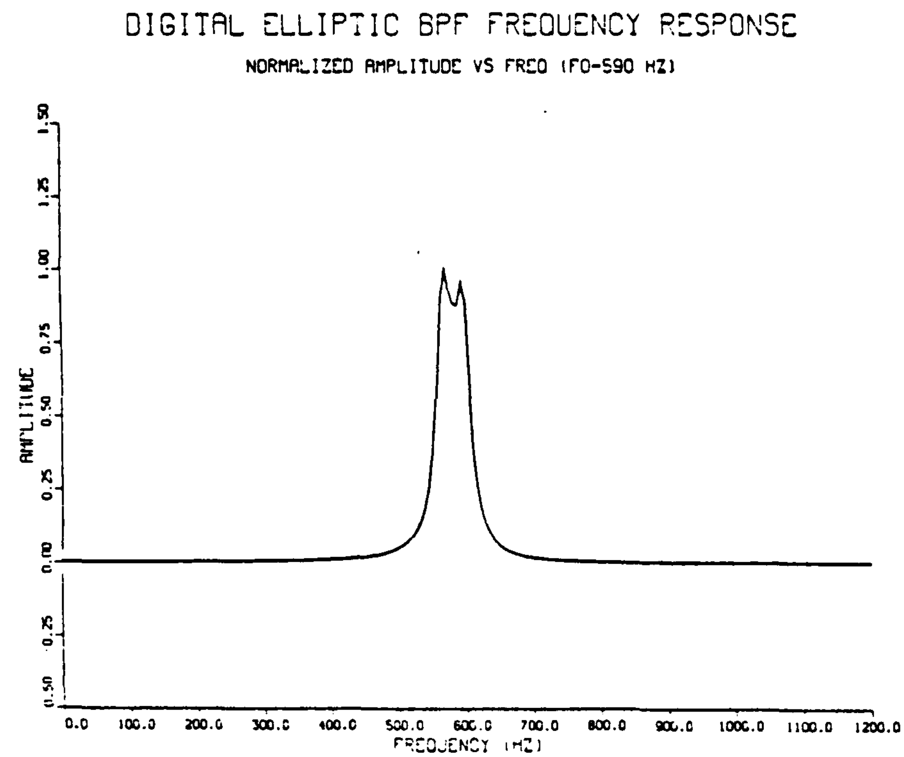


Figure 4.4 Digital Elliptic BPF Frequency Response (Computer Simulated Amplitude Response)

In Figure 4.5 we observe the digital filter frequency response as measured in dB. This curve appears somewhat different from its analog counterpart but the important feature is maintained. A steep filter rolloff is realized out of the passband and the response is diminished by about 30 dB at approximately 500 Hz and 700 Hz according to design specifications. Although the analog filter did not deviate much from this 30 dB down figure we see an added benefit of the digital filter wherein the rolloff continues monotonically over our observed spectrum.

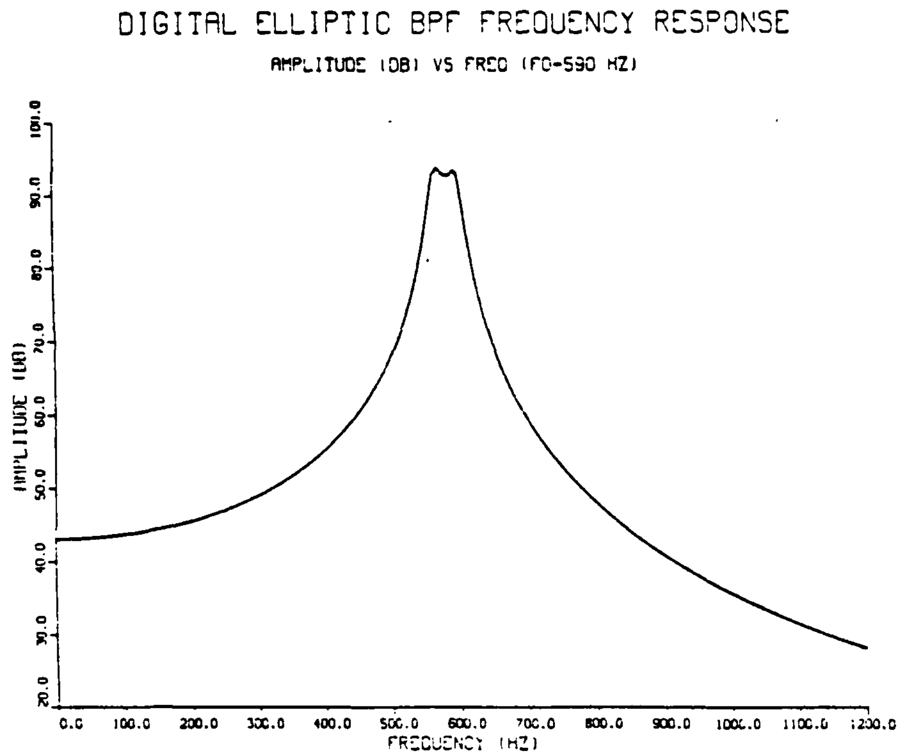


Figure 4.5 Digital Elliptic BPF Frequency Response (Computer Simulated Amplitude Response in dB)

Finally, in Figure 4.6 we observe the phase response of our digital filter. Once again this closely approximates the severe phase distortion we observed with the analog filter although the center frequency is again confirmed to be significantly less than the nominal 600 Hz we expected of the earlier filter design. To reiterate, this phase distortion is a hallmark of elliptic filters and the

Bilinear transformation, but our application is not phase dependent. Thus we may ignore this effect.

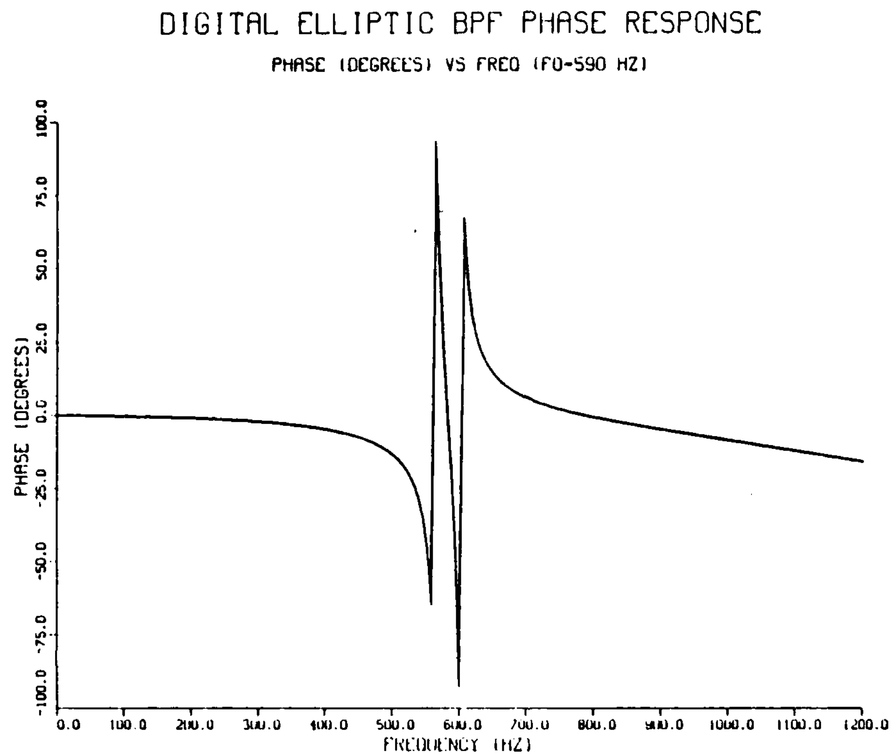


Figure 4.6 Digital Elliptic BPF Frequency Response (Computer Simulated Phase Response)

C. DIFFERENCE EQUATION REPRESENTATION

The transfer function for the fourth order Elliptic Filter was given previously in Equation 4.4. Section 4 of the program ABPDBP introduced earlier in Appendix C accomplishes a transformation of this quotient of fourth order polynomials and provides an equivalent cascaded representation of two second order filter stage blocks, each of the form

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{b_0 + b_1 z^{-1} + b_2 z^{-2}}$$

where X(z) and Y(z) refer to the filter input and output, respectively. The values of the dual quadratic coefficients of the cascaded second order transfer function and as computed by the program in Appendix 4 are given in Table 4.1.

Table 4.1

ELLIPTIC BPF SECOND ORDER STAGE COEFFICIENTS

1st Stage		2nd Stage	
Coefficient	Value	Coefficient	Value
a_0	1.0000	a_0	1.0000
a_1	-1.7503	a_1	-1.8780
a_2	1.0017	a_2	0.9981
b_0	1.0000	b_0	1.0000
b_1	-1.8163	b_1	-1.8092
b_2	0.9910	b_2	0.9822

Both second order z-domain filter stage transfer functions can be manipulated in a familiar way to realize the following z-domain difference equation.

$$b_0Y(z) = a_0X(z) + a_1X(z)z^{-1} + a_2X(z)z^{-2}$$
$$- b_1Y(z)z^{-1} - b_2Y(z)z^{-2}$$

Applying the inverse z-transform to this z-domain difference equation yields the time domain digital difference equation .

$$b_0y(k) = a_0x(k) + a_1x(k-1) + a_2x(k-2)$$
$$- b_1y(k-1) - b_2y(k-2) \tag{4.5}$$

The Signal Flow Graph corresponding to the difference equation given by Eq. 4.5 is shown in Figure 4.7. The difference equation representation is important because this is the basis for the hardware implementation of the digital filter which shall follow in Chapter 5.

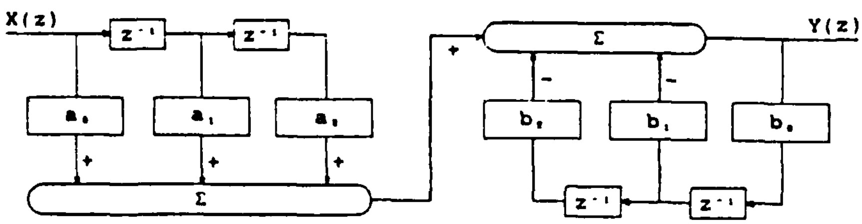


Figure 4.7 Second Order Stage Signal Flow Graph

D. DIFFERENCE EQUATION IMPLEMENTATION SIMULATION

We have just stated that the difference equation representation of the digital filter will become the basis for the INTEL 2920 Signal Processor implementation which is to follow. In order to show the adequacy of this method of implementation it is useful to demonstrate the impulse response and filter frequency response by computer simulation. In the following chapter sections we will demonstrate these simulations and show that they yield results in keeping with our design expectations.

1. Digital Filter Impulse Response

In Appendix F is presented the complementary FORTRAN programs S22I and S22IG. Both implement the impulse response simulation for the difference equation representation of the digital elliptic band-pass filter. In the case of S22I the output is digital and is shown to exemplify the filter response over a greater period of time than is usefully represented otherwise. In the case of S22IG the output is graphical and will be presented here.

In both these programs the impulse is equal to the greatest allowable input which guarantees an output of less than unity. This is done for reasons of filter stability as well as a limitation of the INTEL 2920 which will be discussed in the next chapter. By examining the impulse response we confirm the stability of the filter design by ensuring that the output decays to zero over time. In addition we can observe the natural response of the system by establishing the frequency of the decaying sinusoid.

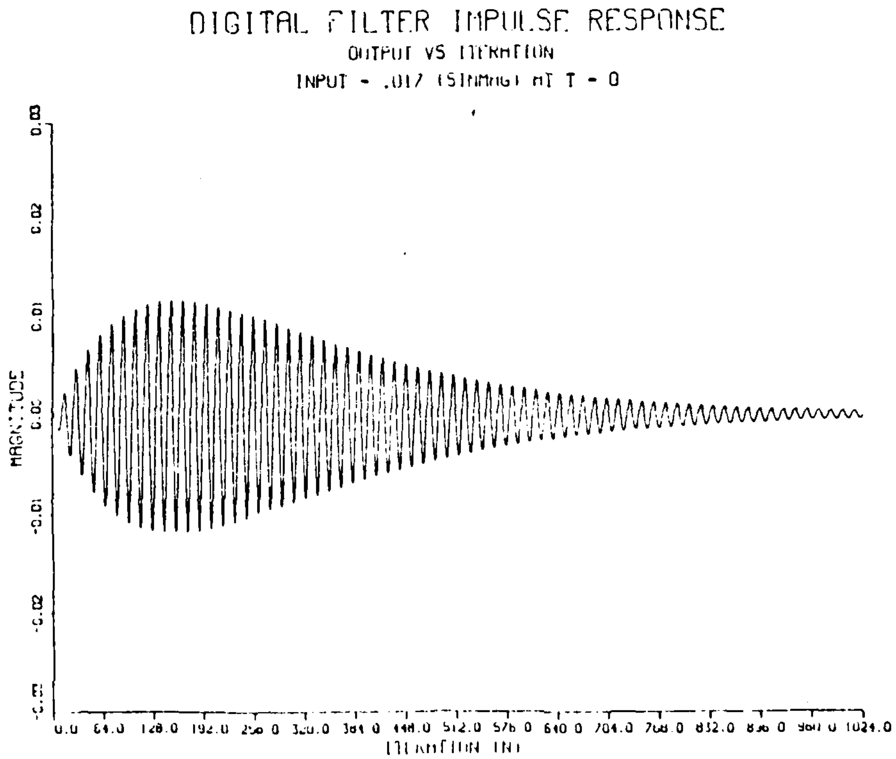


Figure 4.8 Digital Filter Impulse Response

In Figure 4.8 we observe the the response of the two stage difference equation filter for an input of 0.0172 applied at time T = 0. By counting the number of cycles which occur over a corresponding number of iterations, and realizing that the sample period is 0.11521 milliseconds we arrive at natural frequency which is very close to the expected 600 Hz. Additionally, it is apparently true that the response decays to zero with time, at least over the

Although we have ensured an input less than one volt this is not sufficient to guarantee that the post-processing value will not exceed the internal arithmetic limit. Internal arithmetic is limited to a range of values which cannot exceed -1.00000000 to +0.99999999. These 8 decimal place accuracies are established by the internal 25 binary bits (1 sign bit and 24 magnitude bits) available for arithmetic computations within the 2920. Actually the range of multiplicative inputs is only good to within 4 decimal places due to the scaling problem. But this will be seen to be more than adequate for our purposes.

To ensure that the processed values do not exceed one volt we scale down the digitally sampled input value by 64 by way of program step 44. The difference equation manipulations of the input value are then accomplished in program steps 47 through 130.

Digital arithmetic is performed in the 2920 by means of binary shifting and adding which is predicated on a transformation of coefficients to a nearest binary equivalent. The FORTRAN program and its output which performs this transformation is labelled CTRANS2 and is shown in Appendix I. Although a binary transformation does involve some approximation error, we see in the appendix that the worst case approximation of coefficients is still within .02 percent of the actual value. This is a relatively insignificant error.

D. A 2920 DIGITAL FILTER IMPLEMENTATION

Here we shall describe the particular 2920 software and hardware components which comprise the digital filter.

1. <u>2920 Assembly Language Program</u>

In Appendix H we find the 2920 assembly language program which implements the two stage difference equation developed in Chapter 4. Recognizing the characteristics of the 2920 processor, it is instructive to review the programming devices which are brought to bear to realize this filter. We will proceed in the order in which these devices are used in the program. Appendix H should be consulted as reference for the discussion which follows. A detailed discussion of the 2920 Assembly Language should be consulted for particulars concerning the language [Ref. 11].

After initializing the DAR register we accept the input analog sample from the sensor microphone/preamplifier ensuring that the level does not exceed 1.0 volts. This limit is established by the voltage reference circuitry at pin #8 of the 2920. The input analog value is stored in the Sample/Hold register.

We then begin a sequence of steps, according to 2920 protocol, which accomplish the analog to digital transformation of the input value in the Sample/Hold register. This procedure is completed at program step 43 and the resulting digital value is then found in the DAR register.

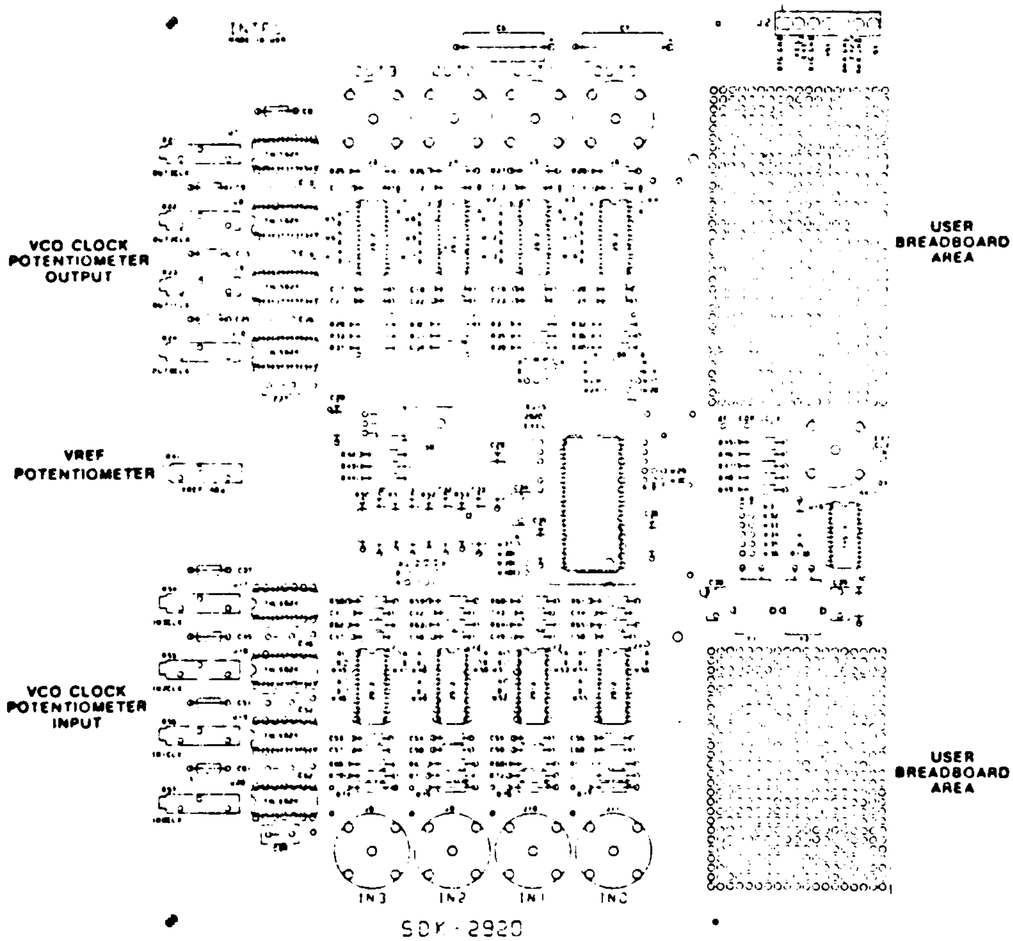


Figure 5.3 SDK-2920 Applications Board

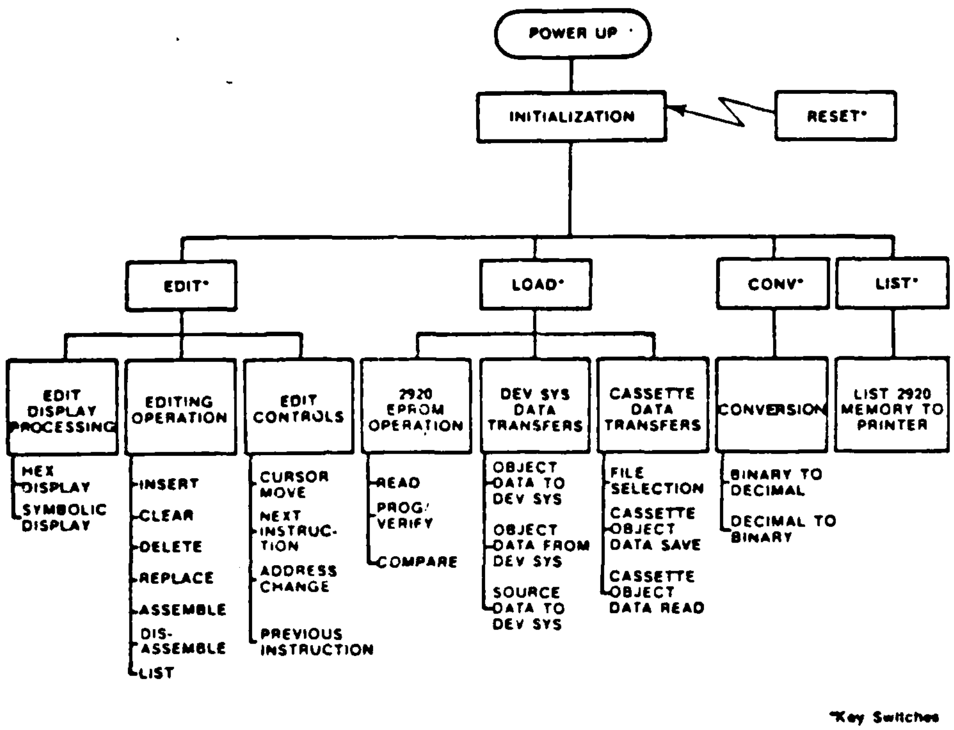


Figure 5.2 SDK-2920 Monitor Command Structure

of the SDK-2920 by examination of inputs and outputs. The applications board is shown in Figure 5.3 on the page following. Provisions are made on the board for assembly of either internal or external clocks, four input and four output channels with associated waveshaping circuitry, reference voltage development, and two user breadboard areas for specific applications development. Furthermore, TTL compatible output signals can be delivered to the output vice analog outputs if desired. We shall make use of this feature to send a signal detection pulse when the APU ignition is detected.

Applications software was developed using an Intellec Microcomputer Development System running a 2920 Assembly Program and Software Simulator. Transfer of 2920 software between the development system and the SDK-2920 is easily accomplished.

The SDK-2920 is physically divided into a development side and an applications side. The development side can be used to load, test and modify EPROM resident programs under 8085A microprocessor control. System control is accomplished with the use of a keypad monitor. The composition and hierarchy of the monitor command structure is shown in Figure 5.2 on the following page.

The applications side includes a prototype area for circuit construction and testing. It functions independently of the development side. After program development has taken place two methods may be used to accomplish program verification. The first method uses the Intel SM2920 Simulator Software to simulate the execution of programs written for the 2920. This simulator allows the use of symbolic references for changing and displaying all 2920 registers, flags and user-defined locations in program and memory storage. A trace feature also allows monitoring of selected parameters as they are changed under program control.

The second method of 2920 program verification is done by monitoring circuit performance on the applications side these functions the analog section includes the following subsections:

-- a four input multiplexer

-- an input sample-and-hold circuit

-- a D/A converter

-- a comparator

-- an output multiplexer with eight output sample-and-hold and buffer amplifiers.

-- a special digital-to-analog (DAR) register which acts as an interface between the digital and analog sections.

C. THE SDK-2920 DEVELOPMENT SYSTEM

The SDK-2920 Development System is an integral component in the development of any applications package which uses at its core the INTEL 2920 Analog Signal Processor [Ref. 10]. Within the scope of the system are many development capabilities including

-- Breadboarding: The breadboard is used to develop circuits for evaluation or prototype applications.

-- Assembling and Editing: This feature is comprised of an assembler, disassembler, hexadecimal display, symbolic 2920 instruction display, and single keystroke entry of many 2920 instruction fields.

-- 2920 EPROM Programming: The development board includes hardware and control elements necessary to program the 2920.

-- Communications: The development also interfaces with Intel Developments Systems (such as the Intellec Series) to pass object and source code listings of 2920 programs.

function, any applications program cannot make use of more than 192 instructions to process whatever number of input and output signals are being manipulated. But despite this restriction the power of the 2920 is evident. In our application we will only make use of a single input/output channel.

B. 2920 FUNCTIONAL DESCRIPTION

Figure 5.1 on the previous page details the block configuration of the 2920 [Ref. 9]. It is divided into the three major subsections described as follows.

The 192 x 24-bit Program Memory Section is a storage area implemented with EPROM. This section includes the instruction clock and timing circuits and program counter which control the operation of the entire device, including the other two sections.

The Arithmetic Section includes a 40 word by 25-bit scratchpad RAM and an arithmetic and logic unit (ALU). Both the RAM and the ALU are two port access devices. In the case of the ALU one of the ports is passed through a barrel shifter scaler. The function of the arithmetic section of the 2920 is to execute the commands dictated by the program memory.

The Analog Section performs A/D and D/A functions upon command from the program memory. In order to implement

In addition to the precision and speed of computation which the 2920 offers, it also allows for sequential processing of up to four separate input signals and eight analog output signals in a single program pass. This is of course dependent upon program complexity--regardless of

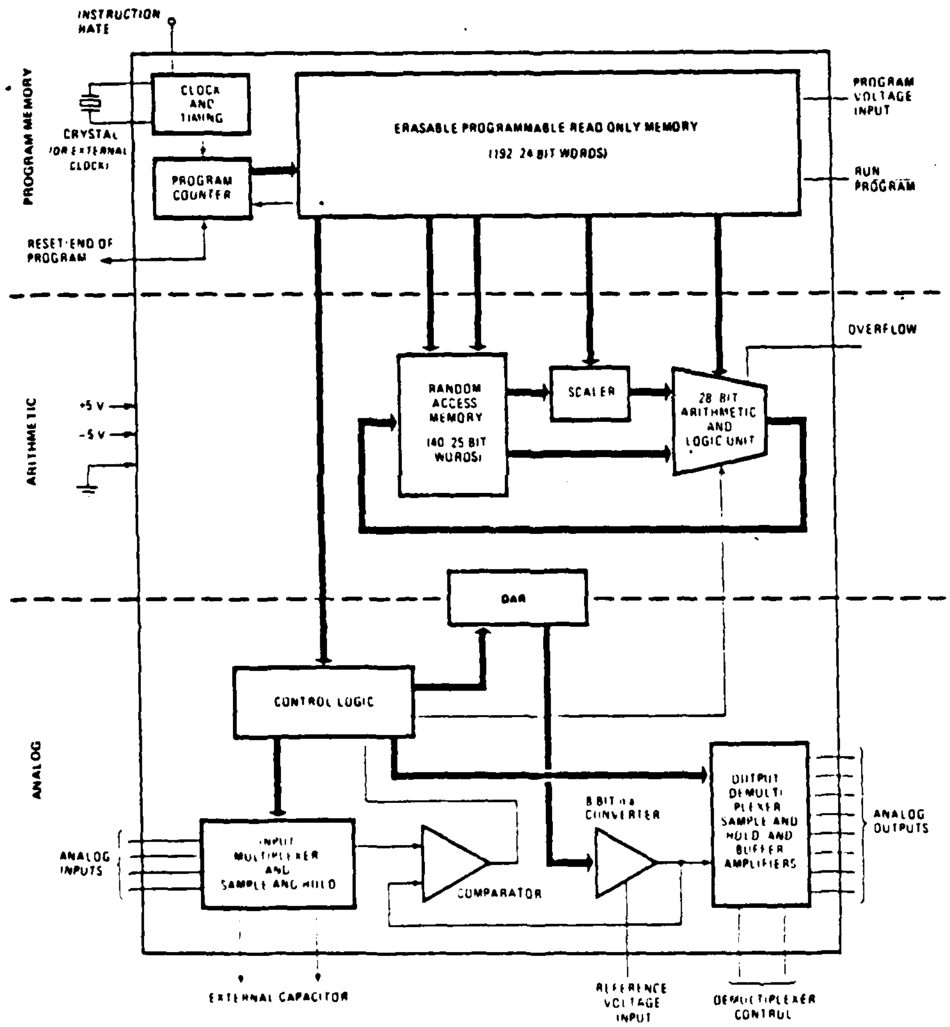


Figure 5.1 2920 Functional Block Diagram

Several logical conditions are allowed which affect program manipulation of data, but none will cause the processor to execute a program step out of sequence. In fact the only effective jump is performed at the last instruction which provides for a return to the beginning of the program loop. In this way the programmer may provide for an exact digital sample interval based upon program loop execution time. The shorter the program implementation loop the greater is the processor capacity to provide a faster sampling frequency.

The necessity of providing for an accurate sampling interval arises out of an understanding of the characteristics of the sampled analog signal being processed. Without an accurate clock interval, provided for in the 2920 by the program execution loop time, significant noise can be introduced in the system. Even small variations in the sampling interval can render the analysis useless through the introduction of intolerable measurement noise.

Each 2920 program instruction requires four clock cycles to execute. Given our nominal 6.666 MHz clock (the maximum allowed) the 2920 can therefore realize a maximum sampling frequency of 8.680 kHz over a 192 instruction program loop. This allows for a device bandwidth of greater than 4 kHz. Shorter programs naturally allow for a greater sampling frequency and thus higher device bandwidth.

V. THE INTEL 2920 ANALOG SIGNAL PROCESSOR

A. OVERVIEW

The INTEL 2920 Analog Signal Processor is actually a digital processor which is implemented to perform analog signal processing functions. Introduced in 1979, the 2920 system is centered about the 2920 single-chip microcomputer which is specially designed to process real-time analog signals. This single chip includes within its 28-pin DIP configuration sufficient hardware to provide 192 program memory locations, scratchpad memory, digital to analog (D/A) circuitry for up to four separate sampled inputs, analog to digital (A/D) capabilities for eight individual outputs, a digital pipeline processor capable of up to twenty-five bit accuracy, and input/output (I/O) control circuitry [Ref. 7: p. 3-1 through 3-2] . The 2920 is capable of implementing a wide variety of functions which rely upon sampled digital data techniques. We will use the 2920 to implement our matched filter design which will detect the APU start.

At the heart of the 2920's significant power is its on-board erasable programmable read-only memory (EPROM) which allows the user the convenience of customizing the 2920 for each intended application. Because the 2920 is a pipeline processor all program steps are performed sequentially without any conditions which may impact upon execution time.

filter. We shall see in the discussion which follows that the sampling frequency will be 8680 Hz. Thus our band of input frequencies is limited to less than one-half of this value, or 4340 Hz. Because our frequency of interest is 600 Hz we have considerable freedom in choosing the cut-off frequency of our anti-aliasing filter.

One option available to us is to design a low-pass filter with a rolloff which meets our needs. However, there are such filters commercially available which implement a compatible response which minimizes the effort required of the designer. One such filter is the INTEL 2912A which has been specifically included in the hardware kit we shall use to implement the digital filter we have just developed. This hardware implementation is the subject of Chapter 5.

output at 600 Hz is significantly less than at 590 Hz and even 575 Hz. This is indicative of both a steeper filter rolloff at frequencies greater than the center frequency and the effect of coefficient approximation which will be discussed more fully in the next chapter. The frequency response at both 500 Hz and 700 Hz is expectedly minimal but may not be usably low. If we find that the filter rolloff is not great enough and the response out of the passband is too great for our purposes then further design modifications may be undertaken. Accordingly, we could increase the order of our filter design. This would increase the number of filter stages in the analog implementation and therefore the complexity of that design. But, as we shall see, to a certain extent this additional filter complexity in the difference equation may be absorbed by the digital implementation we shall pursue without any increase in the hardware. These considerations will have to be examined more completely in the final analysis of the filter design effectiveness.

E. ANTI-ALIASING FILTER

When implementing a digital filter it is necessary to employ an analog input anti-aliasing filter to limit the band of input frequencies to less than half of the Nyquist sampling rate. This corresponds to the need to implement a low-pass filter at the input to the digital band-pass data to recreate the frequency response. This is shown in Figure 4.9. The figure confirms a narrow band-pass filter function with a center frequency at approximately 585 Hz. This is very close to the design center frequency of 591 Hz and is, in fact, within the error of a single bar in this pattern representation.

2. Digital Filter Frequency Response

Now that we have confirmed the stability of our filter design we can proceed to examine the frequency response of the filter over the range of interest. In particular we shall examine the filter frequency response over several frequencies in the range of 500 Hz to 700 Hz. The FORTRAN programs which allow this examination are S22F and S22FG which are included as Appendix G to this thesis. Due to the number of output figures they will be left in the appendix and we shall only give a summary of their content.

The digital filter frequency response was examined for the following frequencies: 500, 575, 590.825, 600, 625 and 700 Hz. The 590.825 simulation was chosen because this is the design center frequency (due to pre-warping) and we wish to confirm an output maximum amplitude at this frequency. From the figures in the appendix it is easy to see that the filter does in fact yield the response we desire. The maximum output amplitude does occur for the expected frequency, although the output at 575 Hz does not diminish appreciably from this value. However, we observe approximately one-tenth of a second represented by the duration of the overall sample period in the figure. To confirm this suspicion we can carry out the impulse response for a substantially longer period of time, say over one full second, or approximately 8192 iterations. The results of this computation are shown in the output of S22I in the appendix. They confirm the occurence of the maximum amplitude of impulse response output at the xxxth iteration which is what we observe in the figure.

Having realized the digital filter impulse response output we can perform a discrete Fourier transform of this

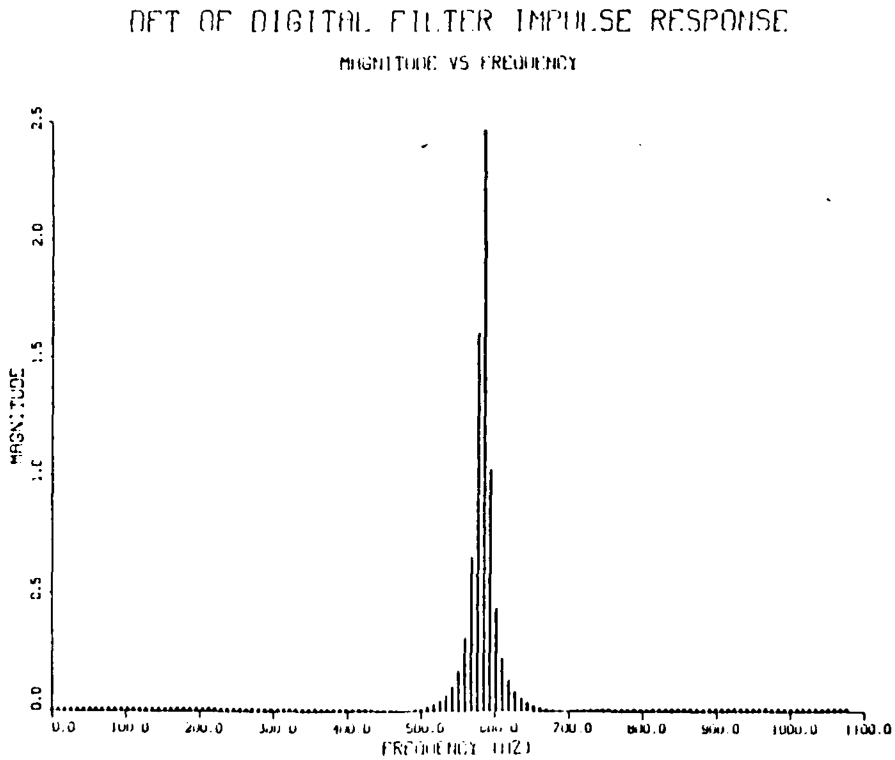


Figure 4.9 Discrete Fourier Transform of the Digital Filter Impulse Response

After the difference equation implementation in each program pass we are left with a binary value in the DAR register which corresponds to the program output for that pass. We have the option of providing a certain amount of linear output gain by an appropriate shift of the output binary value now in the DAR. In program step 132 we provide a gain of four by a left shift of two binary positions. We output this value to channel 0 in steps 139 through 142.

The final program manipulation occurs in program steps 143 to 150. Here we perform a serial register shift of present program pass values in preparation for the next program pass. Program step 191 is the final executable statement which returns us to step 0 for the next pass. The entirety of the 2920 operation consists of an endless loop of these instructions.

2. <u>2920 Hardware Implementation</u>

The 2920 contains an EPROM which is loaded with the hexadecimal code which is equivalent to the assembly language program just described. However, there are several other component devices which are integral to the operation of the 2920. The relation of these devices to the 2920 will now be described. A graphical schematic of these components appears in Figure 5.4.

At the input side of the 2920 an anti-aliasing filter is realized by using a 2912A which actually contains two filters which are cascaded together. This configuration

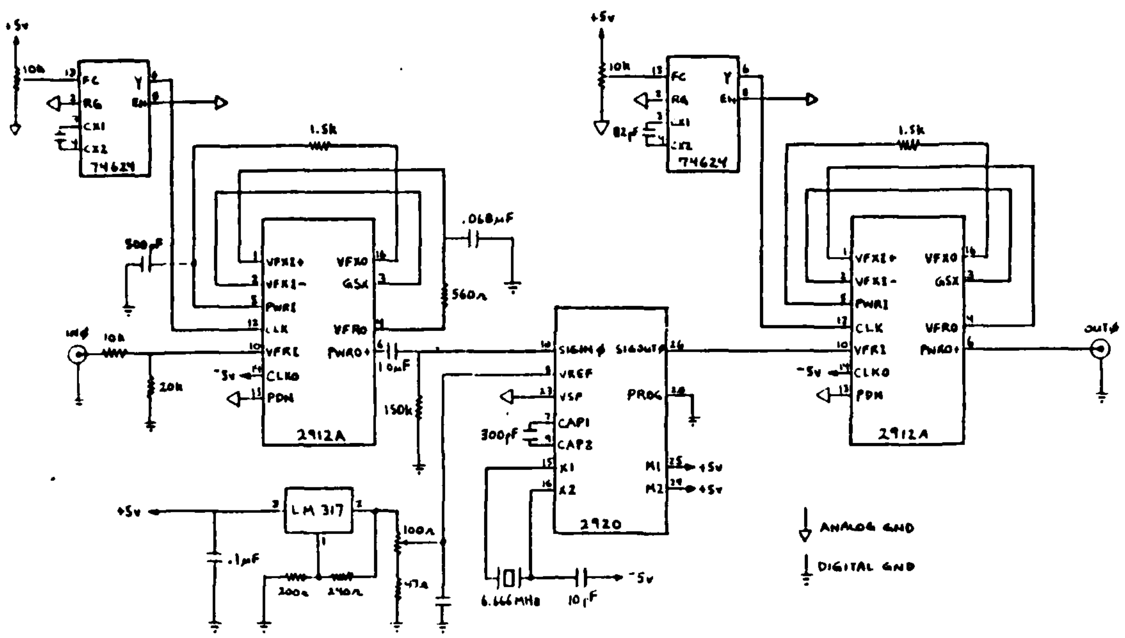


Figure 5.4 2920 Digital Filter Schematic

provides 54 dB of input dynamic range and a nearly flat response for frequencies less than 3 kHz. There a steep roll-off commences and at about 4 kHz the cascaded filter combination provides over 30 dB of attenuation. This supports the Nyquist frequency limit which is 4.34 kHz in this application.

The 2912A is a pulse code modulated filter which requires a clocking input to realize its filter function. This is provided by the 74624 at its input.

At the output of the 2920 another 2912A is employed in identical configuration and now provides a reconstruction filter for our application. This filter smooths the output of the 2920 to provide an analog signal for follow-on logic discrimination as shown earlier in Chapter 3.

A 2920 option which is not demonstrated here yet will be employed in final filter configuration is to obviate the need for external signal conditioning by allowing program discrimination of the output value and thus providing a processed TTL signal output. This eliminates the need for the external circuitry shown in Figure 3.8 and therefore represents one significant advantage of the 2920 digital design over the analog implementation.

E. 2920 DIGITAL FILTER IMPLEMENTATION RESULTS

We will now proceed to demonstrate the results of the 2920 digital filter implementation in much the same manner as the presentation which accompanied the analog filter design in Chapter 3. We begin with a photograph of the digital filter frequency response to a ramped sinusoidal input. This is shown in Figure 5.5. The same method was used to generate the sweep oscillation although the range of sweep is not identical to that employed in generating Figure 3.9. The result is that we cannot guarantee the narrow bandwidth of this digital filter relative to its analog counterpart by this means alone. The intent is, as before, only to demonstrate that a narrow band-pass filter response is generated.

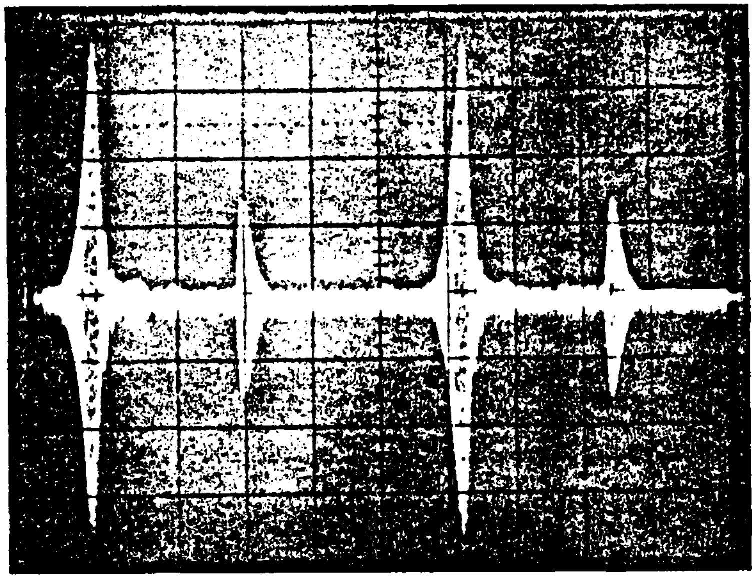
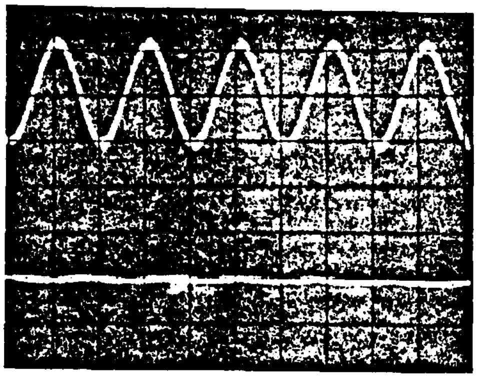
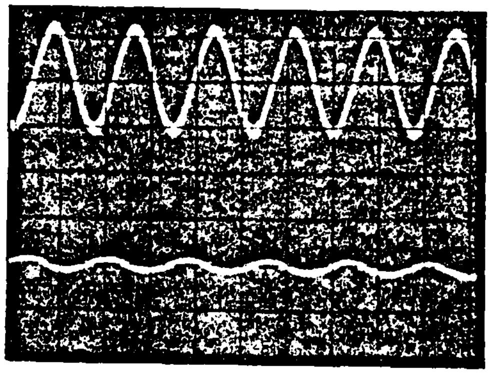


Figure 5.5 Digital Elliptic BPF Frequency Response (Photograph of Actual Filter Amplitude Response to a Ramped Sinusoidal Input)

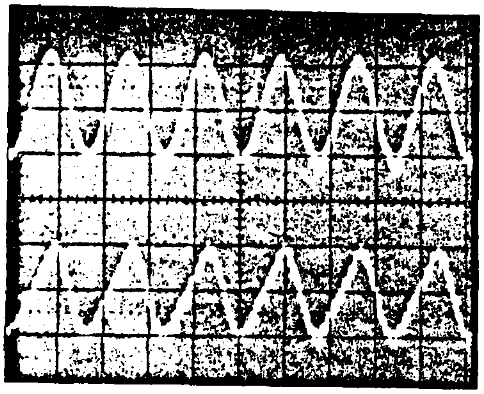
To confirm the operation at the desired band-pass center frequency we next apply discrete sinusoidal inputs to the digital filter at various frequencies arrayed about 600 Hz. The result is the digital analog to Figure 3.10 which is shown here as Figure 5.6. The scale is maintained as in Figure 3.10. The input frequencies are at about double the amplitude of the analog filter to ensure proper operation. This implies that despite the relative immunity of the digital filter to input amplitude variations we must nontheless provide an input above approximately 100 millivolts peak-to-peak. However, once above this threshold



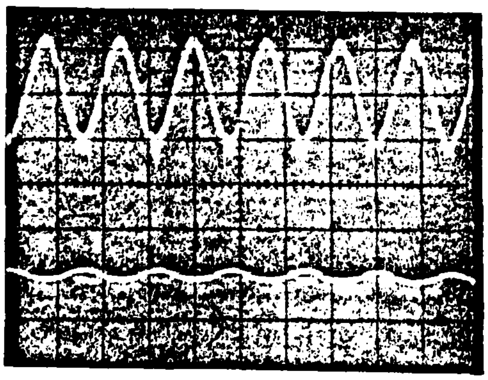
a) 500 Hz



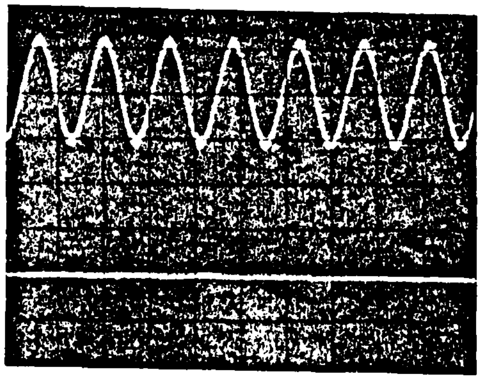
b) 575 Hz



c) 600 Hz



a) 625 Hz



b) 700 Hz

Figure 5.6 Digital Elliptic BPF Frequency Response
Upper trace (Input): 50 mV/div scale
Lower trace (Output): 1.0 V/div scale

value the digital filter provided a relatively undistorted and largely constant amplitude output up to an input amplitude of over 5 volts peak-to-peak (and this despite the 1 volt reference level of the input). Figure 5.6 thus confirms the center frequency maximum at a value near 600 Hz and a steep roll-off on either side of this value.

VI. ALTERNATIVE FILTER CONCEPTS

The preceding development was based upon techniques used in implementing an Infinite Impulse Response (IIR) digital filter. Simply stated, an IIR filter realizes its output based upon the values of all present and previous inputs and outputs. In other words, feedback is employed in an IIR design.

In the general case, an IIR filter will have M finite zeroes and N finite poles. The zeroes of H(z) can be anywhere in the z-plane but the poles must lie within the unit circle to guarantee stability. In the case we have developed, a digital filter realization derived from an analog design, the order of M must be less than or equal to N. This describes an Nth order digital filter.

The hardware implementation of an IIR design usually involves the cascading of elemental single pole filters with double complex pole filters. These elements are derived from the original transfer function using a partial fraction expansion separation scheme.

There are other methods for realizing the filter we desire other than the a priori scheme we have developed so far. These generally use the input signal itself as a basis for the filter transfer function coefficients and involve an adaptive evaluation of the proper coefficients which yield the desired filter response.

A. A WIENER FILTER DESIGN--THE ADAPTIVE LINEAR COMBINER

The Adaptive Linear Combiner (ALC), shown in Figure 6.1, forms the basis for the Adaptive Filter design we shall now discuss [Ref. 12]. An input analog signal may be digitally sampled in accordance with the Nyquist criterion and we may then apply N sequential elements of that sample block to

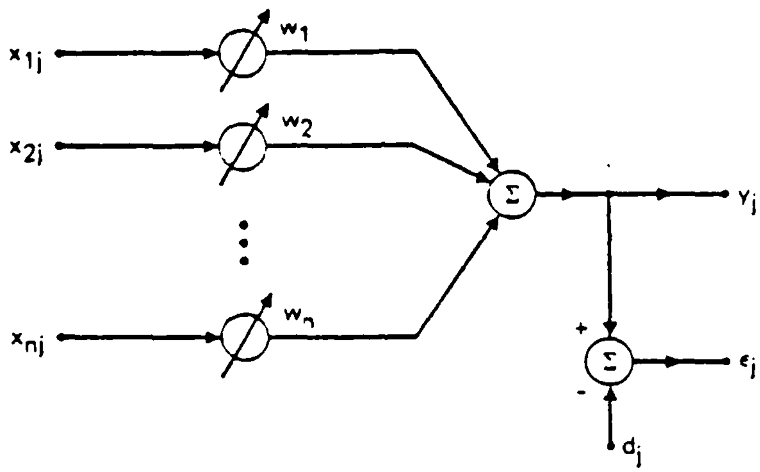


Figure 6.1 The Adaptive Linear Combiner

the ALC inputs. These inputs can be easily derived from a tapped delay line which cascades sample values along in sequential storage for processing. This scheme lends itself well to implementation in a processor such as the 2920 which is designed to accept sequential values by way of its component A/D converter and RAM storage.

The set of measurements x_{nj} is multiplied by a corresponding weighting term W_i, and the results then summed to yield the output y_j. This output is then compared to a desired signal value for that instant and the difference between them constitutes an error signal ϵ_j. The objective of the ALC is to determine W_i so as to minimize ϵ_j for each set of sampled inputs and thus realize the weighted sum of input signals that best matches the desired response.

1. <u>Theoretical Foundations</u>

At the nth instant of time the output of the Non-Recursive Wiener ALC, y(n), is given by [Ref. 13]:

$$y(n) = \sum_{j=0}^{N} x(n-j)W_j$$

$$= W_0 x(n) + W_1 x(n-1) + \ldots + W_N x(n-N)$$

which may be written in matrix form as

$$= \underline{W}^T \underline{X}$$

or equivalently

$$= \underline{X}^T \underline{W}$$

where T represents the matrix transpose operator, the set of N+1 weights is denoted by

$$\underline{W}^T = [W_0 \; W_1 \; \ldots \; W_N]$$

and the set of present and N previous inputs is given by

$$\underline{X}^T = [x(n) \; x(n-1) \; \ldots \; x(n-N)]$$

The error signal e(n) for time n is given by

$$e(n) = d(n) - y(n)$$

$$= d(n) - \underline{W}^T\underline{X}$$

and the square of the error (using the latter matrix notation) by

$$e^2(n) = e(n) \bullet e^T(n)$$

$$= [d(n) - \underline{W}^T\underline{X}][d(n) - \underline{X}^T\underline{W}]$$

$$= d^2(n) - 2d(n)\underline{X}^T\underline{W} + \underline{W}^T\underline{X}\underline{X}^T\underline{W}$$

The mean square error, obtained by taking the expected value of this last equation, is given by [Ref. 13]

$$E[e^2(n)] = E[d^2(n)] - 2E[d(n)\underline{X}^T]\underline{W} + \underline{W}^T E[\underline{X}\underline{X}^T]\underline{W}$$

Defining the vector Φ_{xd} as the cross-correlation between d(n) and $\underline{X}$ then yields

$$\Phi_{xd} \equiv E[d(n)\underline{X}]$$

$$= E[d(n)x(n), d(n)x(n-1), \ldots, d(n)x(n-N)]^T$$

The input auto-correlation matrix Φ_{xx} is defined as

$$\Phi_{xx} = E[\underline{XX}^T]$$

which may be written in expanded notation as

$$= \begin{bmatrix} x(n) \\ x(n-1) \\ x(n-2) \\ \vdots \\ x(n-N) \end{bmatrix} [x(n) \ x(n-1) \ \ldots \ x(n-N)]$$

Now if we carry out the indicated vector multiplication we arrive at the following result [Ref. 13]

$$= \begin{bmatrix} x(n)x(n) & x(n)x(n-1) & \cdots \\ x(n-1)x(n) & x(n-1)x(n-1) & \cdots \\ \vdots & \vdots & \\ & & x(n-1)x(n-1) \end{bmatrix}$$

And thus we arrive at the following form of the input correlation matrix

$$
= \begin{bmatrix} \varphi_{xx}(0) & \varphi_{xx}(-1) & \cdots & \varphi_{xx}(-N) \\ \varphi_{xx}(1) & \varphi_{xx}(0) & \cdots & \varphi_{xx}(1-N) \\ \vdots & \vdots & & \vdots \\ \varphi_{xx}(N-1) & \varphi_{xx}(-N) & \cdots & \varphi_{xx}(-1) \\ \varphi_{xx}(N) & \varphi_{xx}(N-1) & \cdots & \varphi_{xx}(0) \end{bmatrix}
$$

In order to find the optimal weight vector, $\underline{W}^*$, we can differentiate the mean square error function with respect to the weight vector $\underline{W}$ to yield

$$
\frac{d(e^2(n))}{d\underline{W}} = -2[\Phi_{xd} - \Phi_{xx}\underline{W}]
$$

The optimal weight vector , $\underline{W}^*$, generally called the Wiener weight vector, is obtained by setting the quantity in brackets equal to zero. This results in

$$
\underline{W}^* = \Phi_{xx}^{-1}\Phi_{xd}
$$

The objective of processes involving the ALC is to find a solution to this equation. In fact we may employ an adaptive algorithm which uses the error signal, $\epsilon(n)$, (generated for each instance of filter inputs), as the basis for modifying the filter weights until a minimum error is attained for a particular input block. This describes the Adaptive Transversal Filter shown in Figure 6.2 [Ref. 12].

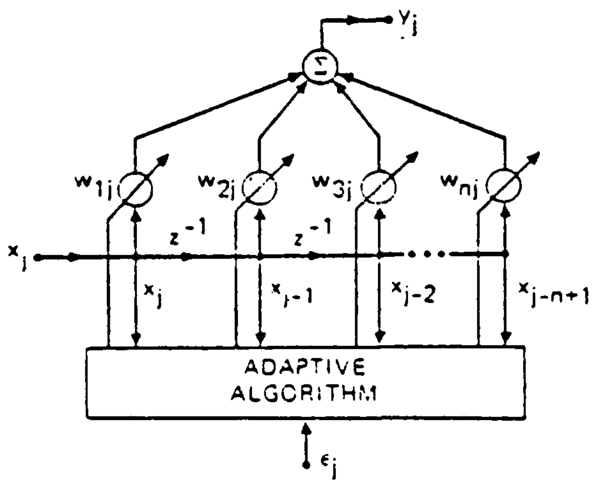


Figure 6.2 The Adaptive Transversal Filter

The Adaptive Transversal Filter (ATF) is a Finite Impulse Response (FIR) filter owing to the lack of direct feedback from output to input. If we employ a tapped delay line at the input to the ALC which comprises the ATF the form of the input vector becomes a finite number of delayed elements of the input signal. It is therefore easy to see that the impulse response of the ATF is just the sequence corresponding to the elements of the weight vector, $\underline{W}$. Such a filter can have any impulse response of length less than or equal to its own length. Allowing for an ideal unlimited length we could realize any impulse response at all, and thus any frequency response. Practically, however, we are limited by filter complexity, error due to misadjustment, and an adaptive time constant which corresponds to filter length.

Thus we have a means of generating the desired filter response by applying the very signal we wish to detect. If we apply a digital series of samples taken from an analog reference signal we can realize the filter weights which will provide our desired signal output stream at a later time.

Thus the idea is to sample the analog recording of the APU noise in the cargo bay prior to launch and to apply that input series of data elements to an ATF to realize the filter weights. We may then build a 2920 circuit which uses these weights as filter coefficients to provide our filter response.

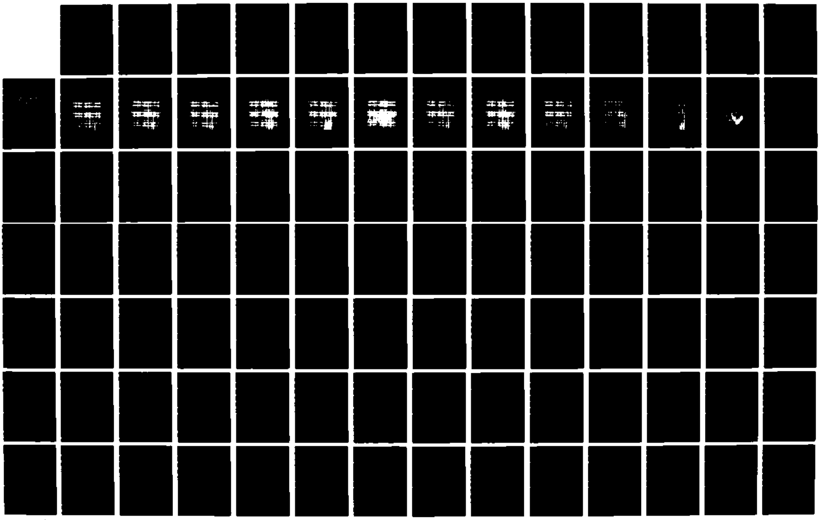
2. A Software Simulation

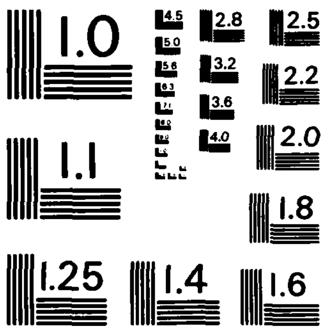
As an example of this methodology we will now present an elementary simulation which was performed for an input which consisted of an equal amplitude application of the three fundamental frequencies of interest: 600 Hz, 1200 Hz and 1800 Hz. We chose to simulate an Adaptive Transversal Filter of fourth order which therefore consists of four weights.

One example of a software implementation which is designed to arrive at the four desired filter weights is

AD-A158 960 A MATCHED FILTER ALGORITHM FOR ACOUSTIC SIGNAL DETECTION(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D W JORDAN JUN 85 2/3

UNCLASSIFIED F/G 17/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

shown in Appendix J as FORTRAN program FIR4. In this program we begin with trial weights and a range of upper and lower bounds. By repeated application of a library coefficient optimization routine (BOXPLX--also included in the appendix) we arrive at a set of four optimal weights within the bounds specified.

The result of this simulation is revealed through application of the FORTRAN program FIR4SIM which is included as Appendix K to this thesis. This result is shown in Figure 6.3. The input analog signal (indicated by the solid line) is a portion of the combined signal corresponding to

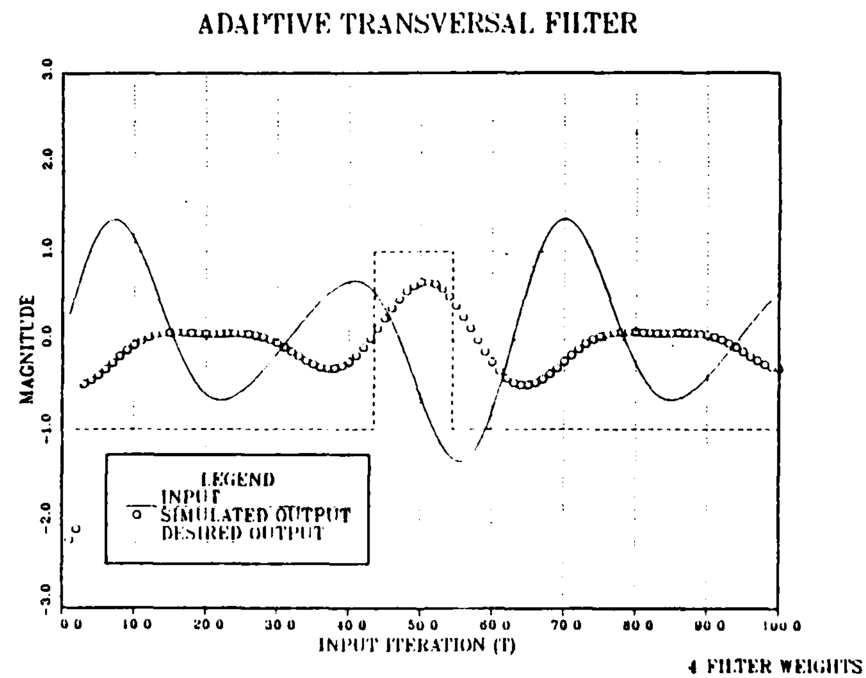


Figure 6.3 An Adaptive Transversal Filter Simulation

the three fundamental frequencies mentioned previously. The desired output (shown by the dashed line) is chosen to be a continuous -1 unless the signal of interest is detected. In that case the desired output jumps to +1.

As indicated in Figure 6.2 the Adaptive Algorithm samples the input signal and uses the successive present and three previous samples to arrive at the desired filter weights which will accomplish the task of signal discrimination. In the algorithm implemented by FIR4 of Appendix J we arrived at the following filter weights.

$$w_1 = -7.5060358$$
$$w_2 = 7.5403662$$
$$w_3 = 4.7097464$$
$$w_4 = -5.3987589$$

In Figure 6.3 we see that over 100 sample output iterations these weights resulted in an output which was at or near zero or below with a significant rise above 0.5 near the desired region. This approximates the filter response which would allow a useful discrimination of the desired signal by detection above a threshold floor (say 0.5 in this example).

This is by no means intended to be an exhaustive discussion of tnis approach to a matched filter design, but merely a consideration of an alternative approach which might be taken to realize a useful filter.

B. AN ANALOG SPEECH PROCESSING SCHEME

A further alternative which may be considered involves the use of commercially available speech processing microcircuits which often use Linear Predictive Coding schemes as the basis for their discriminant filters.

The current state of the art in speech recognition technology does not permit even the most sophisticated (and large) devices to recognize but several hundred words of vocabulary. The breakthroughs are most often in the arena of overcoming the speaker dependent nature of the simpler systems. However, all systems, be they single chip processors or multi-cabinet devices, do have the capability to analyze an audio input signal (conventionally this is speech of course) and to characterize the nature of changes in the formant composition over time.

Because the signal we wish to identify is in the audio spectrum it seems a logical idea to consider that a speech recognition device may prove usable for our purposes. In fact, Interstate Electronics Corporation now markets a single chip voice recognition device (VRC008) which is capable of reliable and independent recognition of up to eight words or phrases which are stored in its vocabulary. While this may seem a minimal vocabulary it is a remarkable ability for a single chip device.

The Interstate VRC008 is capable of being trained to recognize words or phrases of up to 1.2 seconds duration.

To implement an audio signal recognition scheme would require that we somehow "sample" our input audio environment in discrete blocks of about one second apiece. Thus we would simulate a discrete utterance which could be processed by the circuit. In the absence of the characteristic APU signal the device would register no recognition of the input sample. But after APU ignition it is reasonable to assume that the device would treat the APU signal as a recognizable "word" it had previously been taught.

While this approach may seem at first to be a promising one we must also consider the drawbacks, especially in view of the technically simpler approaches we have reviewed so far. First is the cost. Although the Interstate chip is by itself a relatively inexpensive device (on the order of ten dollars in quantity), by itself it is also useless without a twenty-five thousand dollar training and development tool. The intent of the manufacturer is that the cost of the development tool will be amortized by the consumer over a sizable run of usable end devices. Our application does not lend itself to mass production and the cost therefore becomes prohibitive. This is especially true in view of the cost of the simpler technologies discussed in previous chapters which have given us useful designs.

Another drawback of a speech recognition approach is the level of technical complexity versus guarantee of successful results. Unless we use a device of modular circuit size or smaller we risk an excess of power and space consumption. But the state of the art in speech recognition is such that accuracy of recognition is roughly related to the size of the device (although it is directly the vocabulary size which is the truly overwhelming factor here). The Interstate VRC008 claims only an 85 percent accuracy of recognition which is low by the standards of other speech recognition devices. This is the price one pays for small size.

The important point is that our signal of interest is characterized by extremely well-defined and stationary spectral components. This fact allows the use of cheaper and more traditional methods of signal processing and filter design. Were our signal of a rapidly time varying nature then a purely analog approach would be impossible, and even a digital approach would prove difficult if not infeasible. It is then that methods of linear predictive coding which form the basis of speech recognition would become one of a very few viable alternatives.

VII. CONCLUSION

In this thesis I have considered several approaches to the problem of designing a matched filter for the detection of the acoustic signal which characterizes the Shuttle Auxiliary Power Unit. The Analog and Digital IIR filter approaches were treated in some detail, while the Weiner FIR and Voice Recognition methods were given less attention. My purpose was not to present an exhaustive treatise on the subject of filter design, but rather to describe various ways in which a particular problem might be approached.

It is not coincidental that the order of presentation of the considered methods should conform to the chronological introduction of these sciences to the engineering community. As may be expected, the facility with which each of these methods is employed is proportional to their general familiarity among engineers. The analog approach considered first is the best established method of filter design. Not surprisingly, this method is supported by a wealth of literature. Despite this ample documentation, at best the analog approach to filter design is an inexact science which is largely dependent upon the degree to which one is able to characterize the signals we wish to manipulate. Often, however, we have excellent knowledge of these signals, and thus the analog approach to filter design remains a completely reasonable and certainly cost effective approach to simple filter designs.

The APU signal of concern to this study was such a signal. Its signature was stationary over time and could be reliably found at amplitudes well above the noise threshold. The dominant component at 600 Hz was of quality sufficient to preclude examination of sub-dominant spectral harmonics at higher frequencies. The fact that a well-defined signal was evident allowed for a design which emphasized the simplicity of the analog approach.

Mention should be made of the obstacles which did impede the final analog design. Because an analog filter serves only to attenuate those signal component spectral elements out of the passband, but does not eliminate them, it is necessary to know the range of amplitude which may be expected of the sensor microphone output. For a given amplitude of signal input which varies little within the range of input frequencies it may be reliably expected that the analog bandpass filter would reject the frequency components outside of the narrow passband. But if the spectral components were grossly disparate in their amplitude and a component out of the passband were received which was significantly above the amplitude expected of the 600 Hz center frequency, then it is possible that the component out of the passband would be passed regardless of the filter attenuation. This demonstrates the need we have to know the nature of the input signal.

One approach to this problem is to increase the attenuation of the filter. But this does not guarantee signal component rejection out of the passband. The solution for an analog approach lies in Automatic Gain Control (AGC) at the sensor microphone input to the filter. In this manner we can ensure a dynamic range of input signal which is within the limits of filter discrimination. This implies a careful selection of a microphone and preamplifier combination which in turn implies a similarly careful understanding of the dynamic range of the input signal. Indeed, these considerations continue to be the most vexing aspects of a useful final design. The actual dynamic range of the signals recorded on tape was unreliable due to the number of intermediate and indeterminate dubs which the tape underwent prior to our acquisition of a copy.

Furthermore, and even more importantly, the ultimate placement of the sensor microphone in the Shuttle cargo bay will have considerable effect upon the nature of the signal available for discrimination. It will also tell significantly on the dynamic range. This factor will impact upon any chosen filter design regardless of the algorithm selected. Thus in the analog case we must design for a wide dynamic range and provide AGC which yields a narrower range of amplitude input into the filter.

Much of this problem is overcome with the digital filter implementation developed in Chapters 4 and 5. At the foundation of the digital design is a frequency domain scheme whose output is less dependent upon input amplitude variations than the frequency components of the input signal. In fact using an EPROM based filter design such as afforded by the INTEL 2920 we enjoy considerable flexibility in tailoring the range of allowable inputs and outputs through careful selection of program parameters. The limitations are rather imposed by the noise level at the low end and the power limit at the high end.

There are several drawbacks to the digital filter which bear mentioning. The foremost drawback is cost relative to the analog filter. The design presented in Chapter 5 was dependent upon the SDK-2920 Development Kit which is a thousand dollar item. This is the minimum hardware which is necessary to develop a 2920 signal processing design. However, to support any sort of a sophisticated development requires the INTEL Intellec Development System with associated software. This quickly elevates the expense of the system to a range of tens of thousands. Of course there are certainly more uses for the Intellec system than simply a 2920 development application, so this expense can be amortized over those additional uses. But the 2920 applications software which supports the Intellec system is a four thousand dollar expense by itself.

This fact proved significant to the digital design when the simulator software was found to have a bug in it. When the original disks could not be located it was then deemed more practical to develop an application specific simulation on a mainframe computer instead of purchasing a replacement package from INTEL. This meant an additional expenditure of time of course, and was only successful in showing the adequacy of the specific 2920 filter implementation algorithm. However, without the 2920 Simulator software package effective troubleshooting was made significantly more difficult. Nontheless, as indicated in the results of Chapter 5, a successful 2920 implementation was accomplished without a fully healthy simulator. With it the design process would have been considerably more efficient.

An additional consideration is the complexity of the digital design over the analog approach. This is due in large part to the availability of resources which support an analog design relative to the novel approach represented by a state-of-the-art signal processing chip. However, the complexity of a signal processor application is often far outweighed by the considerable flexibility which it provides. One must not forget the power of the 2920 (witness its ability to incorporate all of the hardware elements of the follow-on logic circuitry required in the analog design in but a dozen or so lines of 2920 assembly language code) and weigh this against the short term inconvenience of having to become acquainted with a new approach. Once mastered the significant ability of a signal processing device make an analog approach to any complex filter design seem archaic. In addition the fewer actual circuit components required in an EPROM based device means significant savings in power consumption. This is an especially noteworthy item when considering an electronic device for a space application.

For the purposes of this thesis I must admit that the 2920 was certainly fun to work with. The literature is sketchy in spots and several calls to technical support at INTEL were needed to resolve some issues and errors. But overall the 2920 certainly provides the researcher with a significant amount of flexible and powerful signal processing ability.

The significant advantage of implementing a digital filter over the analog design is the relative immunity to the variations in input amplitude. This was a crucial consideration in the development described in Chapter 5 and by itself would account for the choice of the digital design over the analog approach. When coupled with the further advantages of lower power consumption, less physical space required and considerable flexibility in accommodating future changes without the need for hardware modification, the digital approach implemented in a powerful signal processor becomes an irresistable filter design option.

In Chapter 6 we considered two other approaches to the APU signal detection problem. Unfortunately a lack of time prohibited a serious examination of these additional approaches. Both are well-founded and represent the leading edge of signal processing technology. Given a requirement for detection of a more complex signal than we considered in this paper, these latter methodological options could well represent the only viable means of processing a time-varying signal in the acoustic spectrum.

APPENDIX B

POWER SPECTRAL DENSITY PLOTS OF THE SHUTTLE CARGO BAY PRE-LAUNCH ACOUSTIC ENVIRONMENT

<u>Legend for the Graphical Output on the Following Pages</u>

Shuttle Flight Number: STS-2, STS-3 or STS-4

Microphone Identification: 9405, 9219 or 9403

Sampled Interval Relative to APU Power-Up: PRE or POST

All PSD Sources are from the Original Aerospace Tape Copies (labelled ORIG)

Narrow Band Analysis (N=40 Samples)

Hanning Weighting

5.0 Volt RMS Front End Limiter

Gain: 10 dB per division

Cursor Point Label: X (Hz) and Y(B) (Engineering Units)

Scale: Linear Ordinate (0-2000 kHz)
Logarithmic Abcissa (10^1 to 10^8 EU)

RUN IDENTIFICATION APU HOTFIRE CHANNEL 10 STS-2

PARAMETER D0281A Noise Floor

OVERALL RMS VALUE 0.1905 G's

ENGINEERING UNITS 30.0000 G's

START TIME: (HR: MIN: SEC) 258/12:04:20

SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81

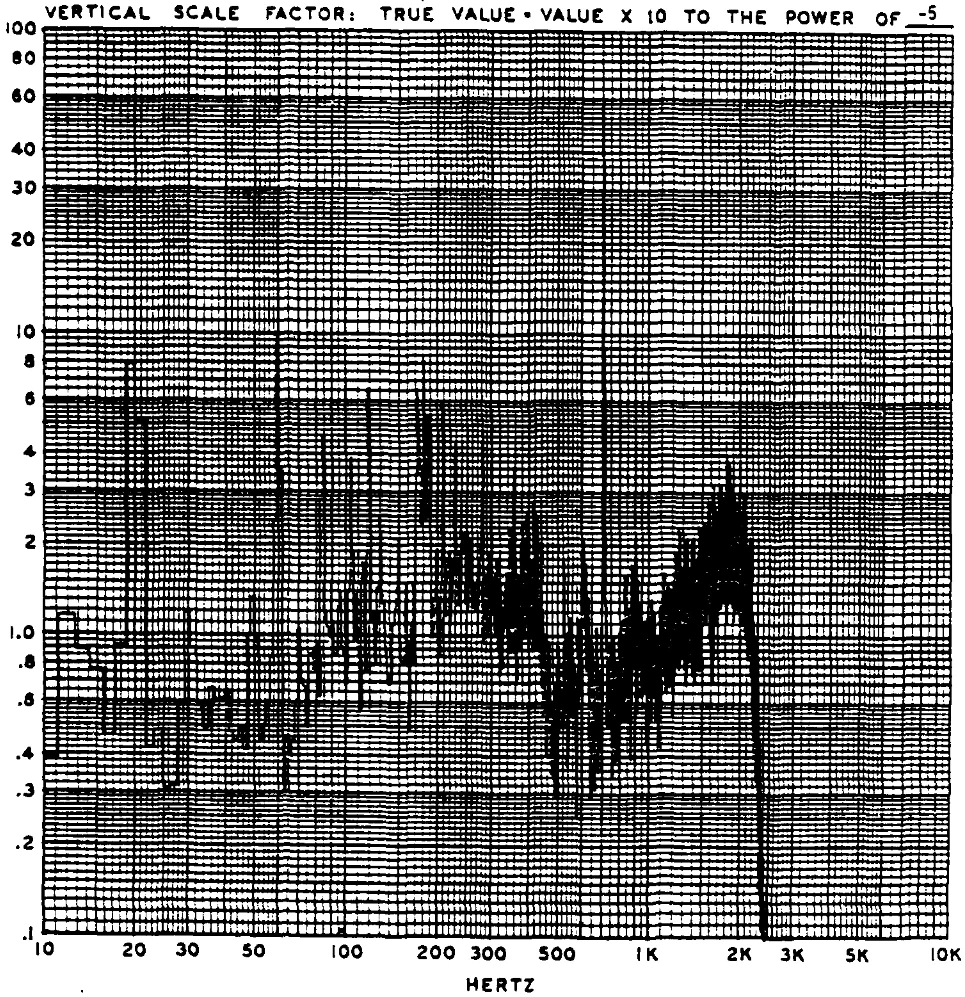
ANALYSIS B W 2.2467 Hz

ANALOG L.P. FILTER B W 2300.0000 Hz

DEGREES OF FREEDOM 9.0000

PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -5



RUN IDENTIFICATION APU HOTFIRE CHANNEL 10 STS-2

PARAMETER 00281A

OVERALL RMS VALUE 2.7251 G's

ENGINEERING UNITS 30.0000 G's

START TIME: (HR: MIN: SEC) 258/12:03:20

SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81

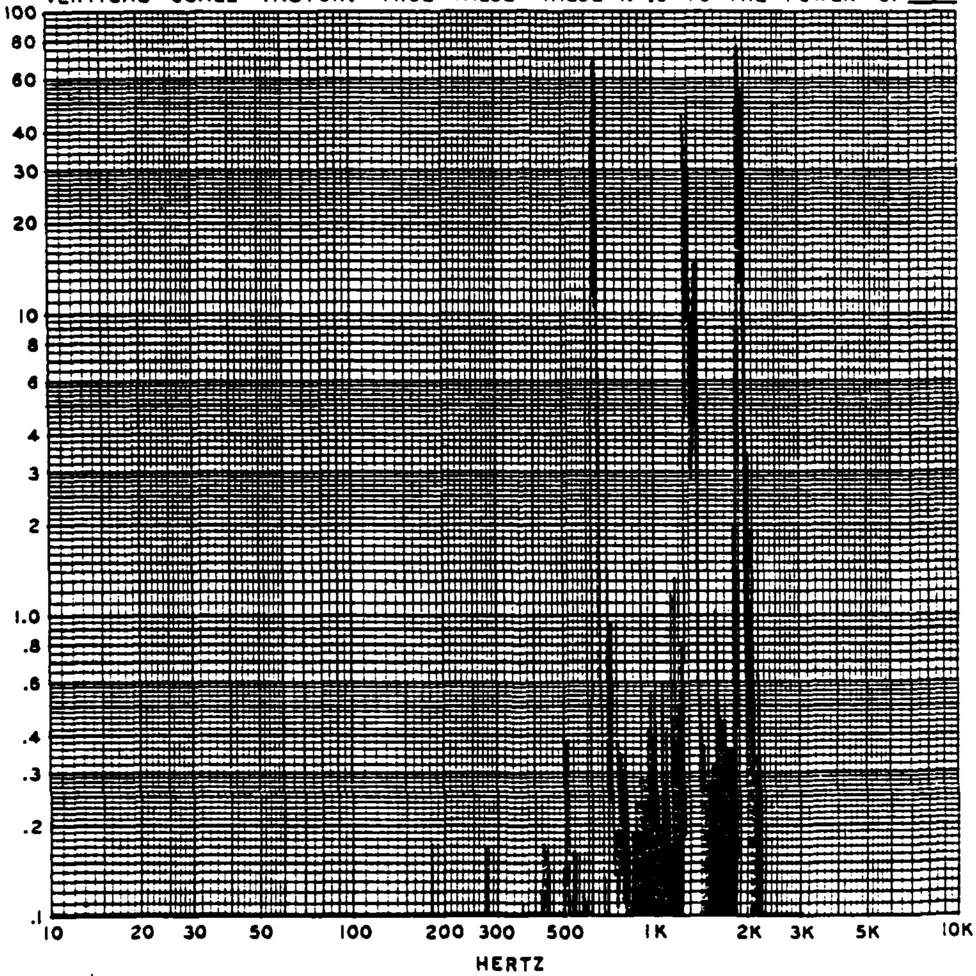
ANALYSIS B W 2.2467 Hz

ANALOG L.P. FILTER B W 2300.0000 Hz

DEGREES OF FREEDOM 9.0000

PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -3

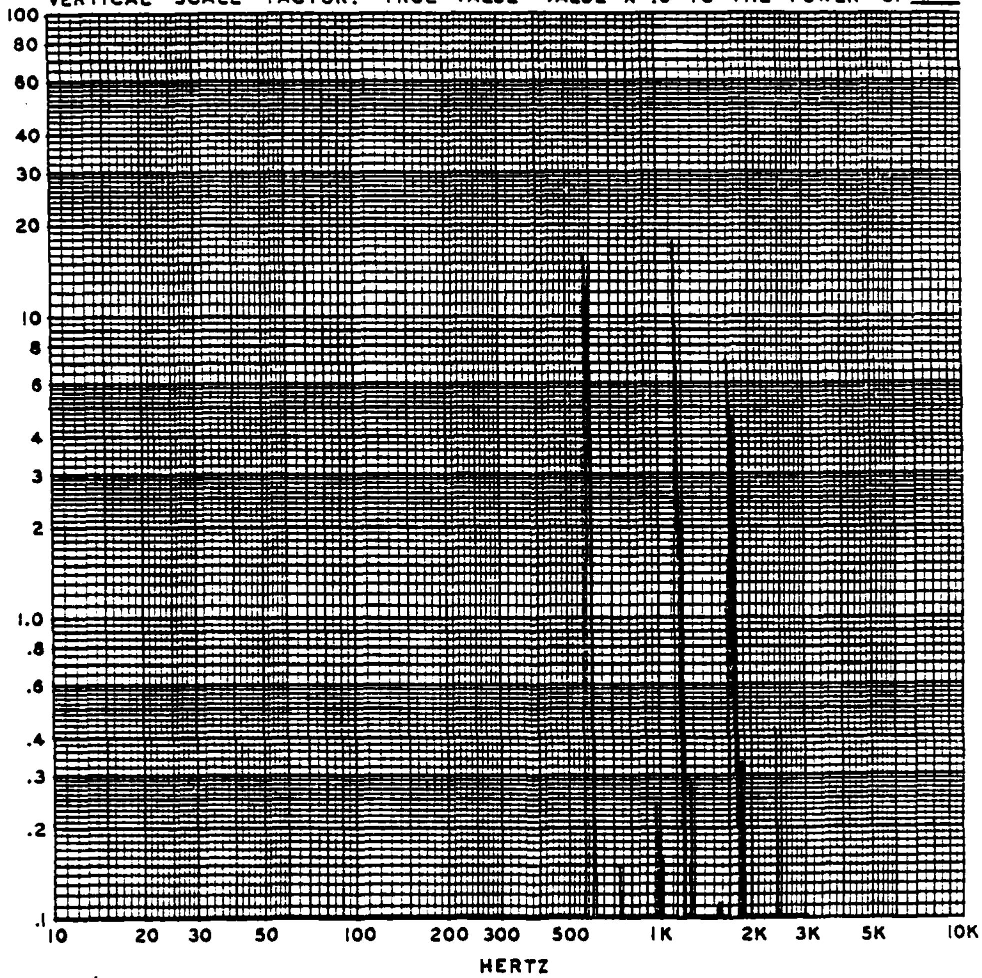


RUN IDENTIFICATION APU HOTFIRE CHANNEL 10 STS-2

PARAMETER D0281A
OVERALL RMS VALUE 2.9631 G's
ENGINEERING UNITS 30.0000 G's
START TIME: (HR: MIN: SEC) 258/12:01:22
SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81
ANALYSIS B W 2.2467 Hz
ANALOG L.P. FILTER B W 2300.0000 Hz
DEGREES OF FREEDOM 9.0000
PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -2



RUN IDENTIFICATION APU HOTFIRE CHANNEL 10 STS-2

PARAMETER D0281A

OVERALL RMS VALUE 6.4787 G's

ENGINEERING UNITS 30.0000 G's

START TIME: (HR: MIN: SEC) 258/11:56:35

SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81

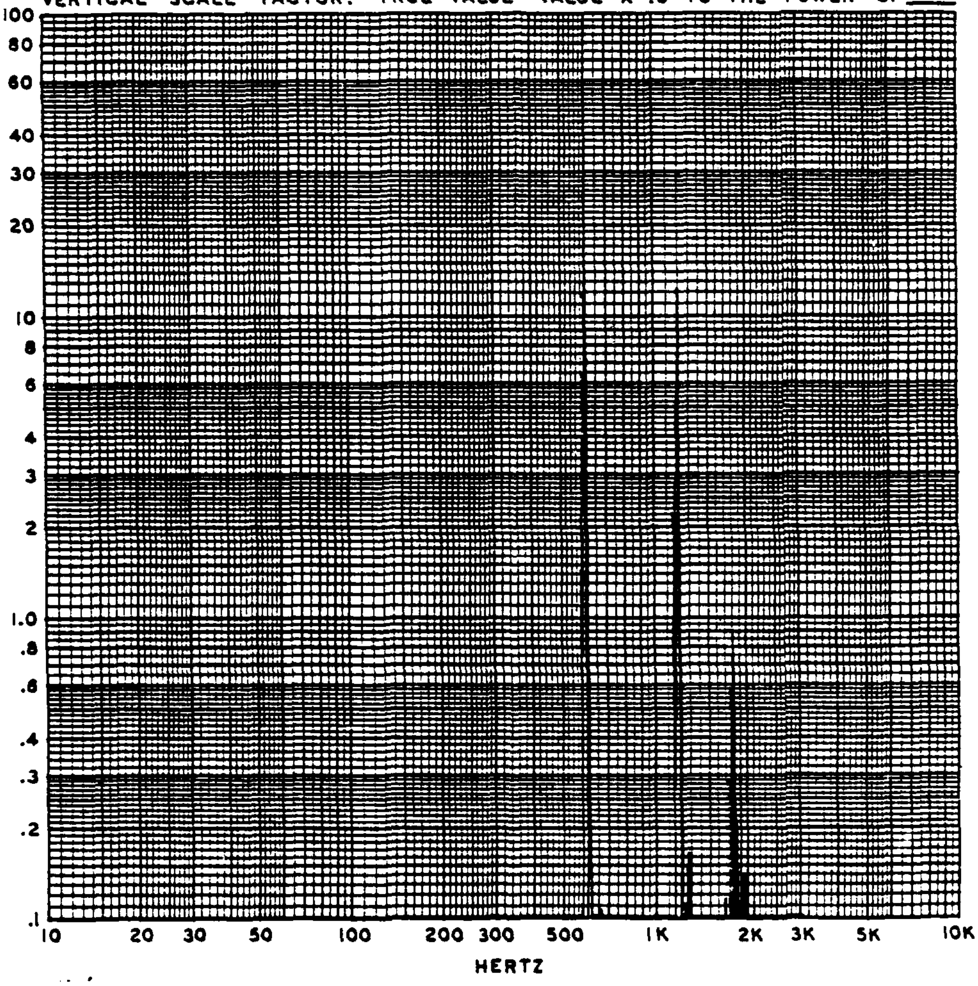
ANALYSIS B W 2.2467 Hz

ANALOG L.P. FILTER B W 2300.0000 Hz

DEGREES OF FREEDOM 9.0000

PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -1



RUN IDENTIFICATION APU HOTFIRE CHANNEL 10 STS-2

PARAMETER D0281 A

OVERALL RMS VALUE 3.6288 G's

ENGINEERING UNITS 30.0000 G's

START TIME: (HR: MIN: SEC) 258/11:58:14

SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81

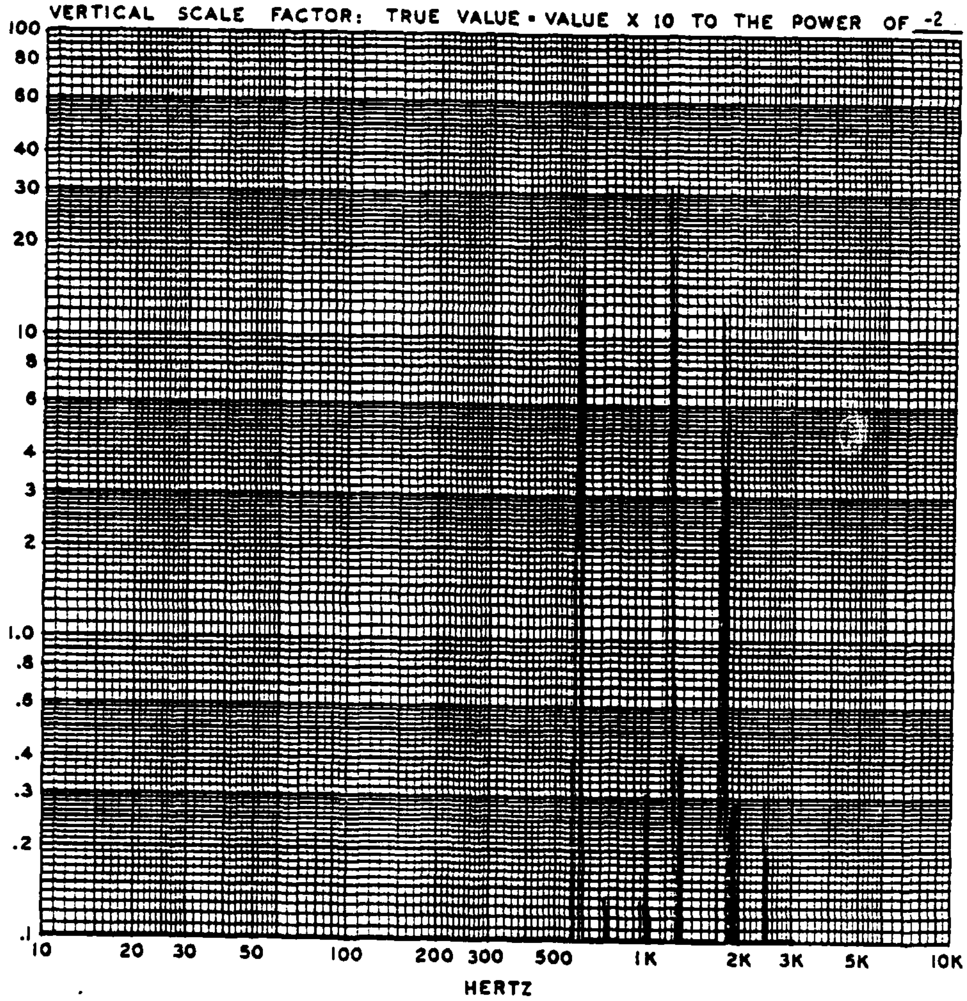
ANALYSIS B W 2.2467 Hz

ANALOG L.P. FILTER B W 2300.0000 Hz

DEGREES OF FREEDOM 9.0000

PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -2

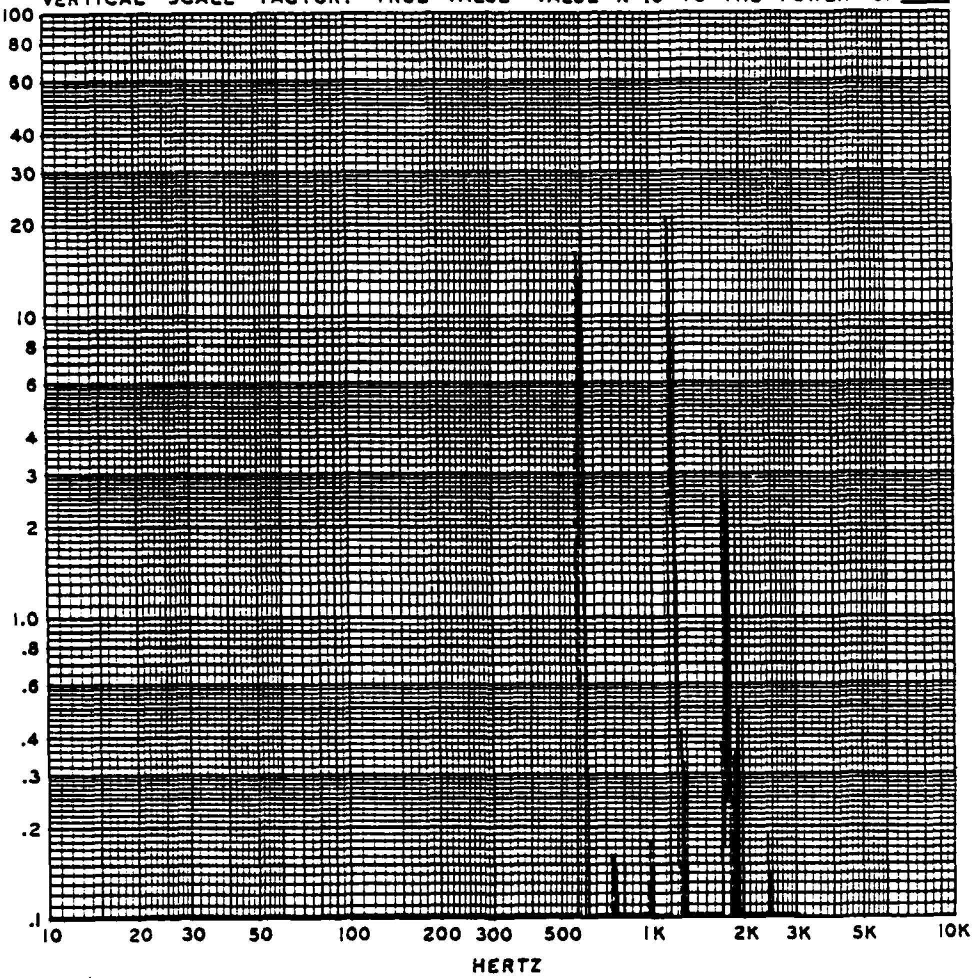


RUN IDENTIFICATION APU HOTFIRE CHANNEL 10 STS-2

PARAMETER D0281A
OVERALL RMS VALUE 3.1738 G's
ENGINEERING UNITS 30.0000 G's
START TIME: (HR: MIN: SEC) 258/11:57:46
SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81
ANALYSIS B W 2.2467 Hz
ANALOG L.P. FILTER B W 2300.0000 Hz
DEGREES OF FREEDOM 9.0000
PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -2

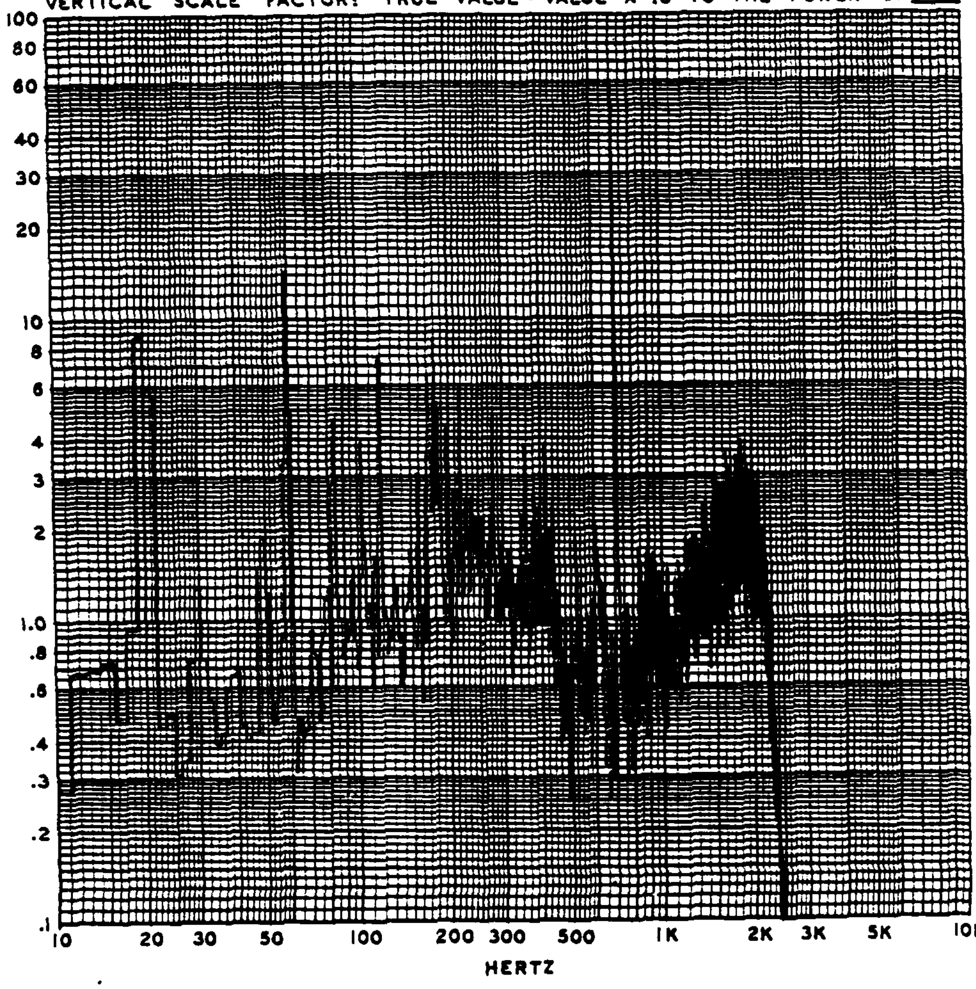


RUN IDENTIFICATION APU HOTFIRE CHANNEL 11 STS-2

PARAMETER D0280A Noise Floor
OVERALL RMS VALUE 0.1920 G's
ENGINEERING UNITS 30.0000 Gs
START TIME: (HR: MIN: SEC) 258/12:04:20
SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81
ANALYSIS B W 2.2467 Hz
ANALOG L.R FILTER B W 2300.0000 Hz
DEGREES OF FREEDOM 9.0000
PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -5

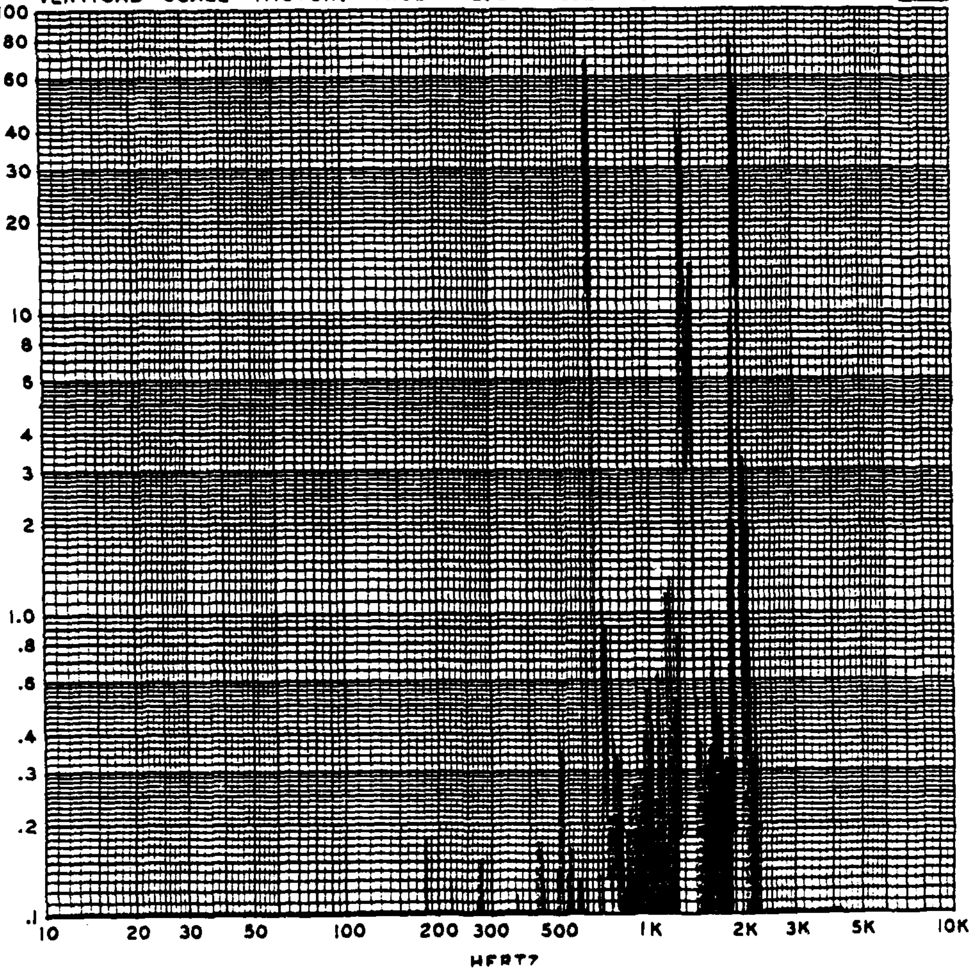


RUN IDENTIFICATION APU HOTFIRE CHANNEL 11 STS-2

PARAMETER D0280A
OVERALL RMS VALUE 2.7269 G's
ENGINEERING UNITS 30.0000 G's
START TIME: (HR: MIN: SEC) 258/12:03:20
SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81
ANALYSIS B W 2.2467 Hz
ANALOG L.P. FILTER B W 2300.0000 Hz
DEGREES OF FREEDOM 9.0000
PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -3

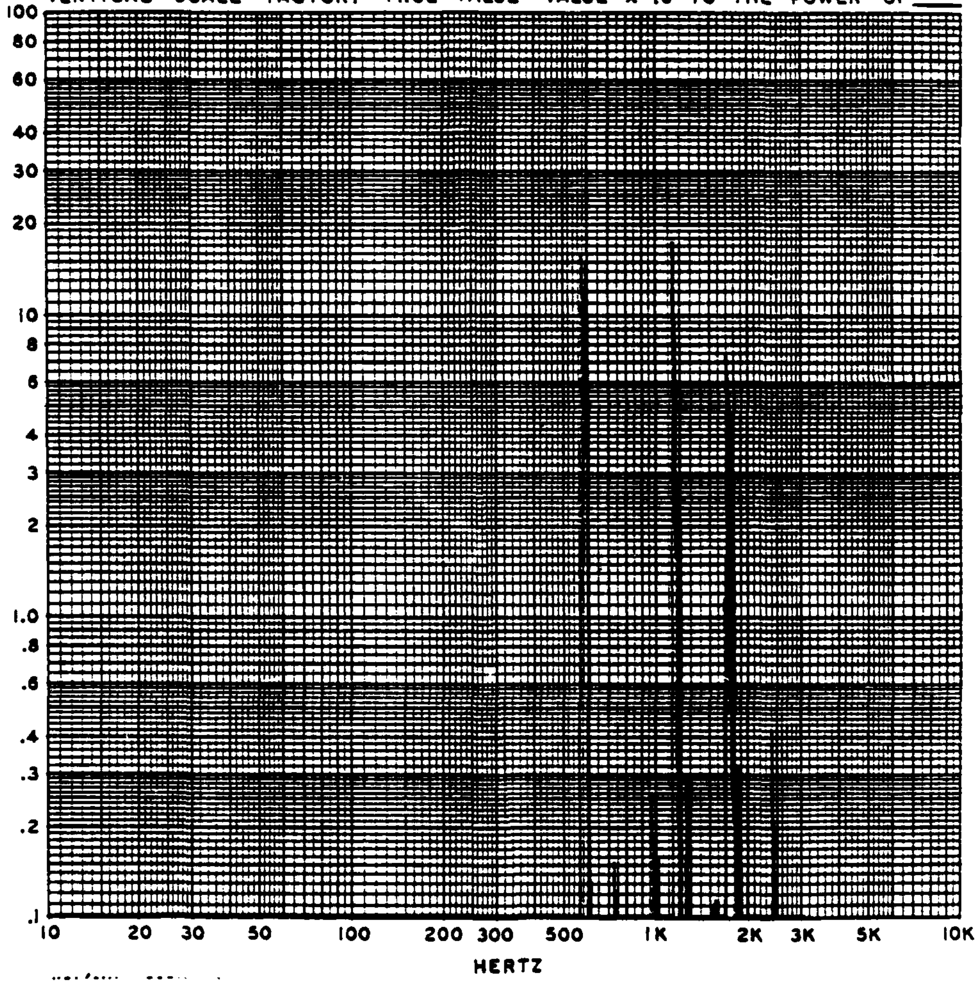


RUN IDENTIFICATION APU HOTFIRE CHANNEL 11 STS-2

PARAMETER D0280A
OVERALL RMS VALUE 2.9619 G's
ENGINEERING UNITS 30.0000 G's
START TIME: (HR: MIN: SEC) 258/12:01:22
SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81
ANALYSIS B W 2.2467 Hz
ANALOG L.P. FILTER B W 2300.0000 Hz
DEGREES OF FREEDOM 9.0000
PROCESS DATE 18 September 81

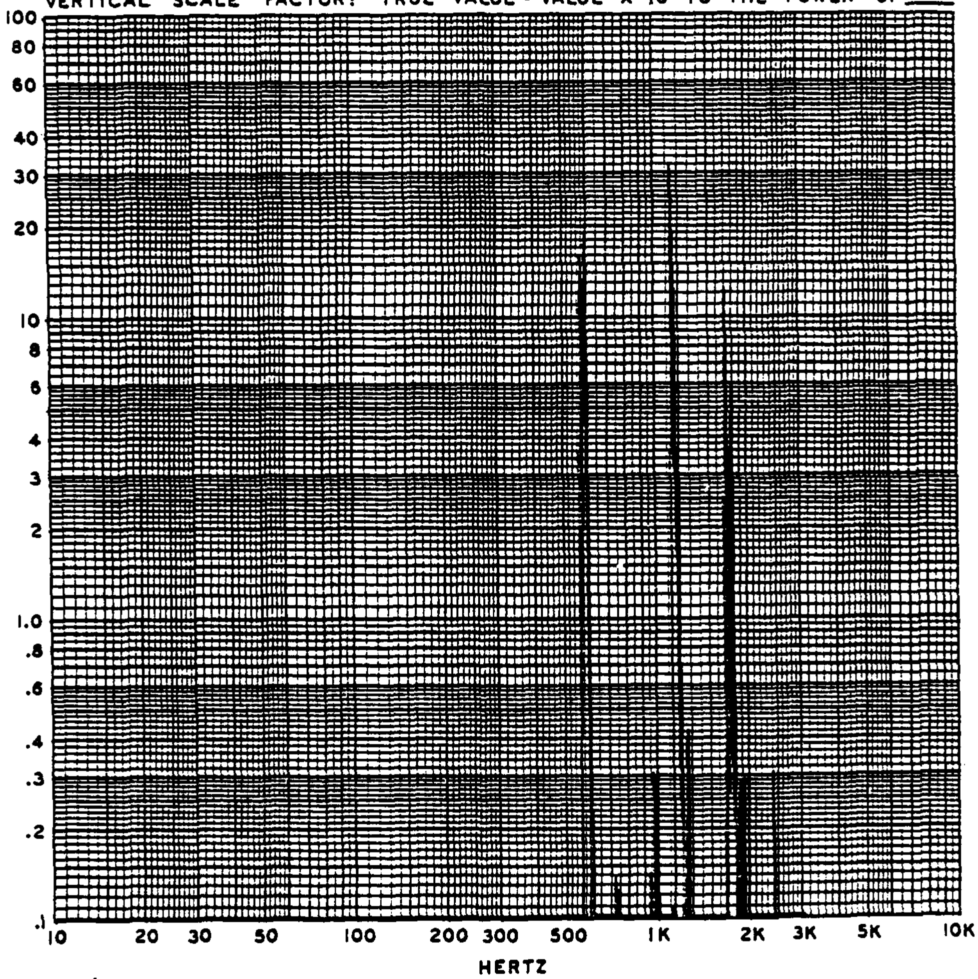
VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -2



RUN IDENTIFICATION APU HOTFIRE CHANNEL 11 STS-2

PARAMETER	D0280A	TEST DATE	15 September 81
OVERALL RMS VALUE	3.6335 G's	ANALYSIS B W	2.2467 Hz
ENGINEERING UNITS	30.0000 G's	ANALOG L.P. FILTER B W	2300.0000 Hz
START TIME: (HR: MIN: SEC)	258/11:58:14	DEGREES OF FREEDOM	9.0000
SAMPLE RATE (S/SEC)	6134.9687	PROCESS DATE	18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -2



RUN IDENTIFICATION APU HOTFIRE CHANNEL 11 STS-2

PARAMETER D0280A

OVERALL RMS VALUE 3.1781 G's

ENGINEERING UNITS 30.0000 G's

START TIME: (HR: MIN: SEC) 258/11:57:46

SAMPLE RATE (S/SEC) 6134.9687

TEST DATE 15 September 81

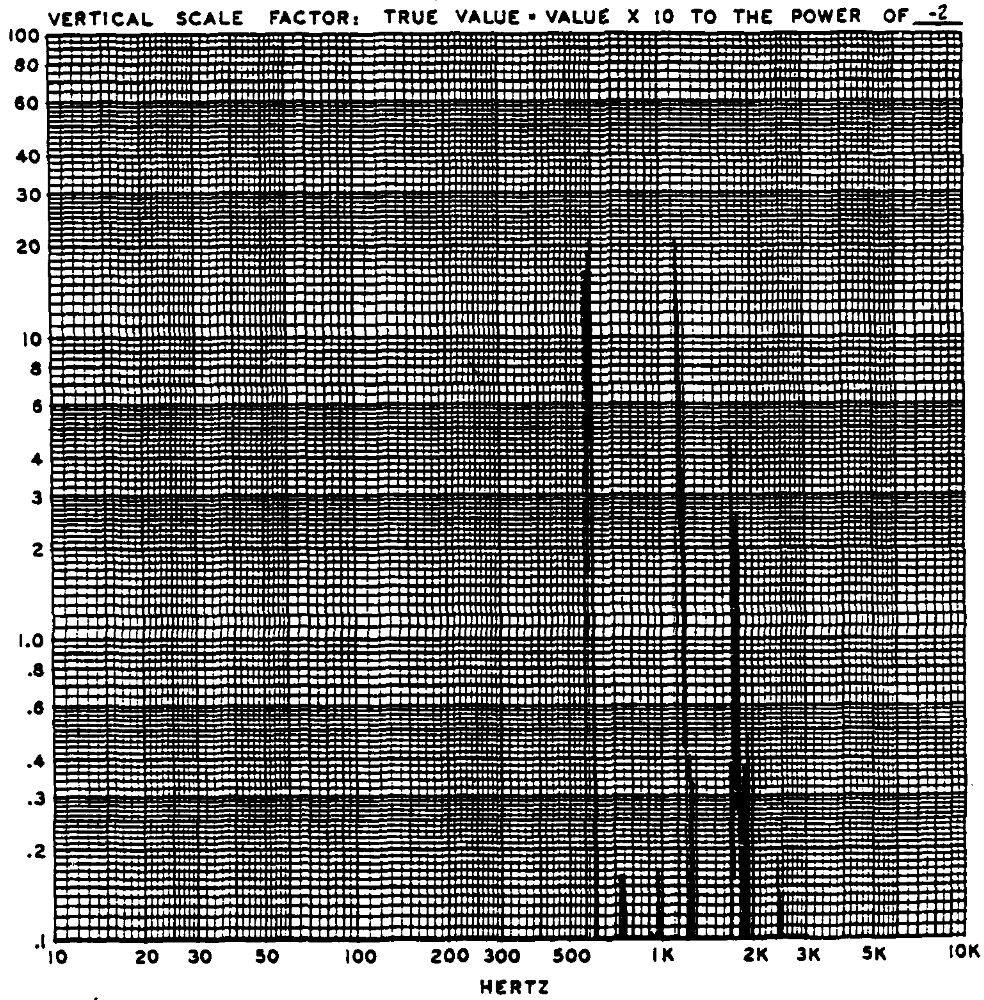
ANALYSIS B W 2.2467 Hz

ANALOG L.P. FILTER B W 2300.0000 Hz

DEGREES OF FREEDOM 9.0000

PROCESS DATE 18 September 81

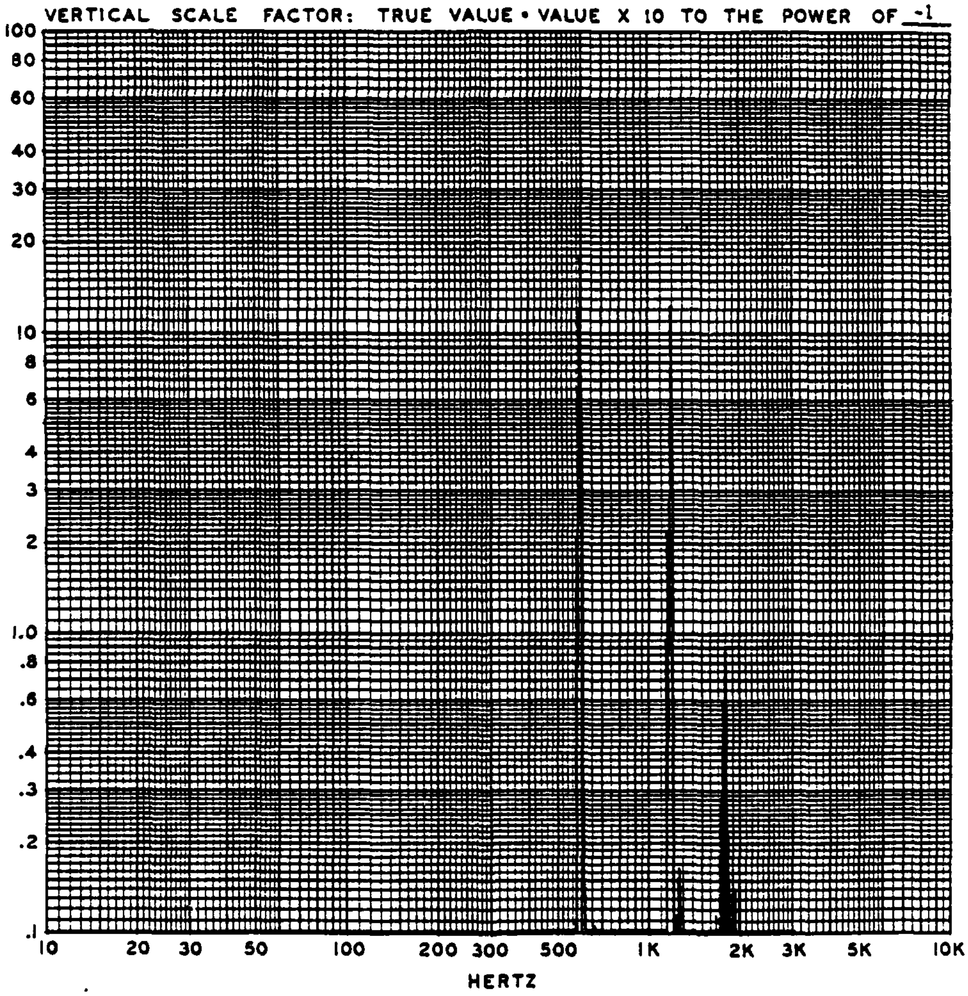
VERTICAL SCALE FACTOR: TRUE VALUE = VALUE X 10 TO THE POWER OF -2



RUN IDENTIFICATION APU HOTFIRE CHANNEL 11 STS-2

PARAMETER D0280A	TEST DATE 15 September 81
OVERALL RMS VALUE 6.4863 G's	ANALYSIS B W 2.2467 Hz
ENGINEERING UNITS 30.0000 G's	ANALOG L.P. FILTER B W 2300.0000 Hz
START TIME: (HR: MIN: SEC) 258/11:56:35	DEGREES OF FREEDOM 9.0000
SAMPLE RATE (S/SEC) 6134.9687	PROCESS DATE 18 September 81

VERTICAL SCALE FACTOR: TRUE VALUE • VALUE X 10 TO THE POWER OF -1



APPENDIX A

QUICK LOOK DATA PACKET

TEST TITLE APU HOTFIRE STS-2

TEST DATE 15 SEPTEMBER 81

RUN NO. (S)

REDUCTION DATE 18 SEPTEMBER 81

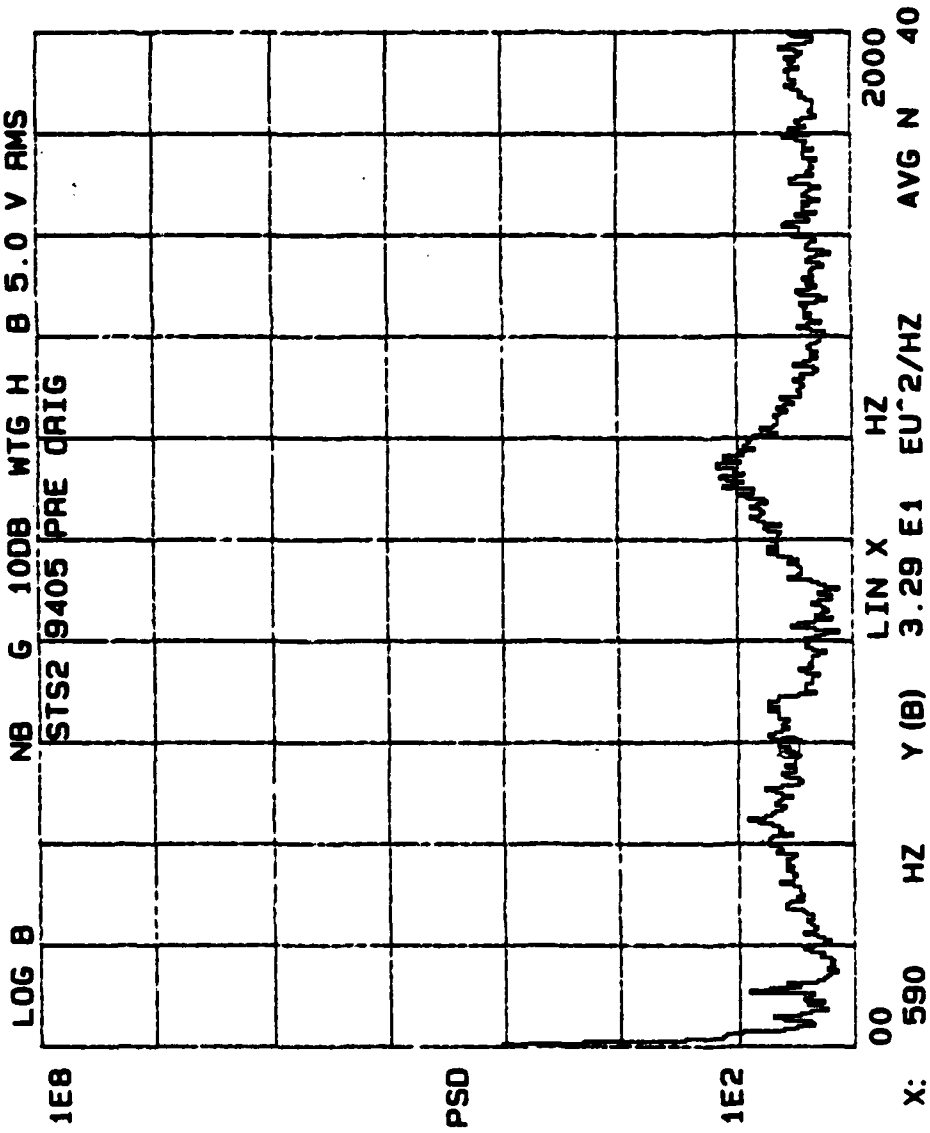
FACILITY JSC NASA VATF BLDG #49

SEQUENCE NO. (S)

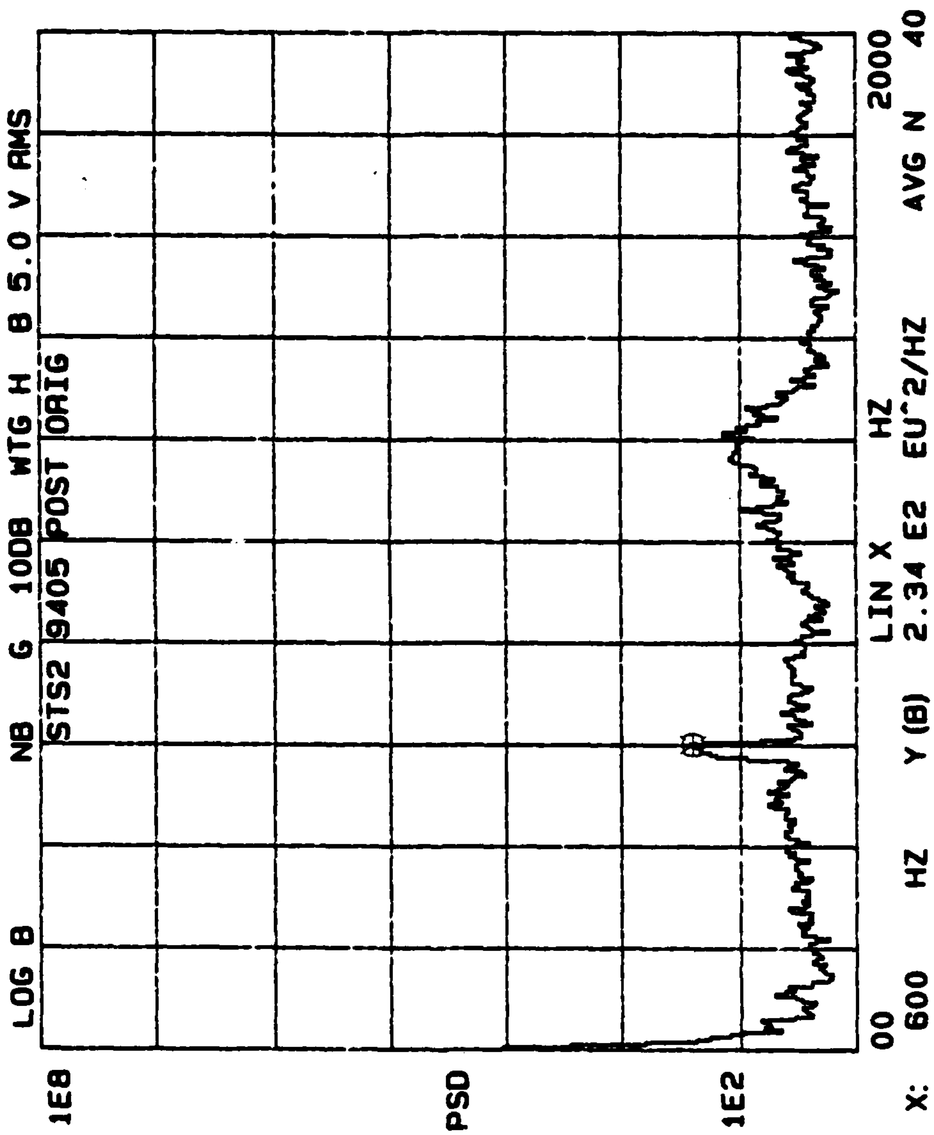
APU 2 x-axis APU 2 Z

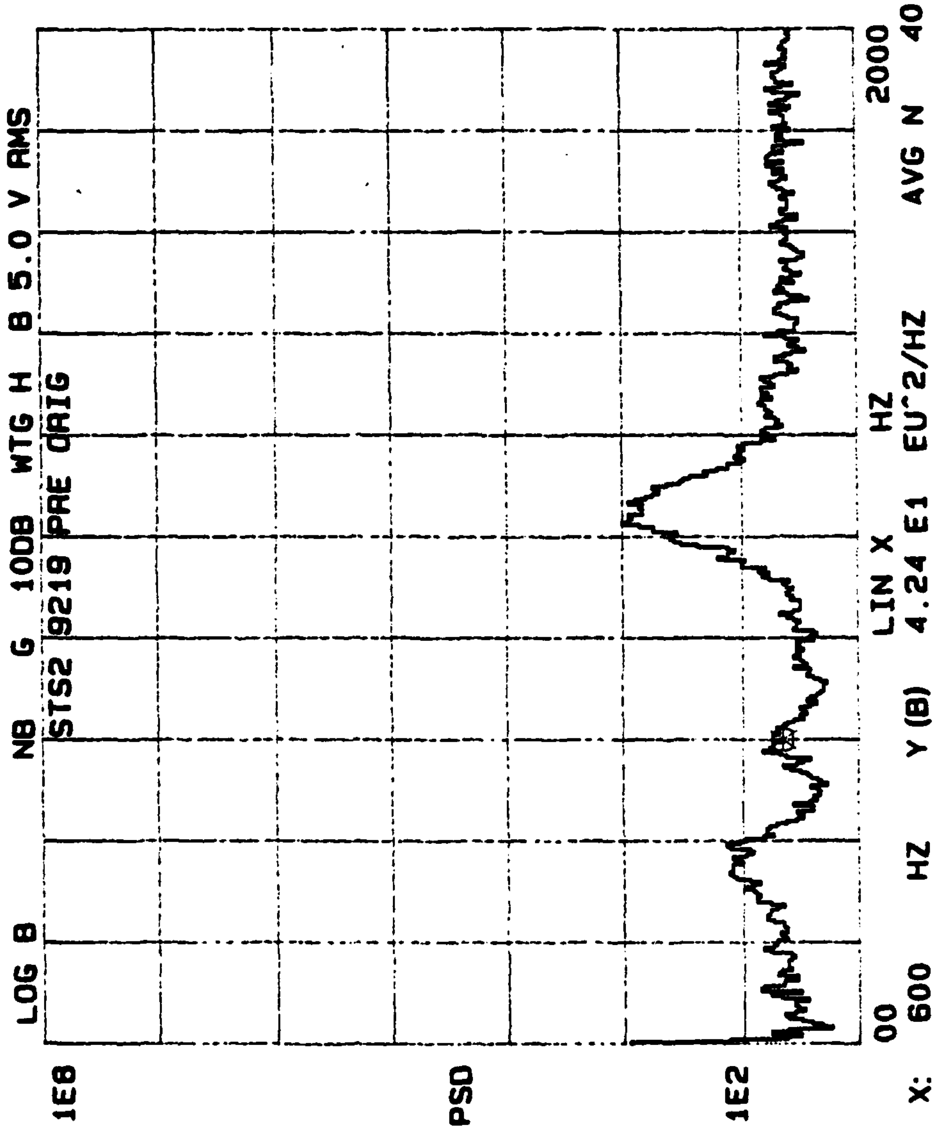
DATA TYPE PSD PLOTS D0280A and D0281A + Noise Floor

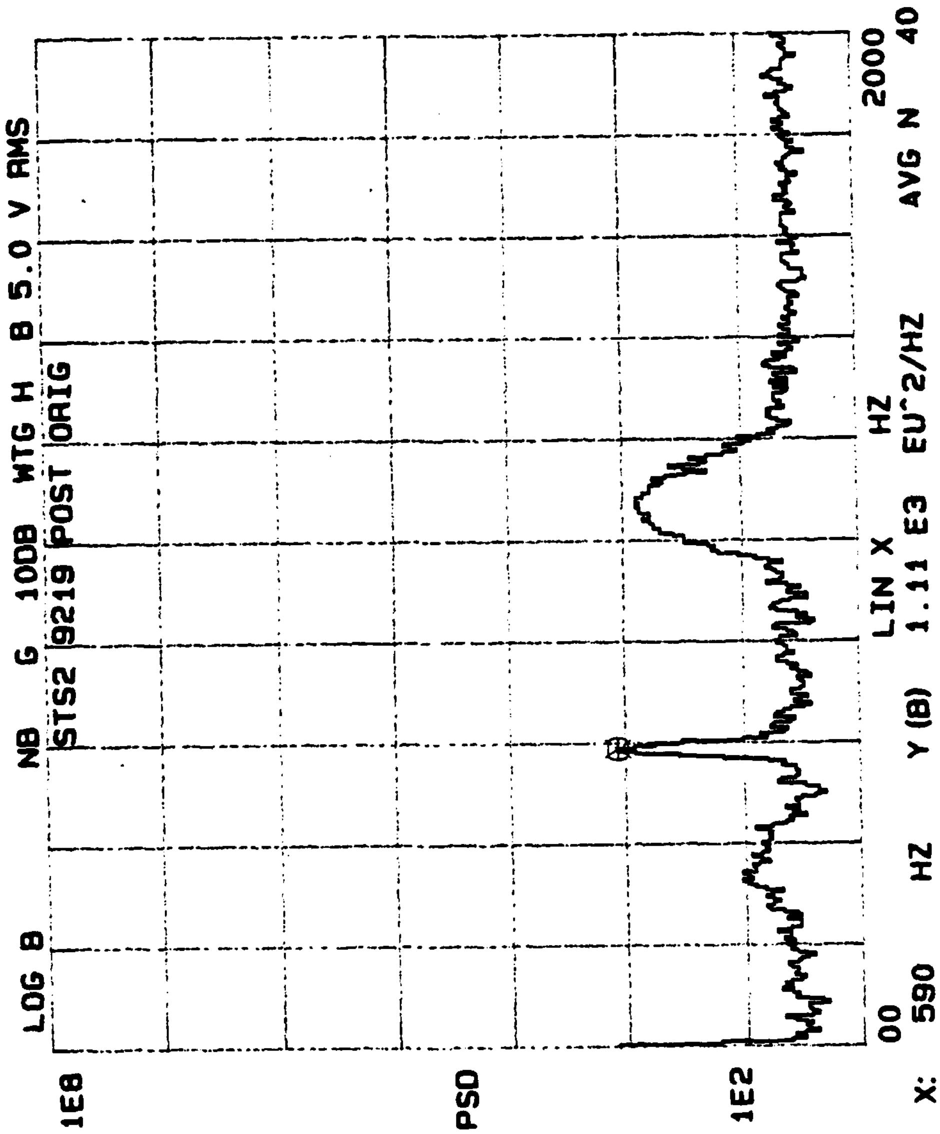
NSI-SML FORM 147 (MAY. 79) NASA-JSC

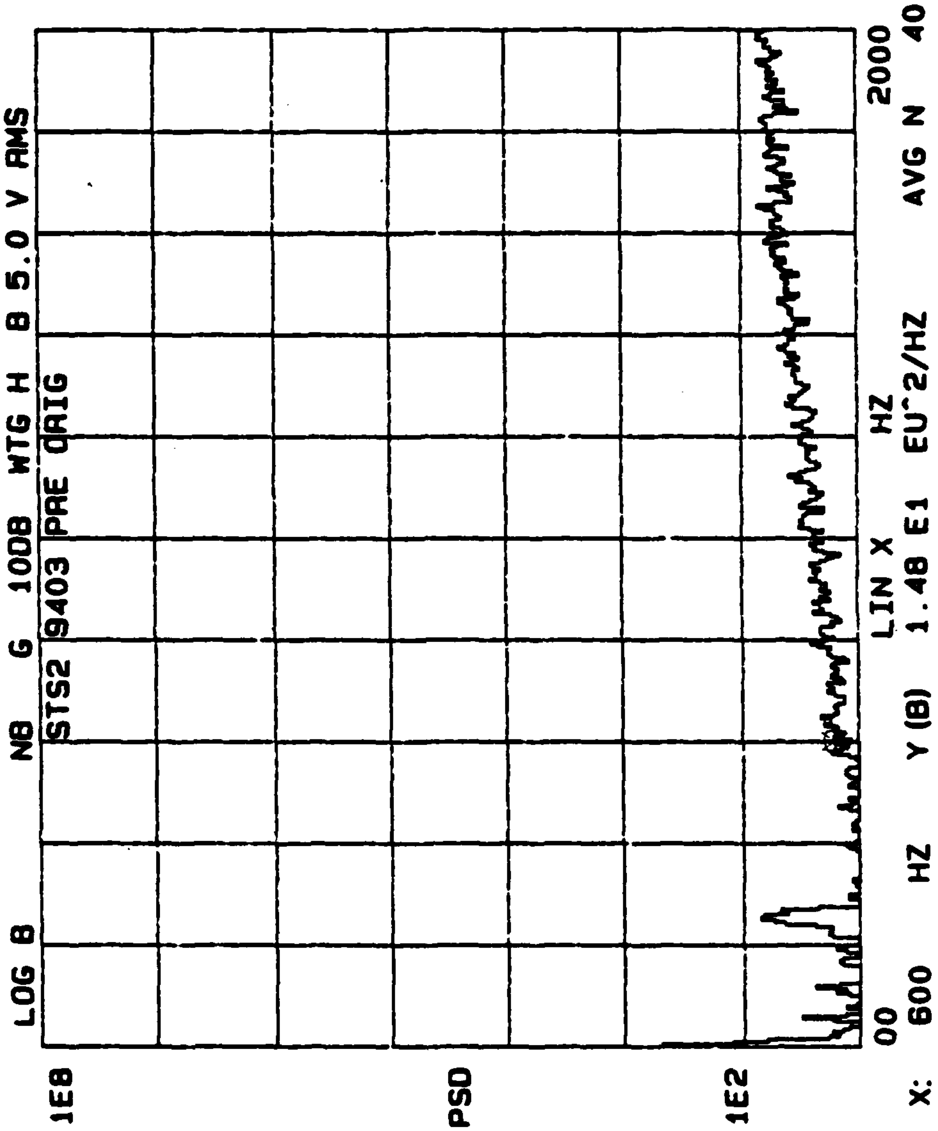


LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS2 9405 PRE ORIG
1E8
PSD
1E2
X: 00 590 HZ
Y (B)
LIN X 3.29 E1
HZ EU^2/HZ
AVG N 2000 40

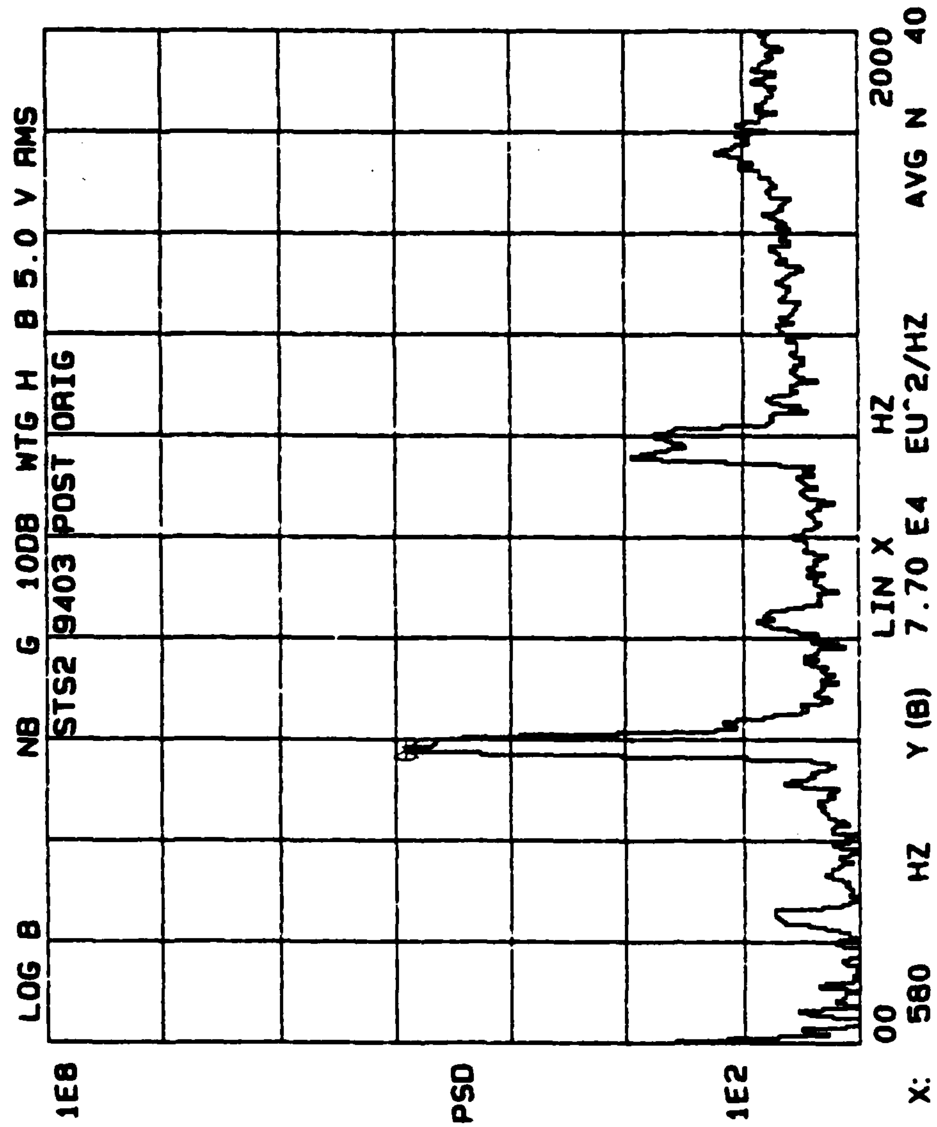

LOG B NB G 10DB WTG H B 5.0 V RMS
STS2 9405 POST ORIG
1E8
PSD
1E2
X: 00 600 HZ
Y (B) LIN X 2.34 E2
HZ EU^2/HZ
2000
AVG N 40

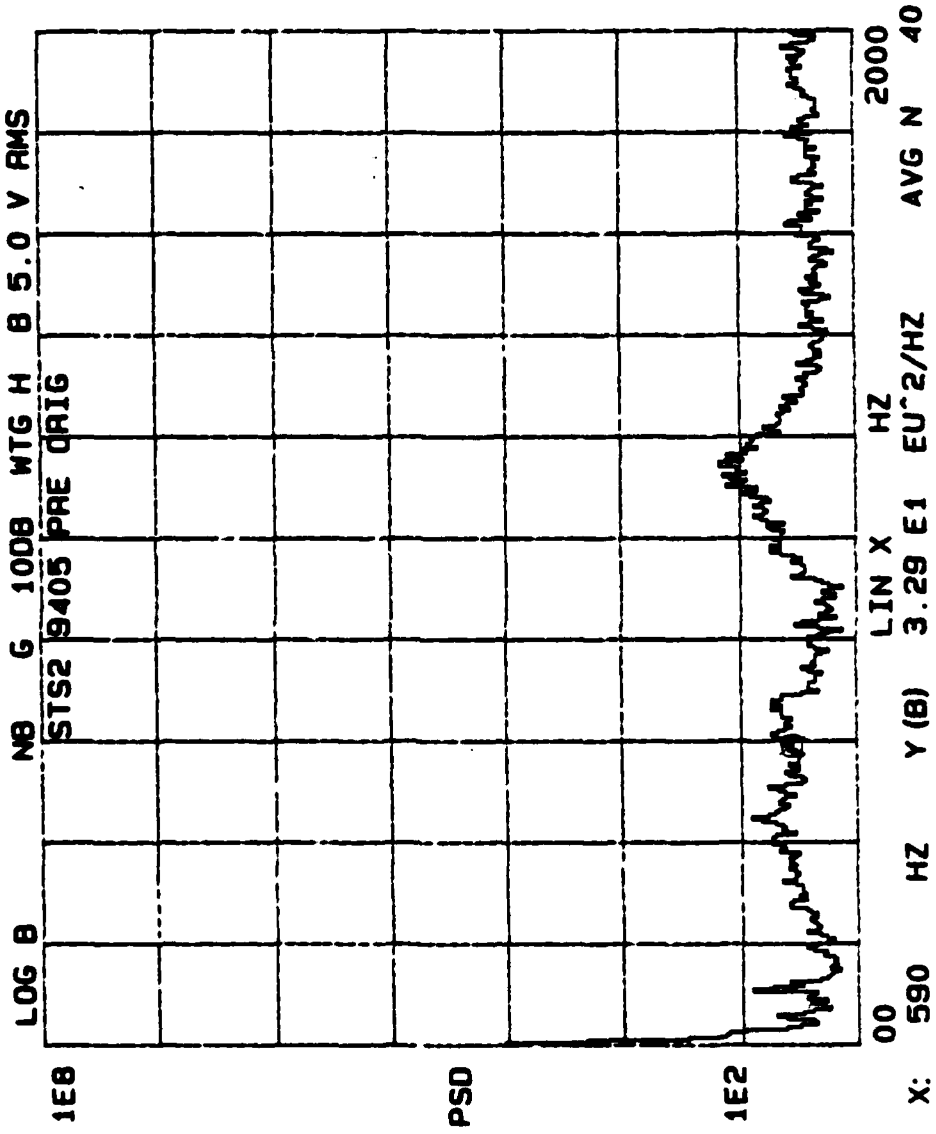

LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS2 9219 PRE ORIG
1E8
PSD
1E2
X:
00
600
HZ
Y (B)
LIN X
4.24 E1
HZ
EU^2/HZ
AVG N
2000
40

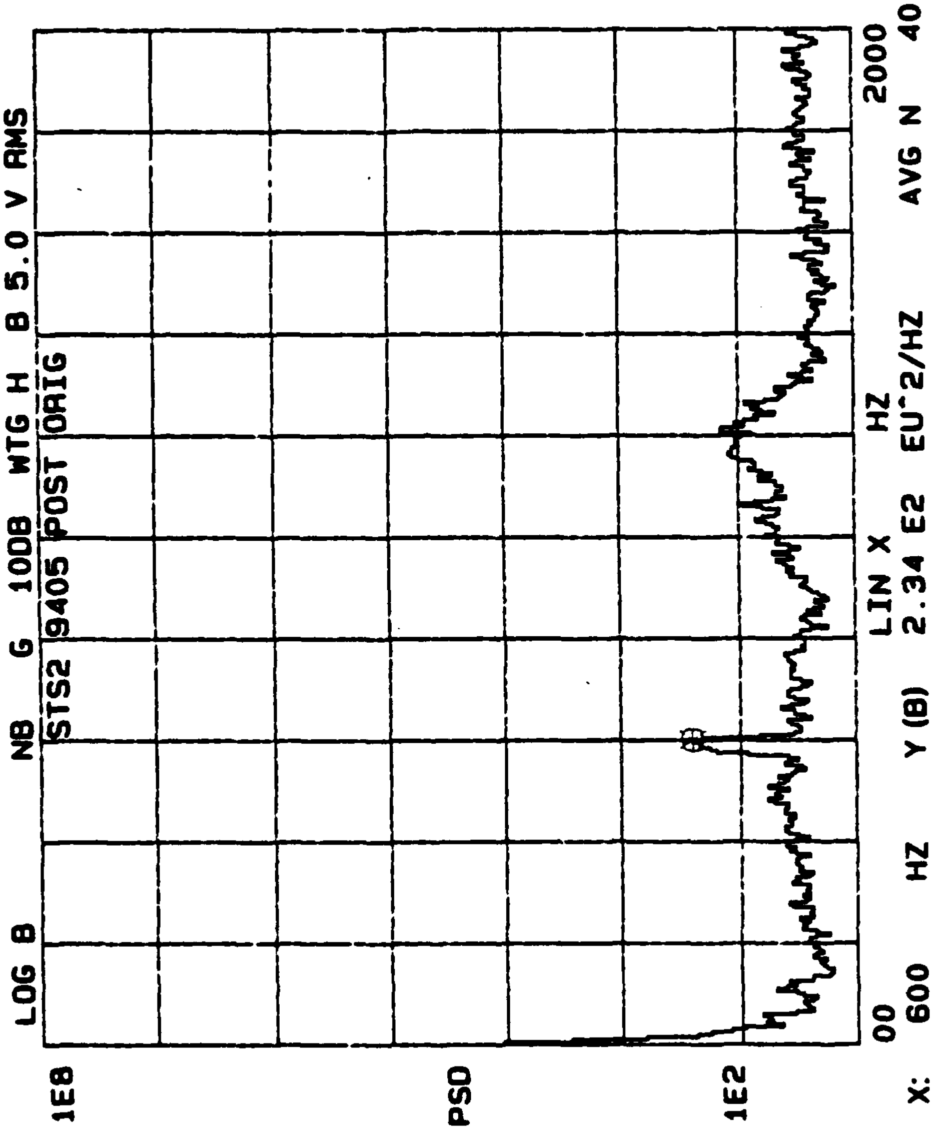

LOG B NB G 10DB WTG H B 5.0 V RMS
STS2 9219 POST ORIG
1E8
PSD
1E2
X: 00 590 HZ Y (B) LIN X 1.11 E3 HZ 2000
EU^2/HZ
AVG N 40



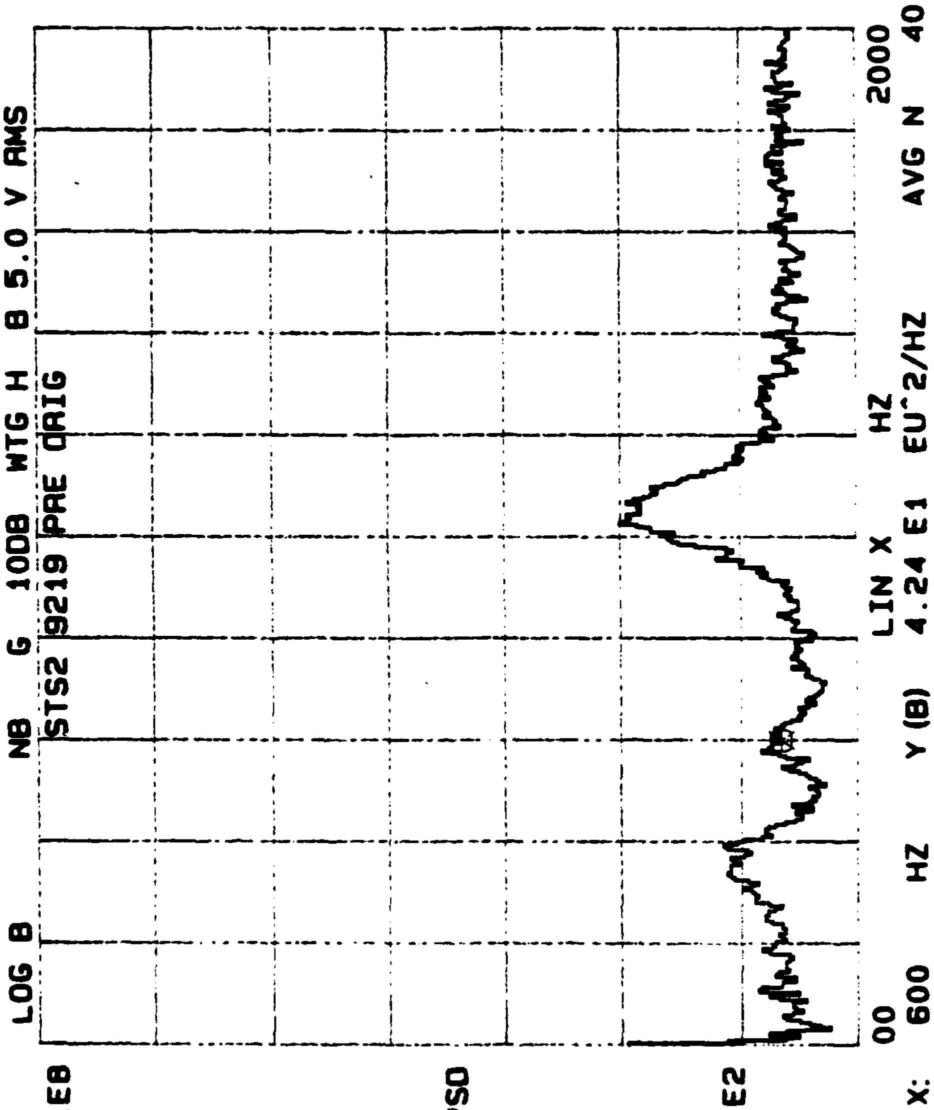
LOG B
NB G 10DB WTG H B 5.0 V RMS
STS2 9403 PRE ORIG
1E8
PSD
1E2
00
2000
X: 600 HZ Y (B) LIN X 1.48 E1 HZ AVG N 40
EU^2/HZ

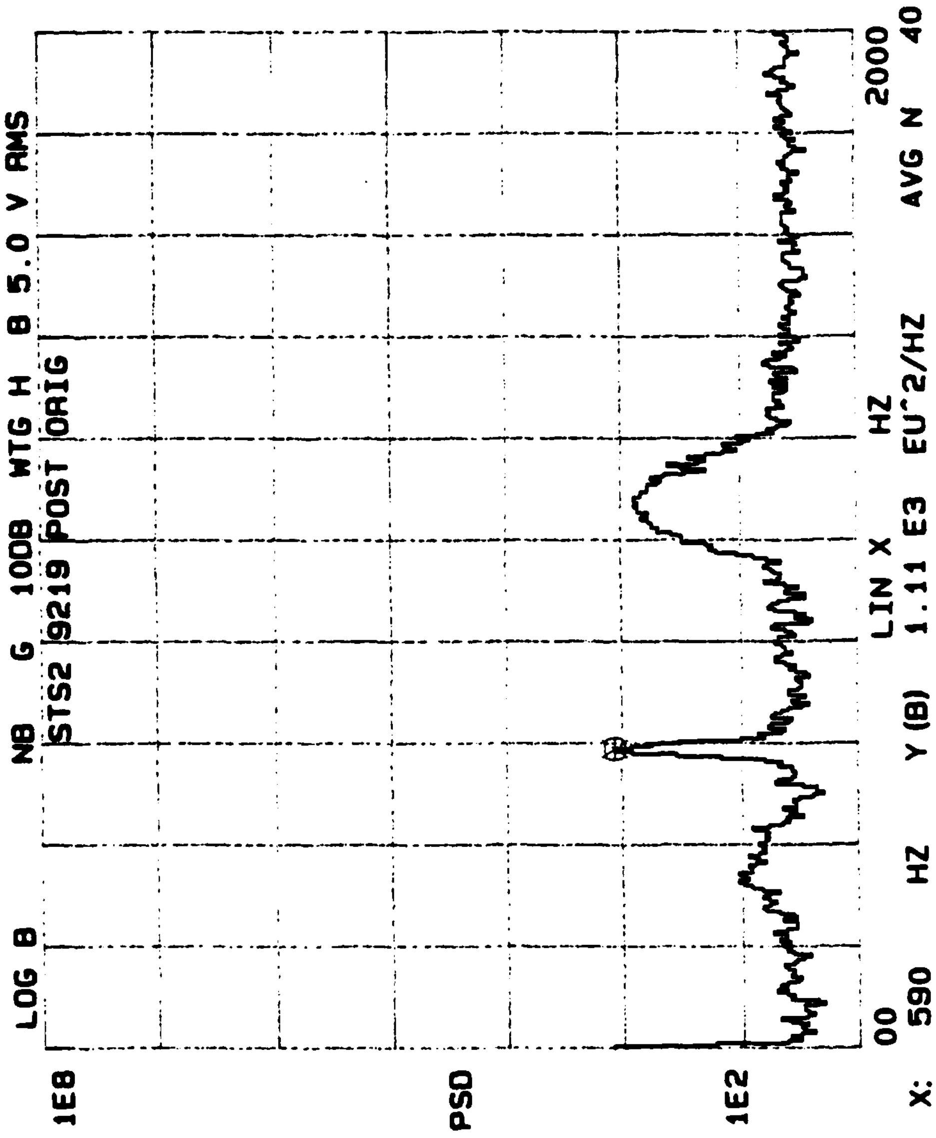

LOG B
HZ
NB
G
10DB
WTG H
B 5.0 V RMS
STS2
9403
POST
ORIG
1E8
PSD
1E2
X: 00 580
HZ
Y (B)
LIN X
7.70 E4
HZ
EU^2/HZ
2000
AVG N 40

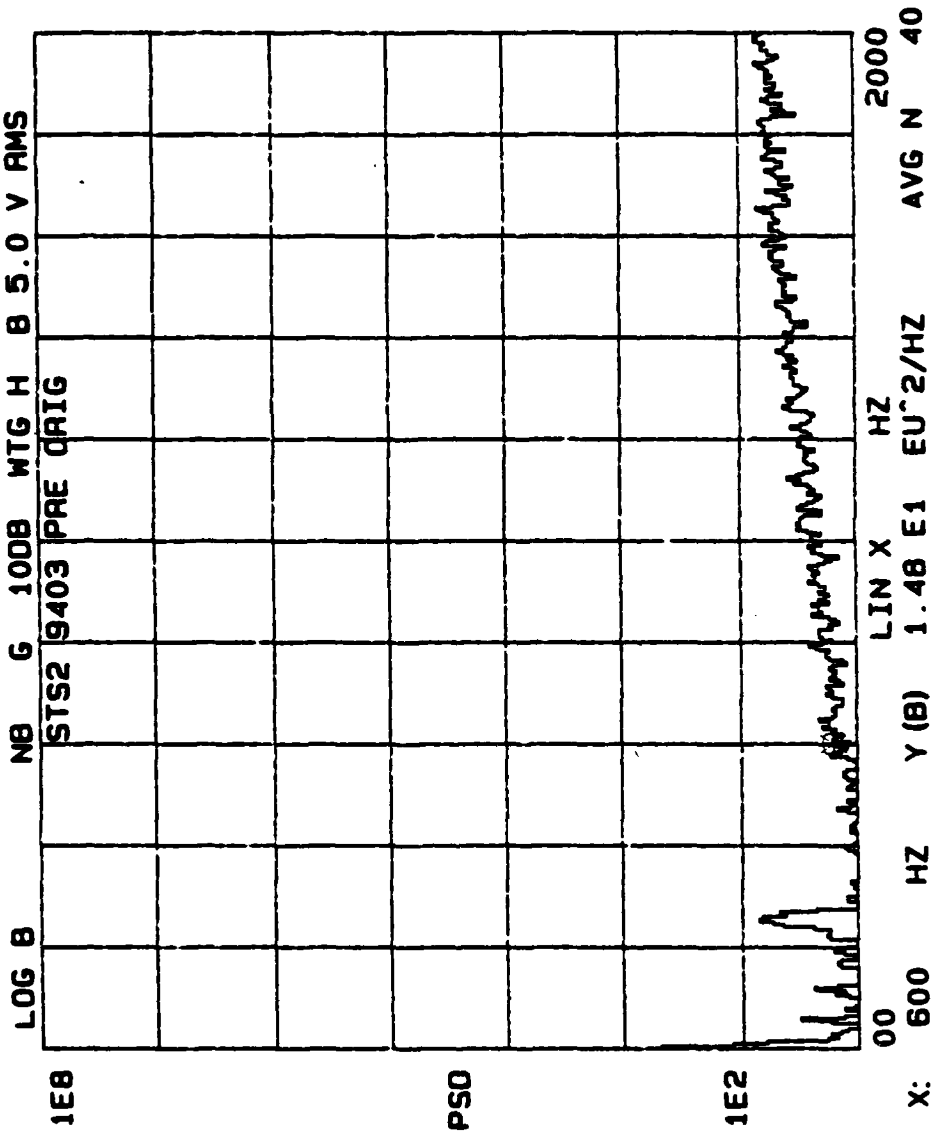

LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS2 9405 PRE ORIG
1E8
PSD
1E2
X:
00
590
HZ
Y (B)
LIN X
3.29 E1
HZ
EU^2/HZ
AVG N
2000
40



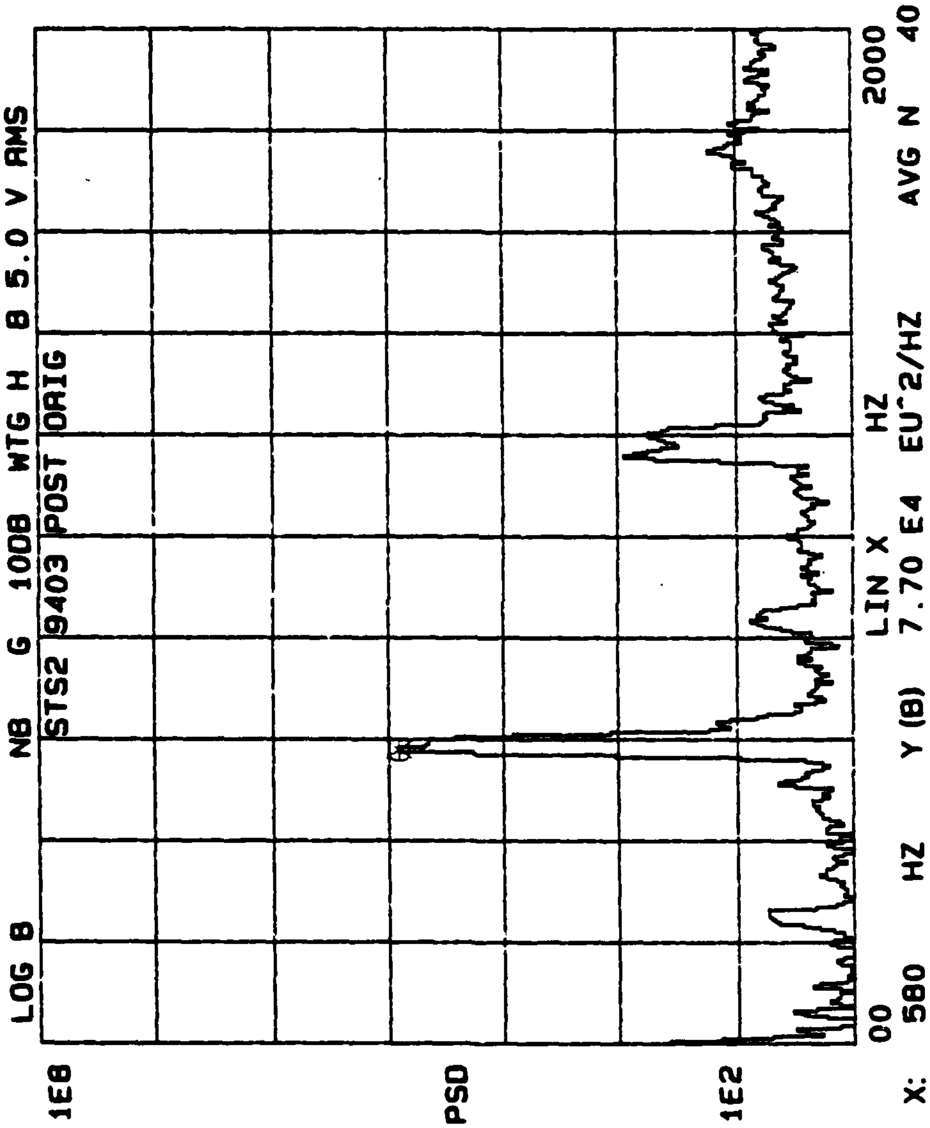
LOG B HZ NB G 10DB WTG H B 5.0 V RMS
STS2 9405 POST ORIG
1E8
PSD
1E2
X: 00 600 HZ Y (B) LIN X 2.34 E2 HZ EU^2/HZ 2000 AVG N 40

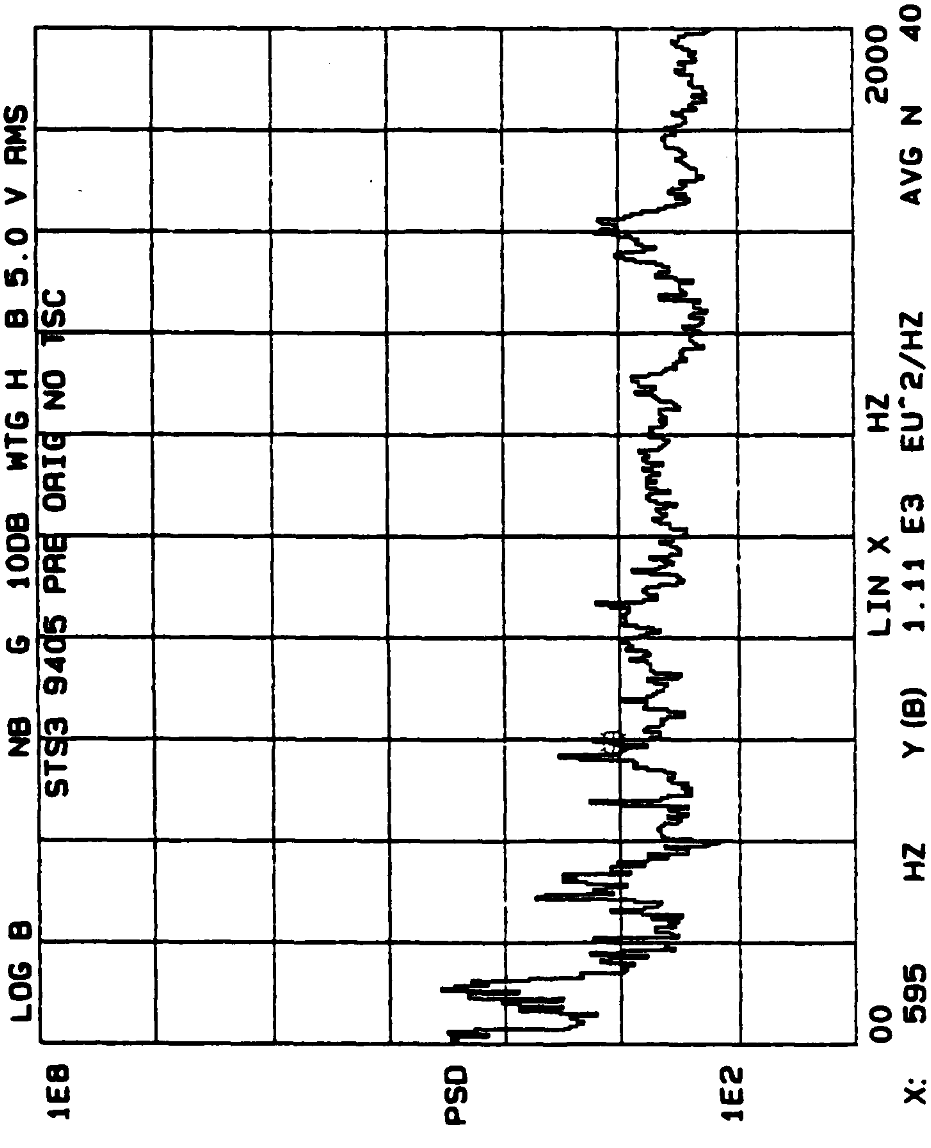

LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS2 9219 PRE ORIG
1E8
PSD
1E2
X:
00
600
HZ
Y (B)
LIN X
4.24 E1
HZ
EU^2/HZ
AVG N
2000
40

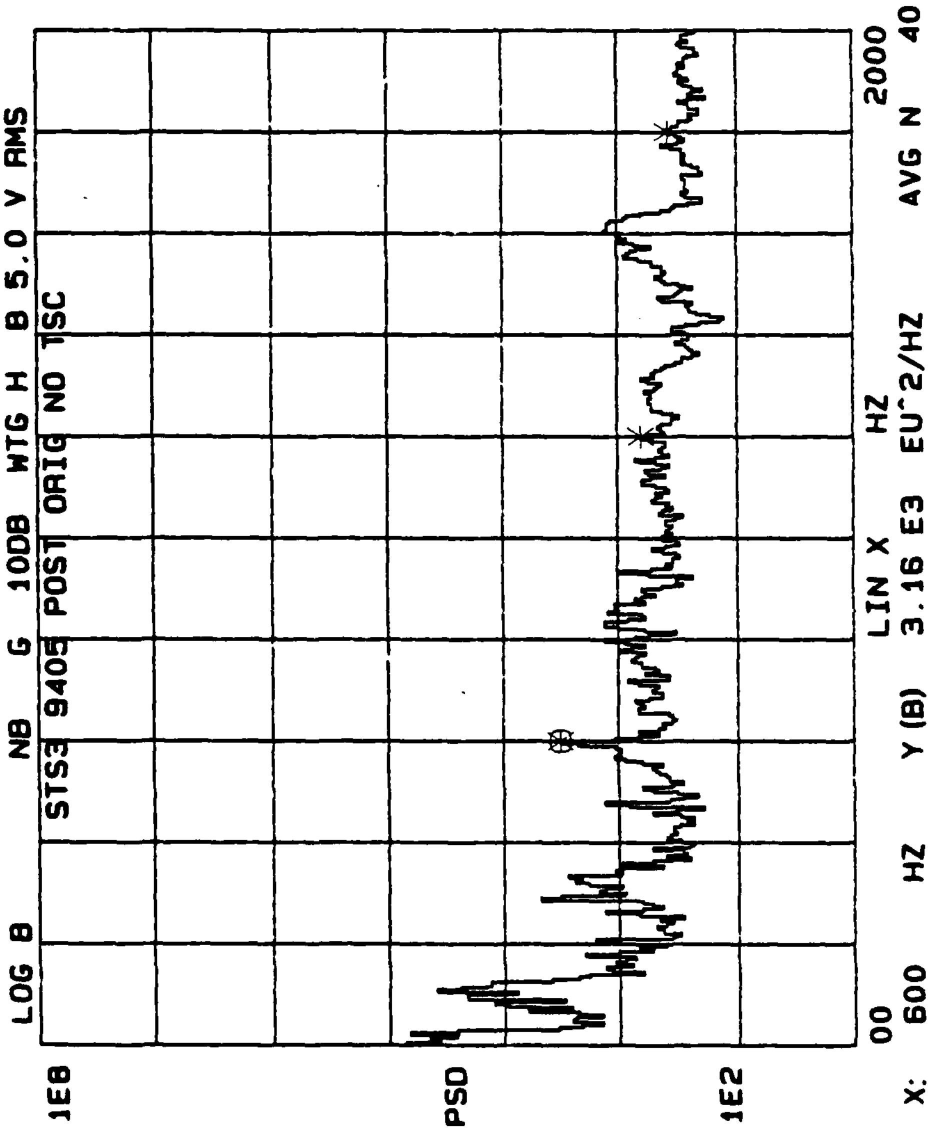

LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS2
9219
POST
ORIG
1E8
PSD
1E2
X:
00
590
HZ
Y (B)
LIN X
1.11 E3
HZ
EU^2/HZ
2000
AVG N
40



LOG B NB G 10DB WTG H B 5.0 V RMS
STS2 9403 PRE ORIG
1E8
PSD
1E2
X: 00 600 HZ Y (B) LIN X HZ 2000
1.48 E1 EU^2/HZ AVG N 40


LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS2
9403
POST
ORIG
1E8
PSD
1E2
X: 580
00
HZ
Y (B)
LIN X
7.70 E4
HZ
EU^2/HZ
AVG N
2000
40


LOG B NB G 10DB WTG H B 5.0 V RMS
STS3 9405 PRE ORIG NO TSC
1E8
PSD
1E2
X: 00 595 HZ Y (B) LIN X 1.11 E3 HZ 2000
EU^2/HZ AVG N 40


LOG B NB G 10DB WTG H B 5.0 V RMS
STS3 9405 POST ORIG NO TSC
1E8
PSD
1E2
X: 00 600 HZ Y (B) LIN X 3.16 E3 HZ 2000
EU^2/HZ AVG N 40

FILE: ABPDBP FORTRAN A1

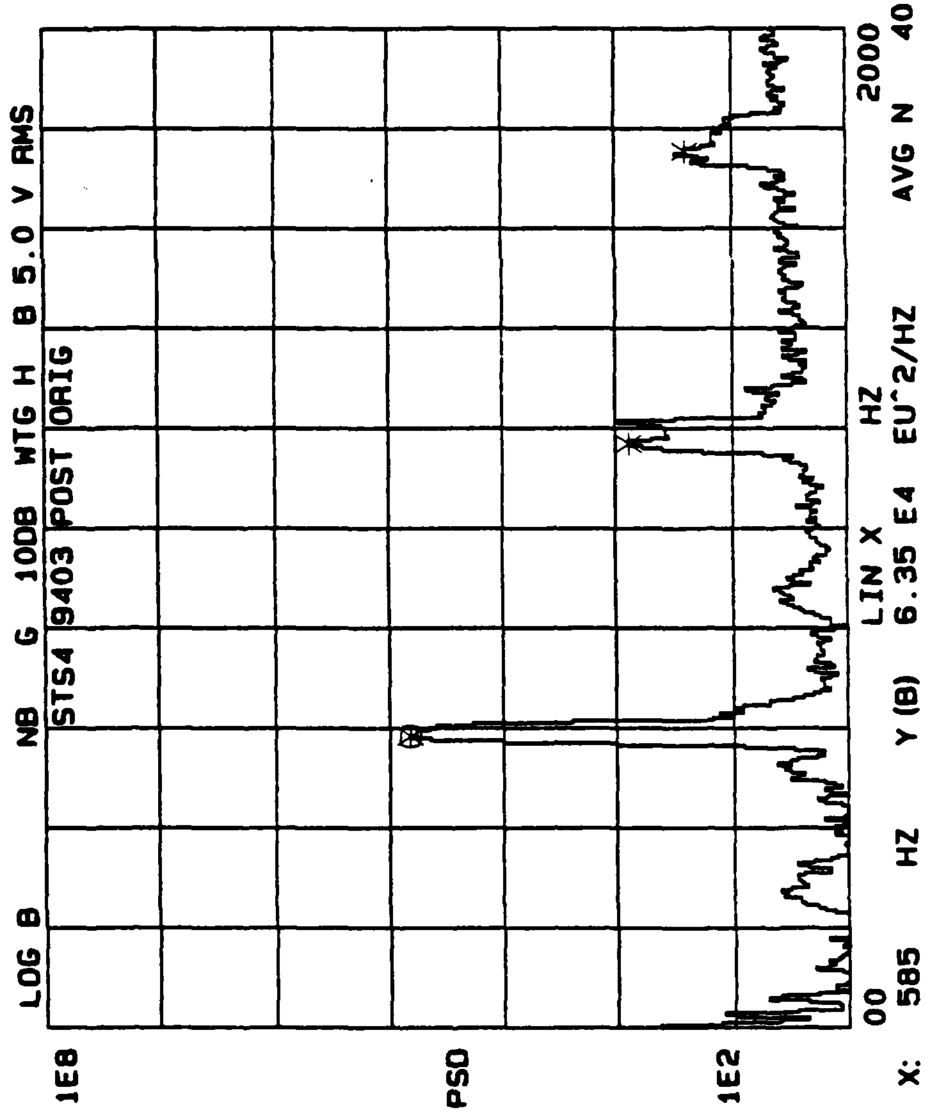
```
      ALFA1=A1
      ALFA2=1./A1
      BETA1=D/E
      BETA2=1./(D*E)
      GAMMA1=D**2.
      GAMMA2=1./D**2.
      NUM1(1)=ALFA1*(WODIG**2.)
      NUM1(2)=0.
      NUM1(3)=1.
      DEN1(1)=GAMMA1*(WODIG**2.)
      DEN1(2)=BETA1*WODIG
      DEN1(3)=1.
      NUM2(1)=ALFA2*(WODIG**2.)
      NUM2(2)=0.
      NUM2(3)=1.
      DEN2(1)=GAMMA2*(WODIG**2.)
      DEN2(2)=BETA2*WODIG
      DEN2(3)=1.
      RHO=RHO1*RHO2
      INUM1=3
      INUM2=3
      IDEN1=3
      IDEN2=3
      CALL PMPY(ANUM,IANUM,NUM1,INUM1,NUM2,INUM2)
      CALL PMPY(ADEN,IADEN,DEN1,IDEN1,DEN2,IDEN2)
      WRITE(4,110)RHO
      WRITE(4,120)ANUM(5),ANUM(4),ANUM(3),ANUM(2),ANUM(1)
      DO 105 I=1,5
         ANUM(I)=ANUM(I)*RHO
  105 CONTINUE
      WRITE(4,130)ANUM(5),ANUM(4),ANUM(3),ANUM(2),ANUM(1)
      DO 106 I=1,5
         ANUM(I)=ANUM(I)/RHO
  106 CONTINUE
      WRITE(4,140)ADEN(5),ADEN(4),ADEN(3),ADEN(2),ADEN(1)
  110 FORMAT('1','SECTION 2 OUTPUT',//,
     &' ANALOG ELLIPTIC BANDPASS FILTER TRANSFER FUNCTION'
     &,//,' NUMERATOR COEFFICIENT = RHO = ',F9.6,/)
  120 FORMAT(' NUMERATOR POLYNOMIAL (NORMALIZED)'
     &,/,E12.5,' S**4 +',/,E12.5,' S**3 +'
     &,/,E12.5,' S**2 +',/,E12.5,' S    +',/,E12.5,//)
  130 FORMAT(' NUMERATOR POLYNOMIAL (UN-NORMALIZED)'
     &,/,E12.5,' S**4 +',/,E12.5,' S**3 +'
     &,/,E12.5,' S**2 +',/,E12.5,' S    +',/,E12.5,//)
  140 FORMAT(' DENOMINATOR POLYNOMIAL (NORMALIZED)'
     &,/,E12.5,' S**4 +',/,E12.5,
     &' S**3 +',/,E12.5,' S**2 +',/,E12.5,' S    +',/,E12.5,///)
C***********************************************************************
C                                                                      *
C                             SECTION 2A                               *
C                                                                      *
C    CALCULATE THE COMPLEX POLES AND ZEROS OF THE ANALOG FILTER        *
C                                                                      *
C***********************************************************************
      IPDEG = 4
```

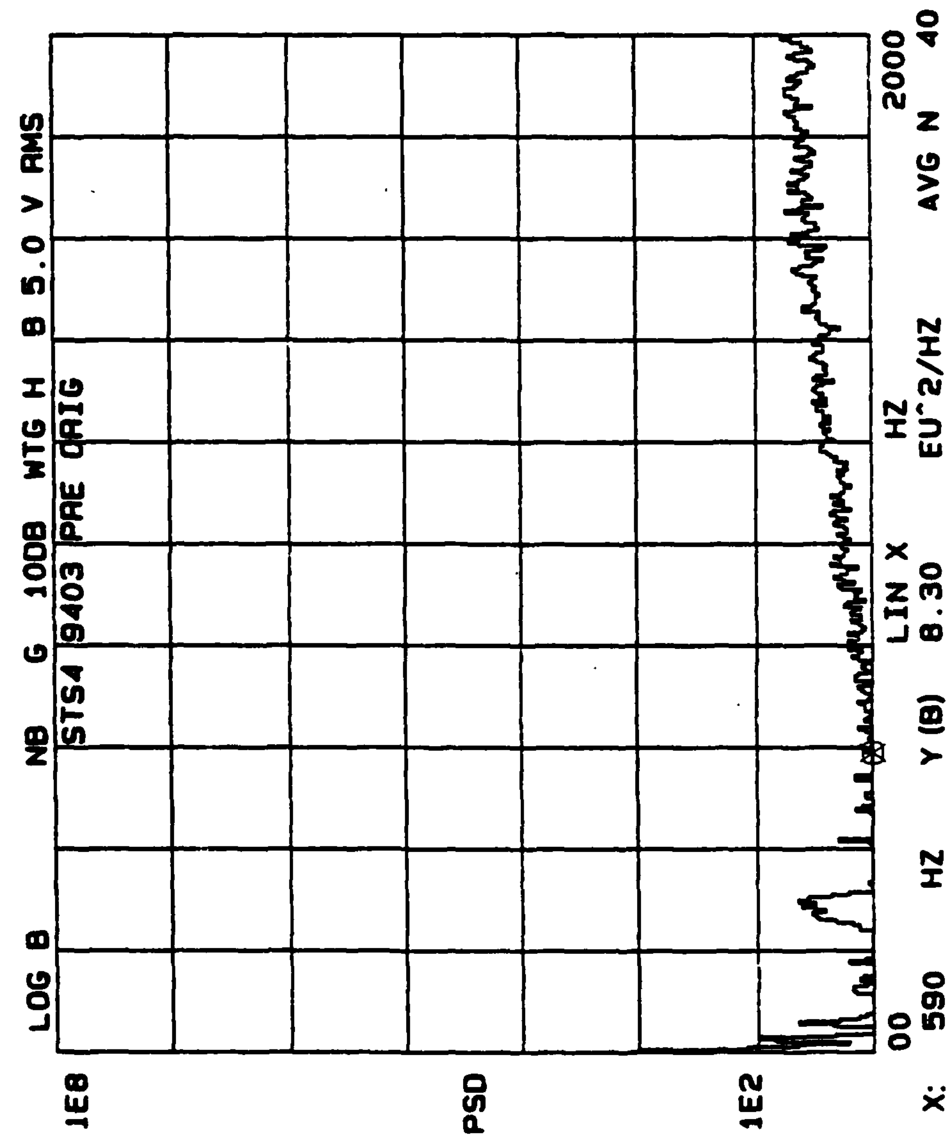
```
C     CHOOSE R17=R27=1./(WO*C1)(APPROX)=26.5 KOHMS
C     USE MEASURED VALUES FOR CALCULATIONS
C
C**********************************************************************
      C11=.00996E-06
      C12=.00995E-06
      C21=.01030E-06
      C22=.01034E-06
      R17=26.7E+03
      R27=26.7E+03
C**********************************************************************
C     ELLIPTIC BANDPASS FILTER STAGE RESISTOR VALUES
C**********************************************************************
      R11=(E/(K*D*WO*C11))*SQRT(A/C)
      R12=K*R11*SQRT(C/A)
      R13=1./(D*WO*C11)
      R14=(R17/K)*SQRT(A/C)
      R15=(D/(K*A1*WO*C12))*SQRT(A/C)
      R16=C11*R13/C12
      WRITE(4,20)C11,C12,R17,R11,R12,R13,R14,R15,R16
      R21=((D*E)/(K*WO*C21))*SQRT(A/C)
      R22=(K*R21)*SQRT(C/A)
      R23=D/(WO*C21)
      R24=(R27/K)*SQRT(A/C)
      R25=(A1/(K*D*WO*C22))*SQRT(A/C)
      R26=C21*R23/C22
      WRITE(4,30)C21,C22,R27,R21,R22,R23,R24,R25,R26
   10 FORMAT(' SECTION 1 OUTPUT',//,
     &' INPUT AND DERIVED PARAMETERS FOR FURTHER CALCULATIONS'
     &,//,' A = ',F9.6,/,' B = ',F9.6,/,' C = ',F9.6,/,' D = ',
     &F9.6,/,' E = ',F9.6,/,' A1 = ',F9.6,/,' FO = ',F9.3,/,' WO = '
     &,F9.3,/,' Q = ',F9.6,/,' K = ',F9.6,/,' K1 = ',F9.6,/,
     &' K2 ',F9.6,/,' WO(DIG) = ',F9.3,/,' FO(DIG) = ',F9.3,///)
   20 FORMAT(' ELLIPTIC ANALOG BPF COMPONENT VALUES',//,' FIRST STAGE'
     &,//,' C11 = ',E8.3,/,' C12 = ',E8.3,/,' R17 = ',E8.3,
     &/,' R11 = ',E8.3,/,' R12 = ',E8.3,/,' R13 = ',E8.3,/,
     &' R14 = ',E8.3,/,' R15 = ',E8.3,/,' R16 = ',E8.3,//)
   30 FORMAT(' SECOND STAGE',//,
     &' C21 = ',E8.3,/,' C22 = ',E8.3,/,' R27 = ',E8.3,
     &/,' R21 = ',E8.3,/,' R22 = ',E8.3,/,' R23 = ',E8.3,/,
     &' R24 = ',E8.3,/,' R25 = ',E8.3,/,' R26 = ',E8.3)
C**********************************************************************
C
C                              SECTION 2
C
C     IN THIS PORTION OF THE PROGRAM WE COMPUTE THE ANALOG TRANSFER
C     FUNCTION OF THE ELLIPTIC PAND PASS FILTER.  IF THE ANALOG
C     FUNCTION ALONE IS DESIRED THEN WE USE WO FOR CALCULATIONS.
C     IF THE ANALOG TRANSFER FUNCTION IS DESIRED FOR DIGITAL
C     TRANSFORMATION THEN WE MUST USE THE PRE-WARPED ANALOG TO WO
C     WHICH IS WODIG.
C
C**********************************************************************
      RHO1=K1*SQRT(C/A)
      RHO2=K2*SQRT(C/A)
```

```
      COMPLEX AZERO(4),APOLE(4)
C SECTION 3
      INTEGER IZM,IZP,IDNUM,IDDEN,IDTMP
      REAL TNCOEF,TDCOEF
      REAL ZM1(5),ZM2(5),ZM3(5),ZM4(5),ZP1(5),ZP2(5),ZP3(5),ZP4(5)
      REAL DNUM(5),DDEN(5),DTMP(9)
C SECTION 3A
      REAL DNMINV(5),DDNINV(5)
      REAL RZ(4),RP(4)
      COMPLEX DZERO(4),DPOLE(4)
C SECTION 4
      INTEGER I4,I2
      COMPLEX*16 DZ4(4),DP4(4),DZ21(2),DP21(2),DZ22(2),DP22(2)
      COMPLEX*16 DN4(4),DD4(4),DN21(2),DD21(2),DN22(2),DD22(2)
      COMPLEX DN45(5),DD45(5),DN213(3),DD213(3),DN223(3),DD223(3)
C***********************************************************************
C     TABULATED INPUT PARAMETERS FOR DESIRED SECOND ORDER LOWPASS
C     FILTER EQUIVALENT HAVING THE FOLLOWING CHARACTERISTICS:
C                N = 2
C                MAXIMUM PASSBAND RIPPLE WIDTH (PRW) = 2.0 DB
C                MINIMUM LOSS IN THE STOPBAND (MSL) = 30.0 DB
C                NORMALIZED TRANSITION WIDTH = 2.2921
C                FILTER GAIN (K) = 1.0
C***********************************************************************
      F0=600.
      A=21.164003
      B=0.787152
      C=0.842554
      Q=12.
      PI=3.1415927
      K=1.
C***********************************************************************
C     DERIVED PARAMETERS
C***********************************************************************
      E=(1./B)*SQRT((C+4.*Q**2.+SQRT((C+4.*Q**2.)**2.-(2.*B*Q)**2.))/2.)
      D=.5*((B*E/Q)+SQRT((B*E/Q)**2.-4.))
      A1=1.+(1./(2.*Q**2.))*(A+SQRT(A**2.+4.*A*Q**2.))
      K1=SQRT(K)
      K2=K1
      T=4.*192./6.666E+06
      TDIV2=T/2.
      WO=2.*PI*FO
      WODIG=(1./TDIV2)*ATAN(WO*TDIV2)
      FODIG=WODIG/(2.*PI)
      WRITE(4,10)A,B,C,D,E,A1,FO,WO,Q,K,K1,K2,WODIG,FODIG
C***********************************************************************
C                                                                      *
C                              SECTION 1                               *
C                                                                      *
C     THIS PROGRAM SECTION COMPUTES ANALOG ELLIPTIC BANDPASS FILTER    *
C     RESISTOR AND CAPACITOR VALUES USING THE ABSOLUTE AND DERIVED     *
C     PARAMETERS CALCULATED ABOVE:                                     *
C                                                                      *
C     CHOOSE C11=C21=.01E-06 (APPROX)=.01 UF                           *
C     CHOOSE C12=C22=.01E-06(APPROX)=.01 UF                            *
```

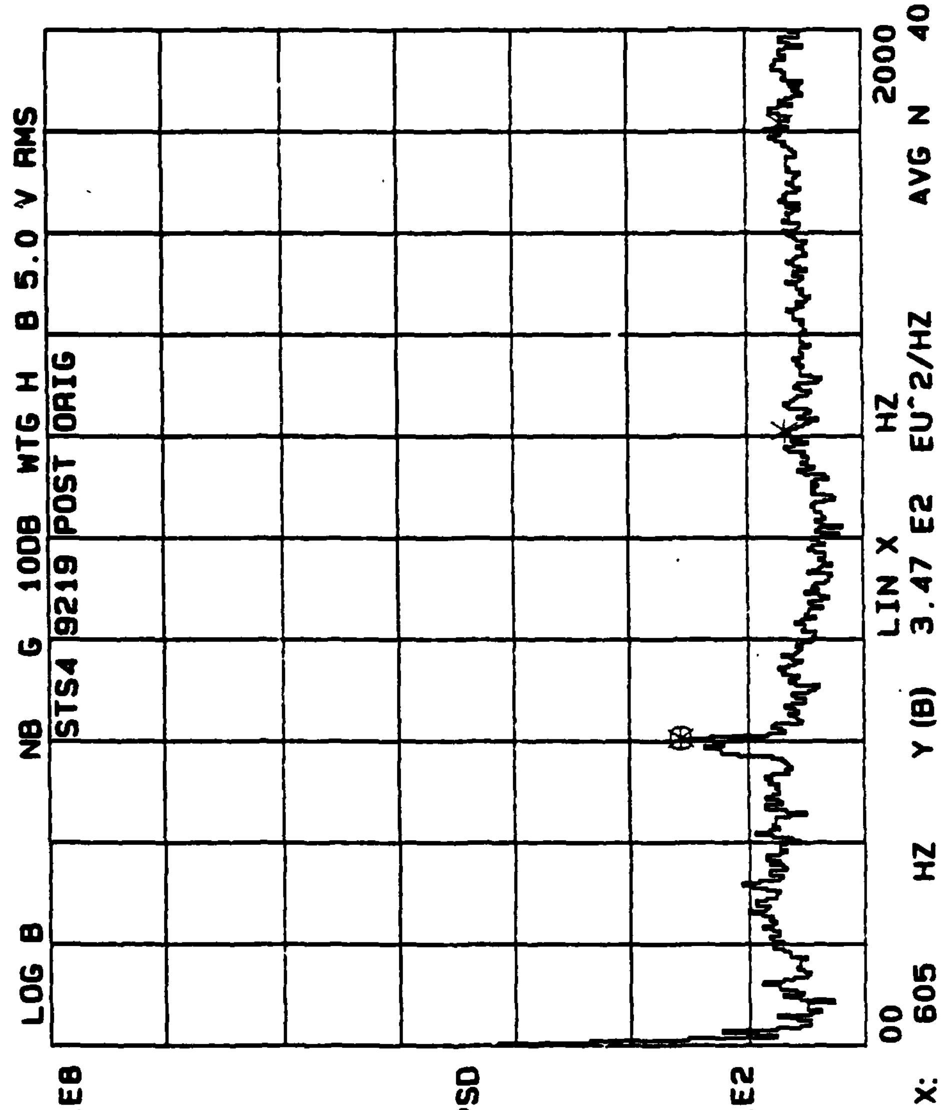
FILE: ABPDBP   FORTRAN   A1

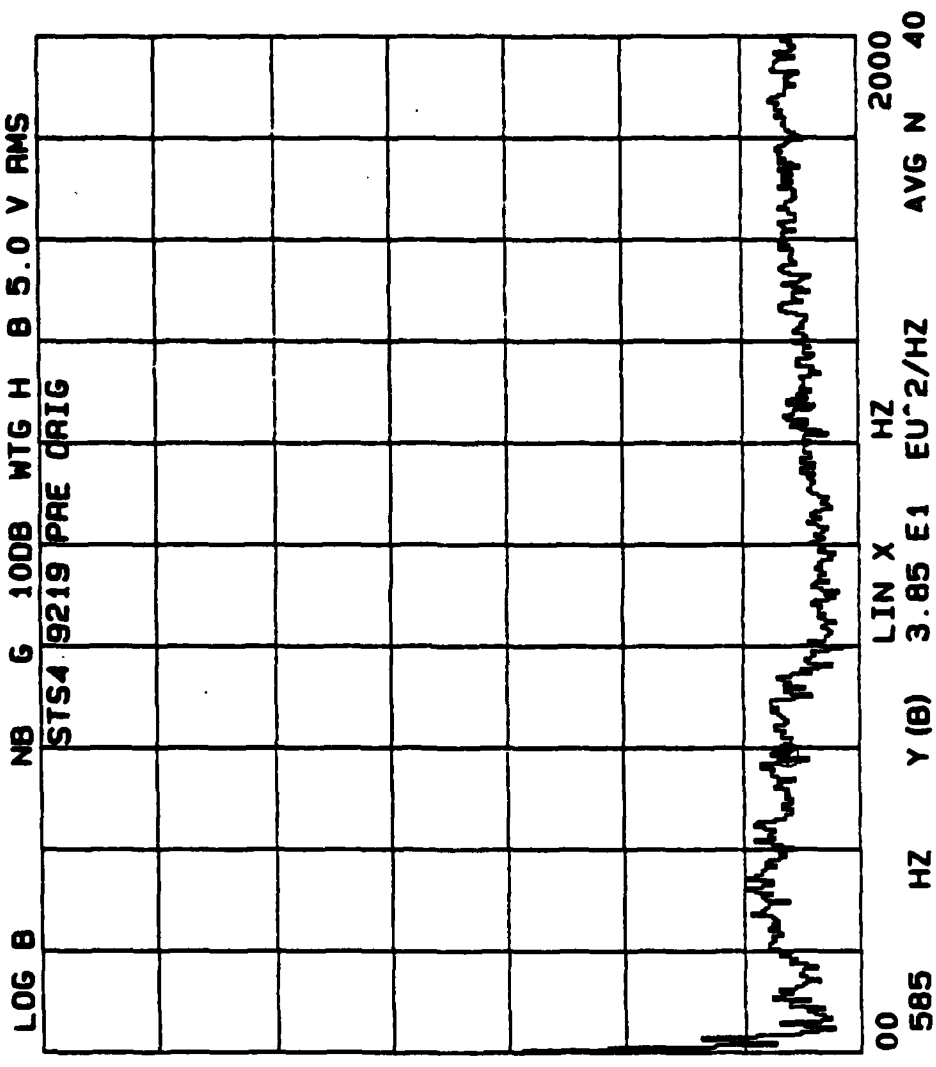
```
C************************************************************************
C                                                                       *
C                               APPENDIX C                              *
C                                                                       *
C             FORTRAN PROGRAM ABPDBP AND PROGRAM OUTPUT                 *
C                                                                       *
C                                                                       *
C     THIS PROGRAM CALCULATES:                                          *
C                                                                       *
C     1)  ANALOG ELLIPTIC BANDPASS FILTER RESISTOR AND CAPACITOR        *
C         VALUES (STARTING FROM TABULATED PARAMETERS CORRESPONDING      *
C         TO THE DESIRED FILTER RESPONSE)                               *
C                                                                       *
C     2)  ANALOG ELLIPTIC BANDPASS TRANSFER FUNCTION (IF DESIRED FOR    *
C         DIGITAL TRANSFORMATION THEN MUST MODIFY CODE IN THIS PROGRAM  *
C         SECTION AND SUBSTITUTE WODIG FOR WO TO REALIZE NECESSARY      *
C         PRE-WARPING COMPENSATION)                                     *
C                                                                       *
C         A)  ANALOG TRANSFER FUNCTION COMPLEX ZEROS AND POLES          *
C                                                                       *
C     3)  DIGITAL TRANSFER FUNCTION (BY APPLICATION OF THE BILINEAR     *
C         TRANSFORM TO THE PRE-WARPED CASE OF THE ANALOG TRANSFER       *
C         FUNCTION, WHICH IS PROVIDED IN SECTION 2 BY USING THE         *
C         PRE-WARPED FREQUENCY ANALOG -- WODIG (VICE WO) -- IN THE      *
C         ANALOG TRANSFER FUNCTION COMPUTATION).  THE BILINEAR          *
C         TRANSFORM IS ACCOMPLISHED BY THE FOLLOWING SUBSTITUTION:      *
C                                                                       *
C                                      Z - 1                            *
C                           S = (2/T) -------                           *
C                                      Z + 1                            *
C                                                                       *
C         WHERE T IS THE SAMPLING FREQUENCY OF THE DIGITAL SYSTEM.      *
C                                                                       *
C         A)  DIGITAL TRANSFER FUNCTION COMPLEX ZEROS AND POLES         *
C                                                                       *
C     4)  POLYNOMIAL COEFFICIENTS FOR FIRST AND SECOND ORDER CASCADED   *
C         TERMS WHICH WILL BE USED TO PERFORM A 2920 ANALOG/DIGITAL     *
C         SIGNAL PROCESSING SIMULATION                                  *
C                                                                       *
C************************************************************************
C    TYPE DECLARATIONS
C************************************************************************
C SECTION 1
      REAL A,B,C,D,E,A1,Q,K,K1,K2,WO,PI,FO,WODIG,FODIG,T,TDIV2
      REAL C11,C12,C21,C22
      REAL R11,R12,R13,R14,R15,R16,R17
      REAL R21,R22,R23,R24,R25,R26,R27
C SECTION 2
      INTEGER INUM1,INUM2,IDEN1,IDEN2,IANUM,IADEN
      REAL RHO,RHO1,RHO2,ALFA1,ALFA2,BETA1,BETA2,GAMMA1,GAMMA2
      REAL NUM1(3),NUM2(3),DEN1(3),DEN2(3)
      REAL ANUM(5),ADEN(5)
C SECTION 2A
      INTEGER IERR,IPDEG
      REAL ANMINV(5),ADNINV(5)
```

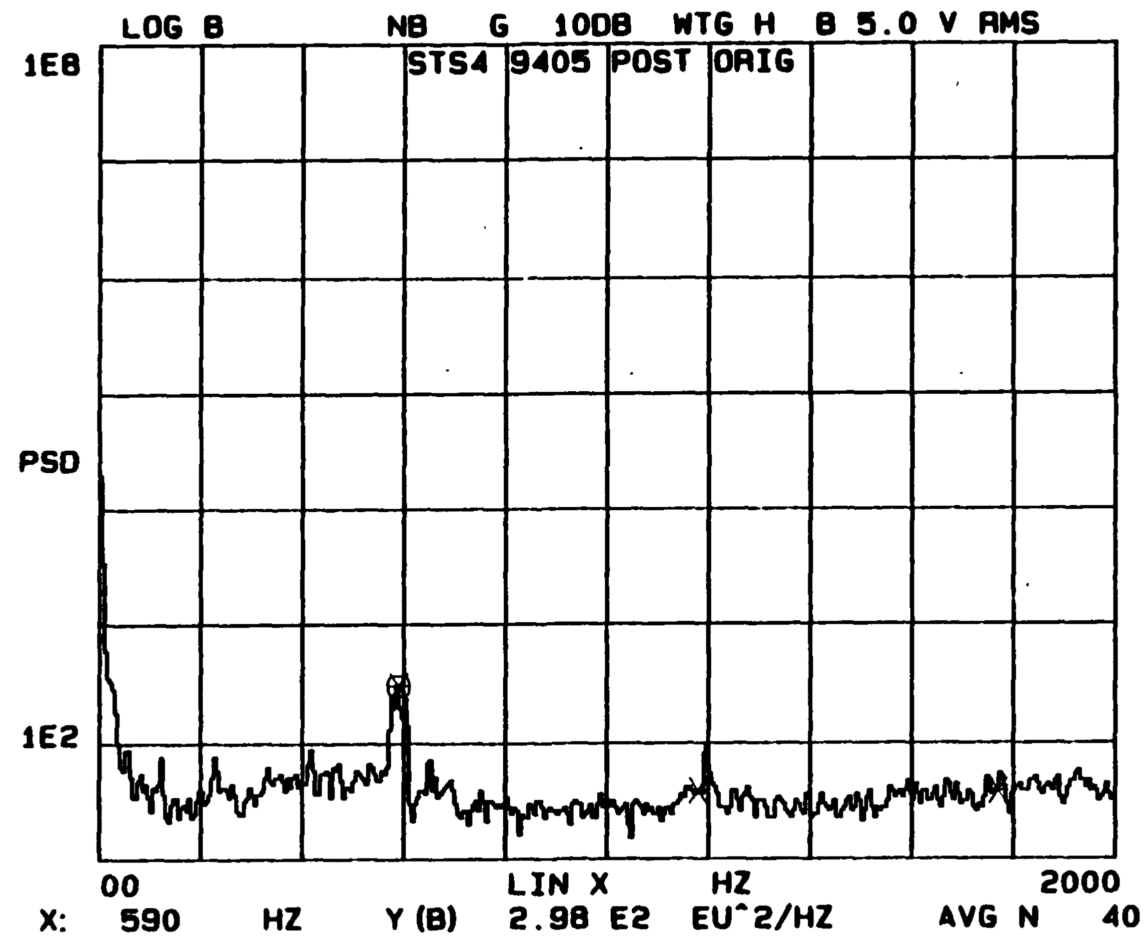

LOG B
HZ
NB
G
10DB
WTG H
B 5.0 V RMS
STS4
9403
POST
ORIG
1E8
PSD
1E2
X:
00
585
HZ
Y (B)
LIN X
6.35 E4
HZ
EU^2/HZ
AVG N
2000
40

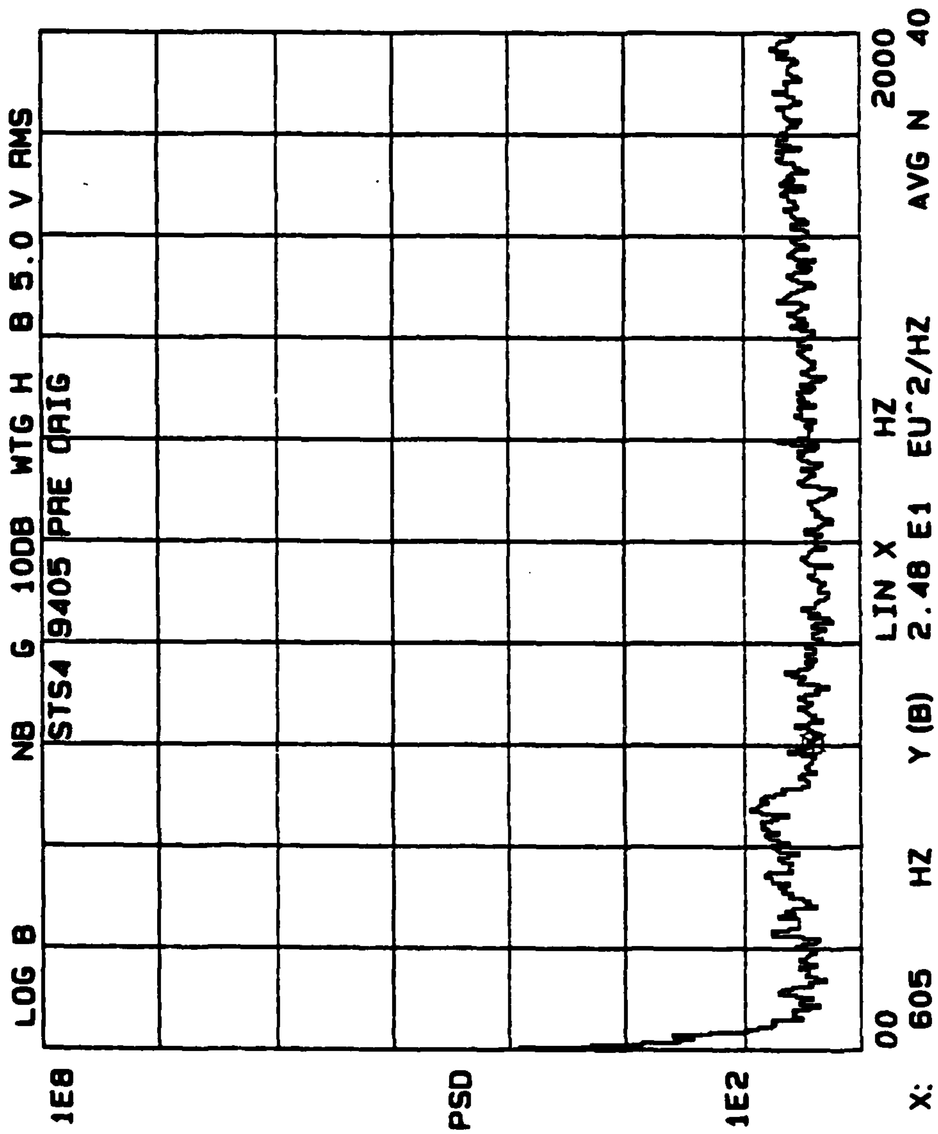


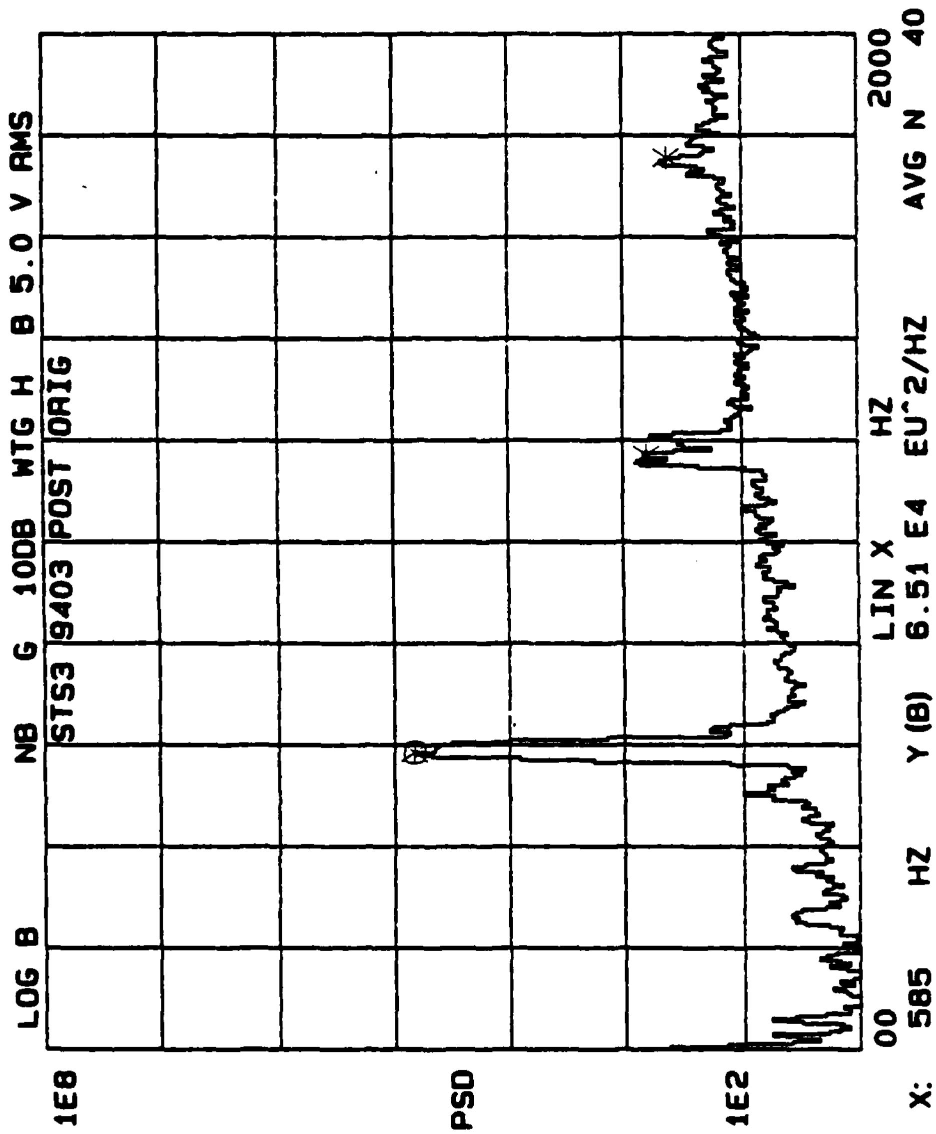
LOG B   NB   G   10DB   WTG H   B 5.0 V RMS
STS4 9403 PRE ORIG
1E8
PSD
1E2
X: 590 HZ   Y (B) 8.30   LIN X   HZ   AVG N 40
00   EU^2/HZ   2000

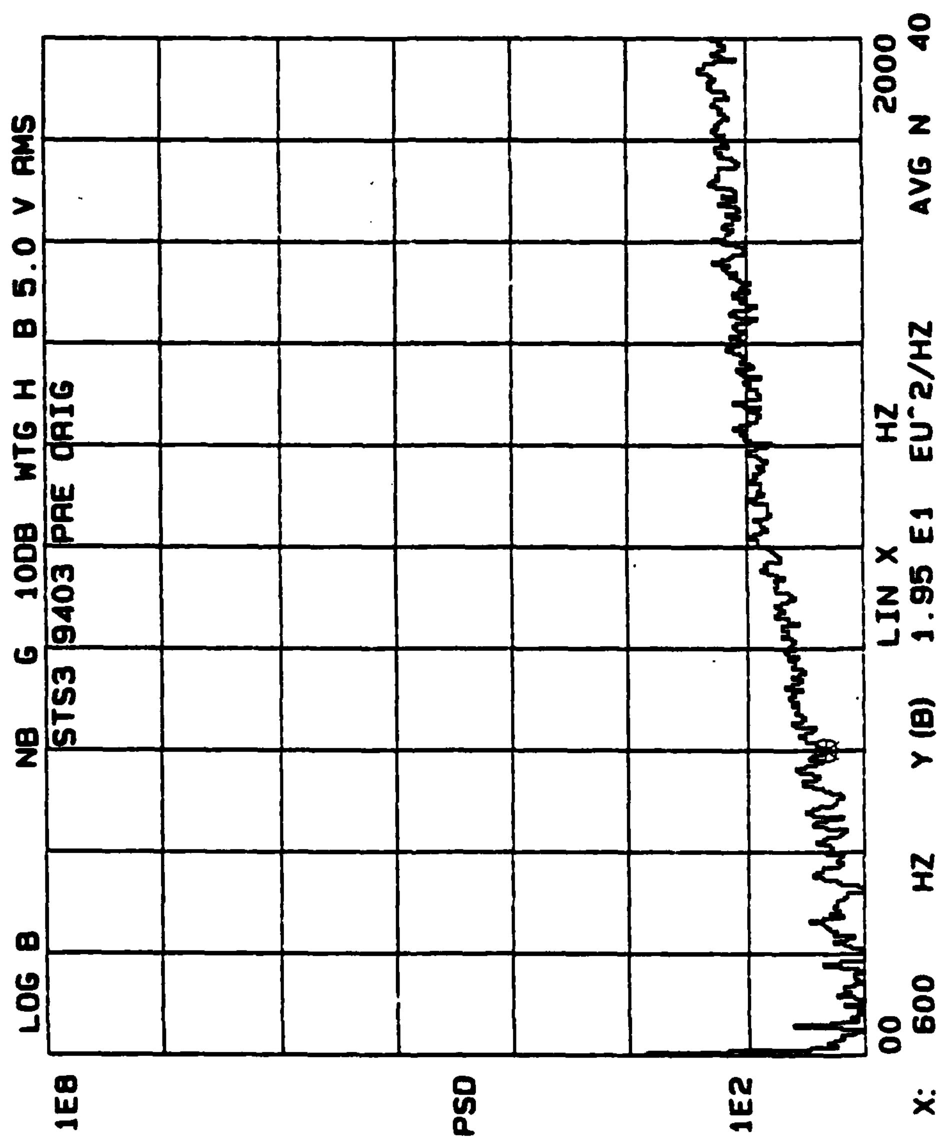

LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS4
9219
POST
ORIG
1E8
PSD
1E2
X:
00
605
HZ
Y (B)
LIN X
3.47 E2
HZ
EU^2/HZ
AVG N
2000
40

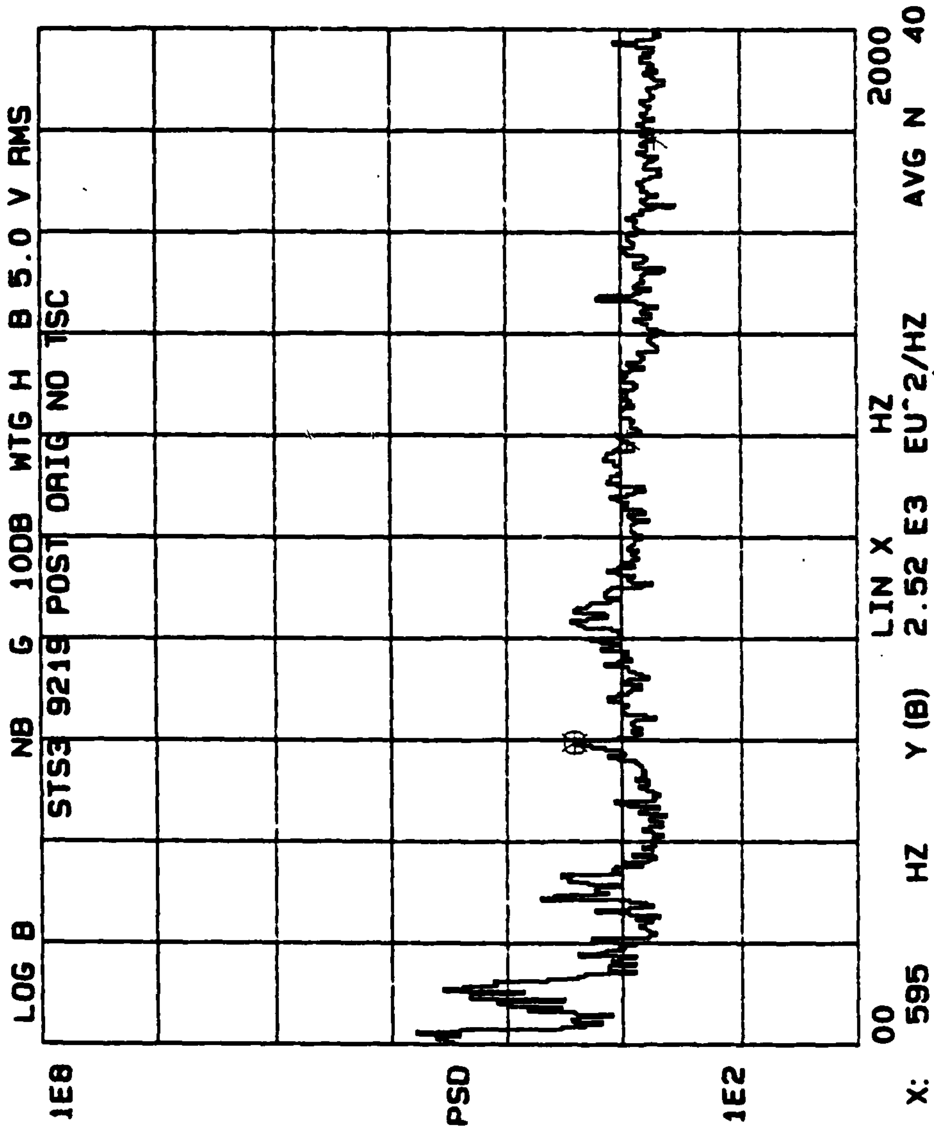

LOG B NB G 10DB WTG H B 5.0 V RMS
STS4 9219 PRE ORIG
1E8
PSD
1E2
X: 00 585 HZ Y (B) LIN X 3.85 E1 HZ EU^2/HZ AVG N
2000 40

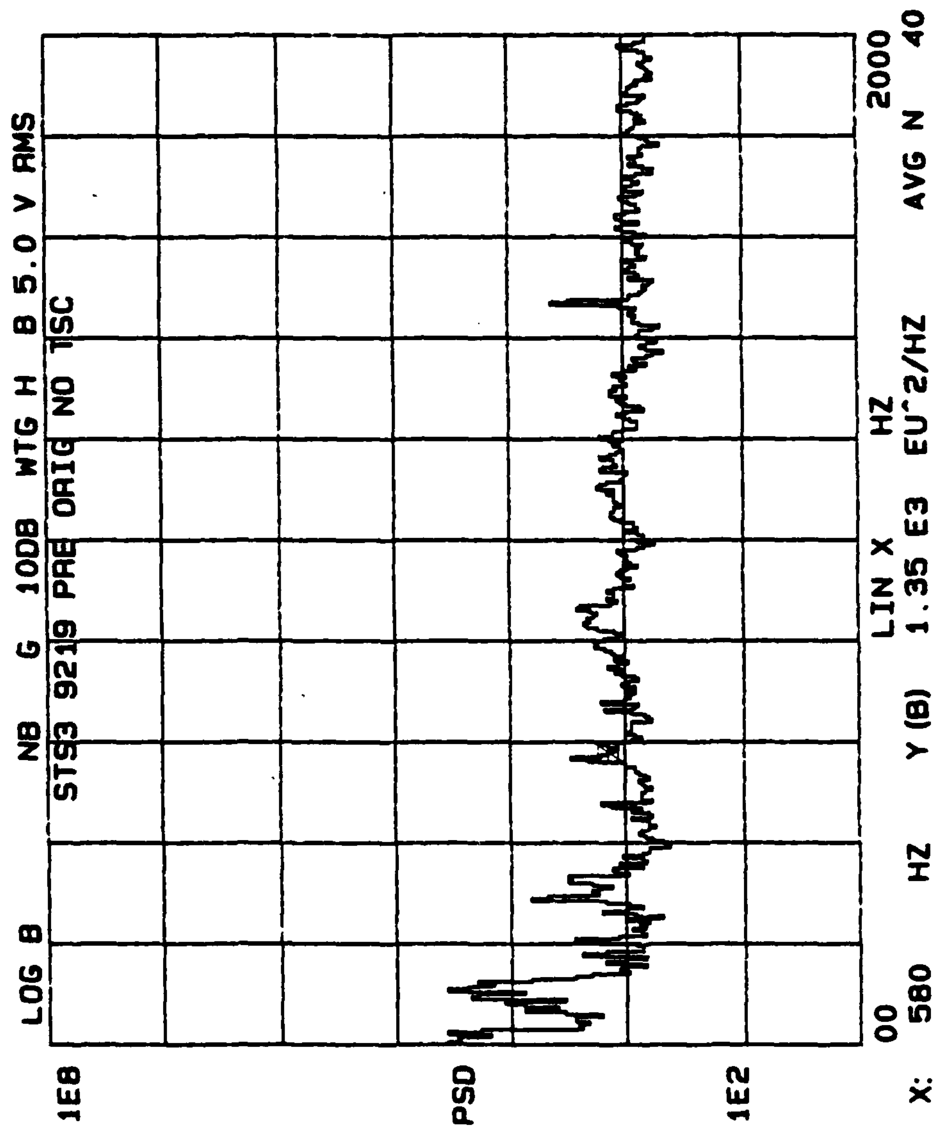

LOG B
NB G 10DB WTG H B 5.0 V RMS
STS4 9405 POST ORIG
1E8
PSD
1E2
00
LIN X
HZ
2000
X: 590 HZ Y (B) 2.98 E2 EU^2/HZ AVG N 40


LOG B HZ NB G 10DB WTG H B 5.0 V RMS
STS4 9405 PRE ORIG
1E8
PSD
1E2
00 LIN X HZ 2000
X: 605 HZ Y (B) 2.48 E1 EU^2/HZ AVG N 40


LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS3
9403
POST
ORIG
1E8
PSD
1E2
X:
00
585
HZ
Y (B)
LIN X
6.51 E4
HZ
EU^2/HZ
AVG N
2000
40


LOG B NB G 10DB WTG H B 5.0 V RMS
STS3 9403 PRE ORIG
1E8
PSD
1E2
X: 00 600 HZ Y (B) LIN X 1.95 E1
HZ EU^2/HZ
2000 AVG N 40


LOG B NB G 10DB WTG H B 5.0 V RMS
STS3 9219 POST ORIG NO TSC
1E8
PSD
1E2
X: 00 595 HZ Y (B) LIN X 2.52 E3 HZ EU^2/HZ AVG N
2000 40


LOG B
NB
G
10DB
WTG H
B 5.0 V RMS
STS3 9219 PRE ORIG NO TSC
1E8
PSD
1E2
X:
00
580
HZ
Y (B)
LIN X
1.35 E3
HZ
EU^2/HZ
2000
AVG N
40

```
      DO 1400 I=1,5                                                     ABP02210
         ANMINV(I)=ANUM(6-I)                                            ABP02220
         ADNINV(I)=ADEN(6-I)                                            ABP02230
 1400 CONTINUE                                                          ABP02240
      CALL ZRPOLY(ANMINV,IPDEG,AZERO,IERR)                              ABP02250
      CALL ZRPOLY(ADNINV,IPDEG,APOLE,IERR)                              ABP02260
      WRITE(4,1410)                                                     ABP02270
      WRITE(4,1420)AZERO(1),AZERO(2),AZERO(3),AZERO(4)                  ABP02280
      WRITE(4,1430)APOLE(1),APOLE(2),APOLE(3),APOLE(4)                  ABP02290
 1410 FORMAT(///,' SECTION 2A OUTPUT',//,                               ABP02300
     &' ANALOG FILTER COMPLEX POLES AND ZEROS',/)                       ABP02310
 1420 FORMAT(' ANALOG FILTER ZEROS',/,4(1X,E12.5,2X,E12.5,/),/)         ABP02320
 1430 FORMAT(' ANALOG FILTER POLES',/,4(1X,E12.5,2X,E12.5,/),///)       ABP02330
C***********************************************************************ABP02340
C                                                                      *ABP02350
C                             SECTION 3                                *ABP02360
C                                                                      *ABP02370
C     THE FOLLOWING STATEMENTS PERFORM THE BILINEAR TRANSFORMATION     *ABP02380
C     OF THE ANALOG BANDPASS TRANSFER FUNCTION COMPUTED ABOVE IN       *ABP02390
C     SECTION 2                                                        *ABP02400
C                                                                      *ABP02410
C***********************************************************************ABP02420
      IZM=5                                                             ABP02430
      IZP=5                                                             ABP02440
      IDNUM=5                                                           ABP02450
      IDDEN=5                                                           ABP02460
      ZM1(1)=-1.                                                        ABP02470
      ZM1(2)=1.                                                         ABP02480
      ZM1(3)=0.                                                         ABP02490
      ZM1(4)=0.                                                         ABP02500
      ZM1(5)=0.                                                         ABP02510
      ZM2(1)=1.                                                         ABP02520
      ZM2(2)=-2.                                                        ABP02530
      ZM2(3)=1.                                                         ABP02540
      ZM2(4)=0.                                                         ABP02550
      ZM2(5)=0.                                                         ABP02560
      ZM3(1)=-1.                                                        ABP02570
      ZM3(2)=3.                                                         ABP02580
      ZM3(3)=-3.                                                        ABP02590
      ZM3(4)=1.                                                         ABP02600
      ZM3(5)=0.                                                         ABP02610
      ZM4(1)=1.                                                         ABP02620
      ZM4(2)=-4.                                                        ABP02630
      ZM4(3)=6.                                                         ABP02640
      ZM4(4)=-4.                                                        ABP02650
      ZM4(5)=1.                                                         ABP02660
      ZP1(1)=1.                                                         ABP02670
      ZP1(2)=1.                                                         ABP02680
      ZP1(3)=0.                                                         ABP02690
      ZP1(4)=0.                                                         ABP02700
      ZP1(5)=0.                                                         ABP02710
      ZP2(1)=1.                                                         ABP02720
      ZP2(2)=2.                                                         ABP02730
      ZP2(3)=1.                                                         ABP02740
      ZP2(4)=0.                                                         ABP02750
```

```
      ZP2(5)=0.
      ZP3(1)=1.
      ZP3(2)=3.
      ZP3(3)=3.
      ZP3(4)=1.
      ZP3(5)=0.
      ZP4(1)=1.
      ZP4(2)=4.
      ZP4(3)=6.
      ZP4(4)=4.
      ZP4(5)=1.
      TNCOEF=ANUM(1)*(TDIV2**4.)
      TDCOEF=ADEN(1)*(TDIV2**4.)
      DO 200 I=1,5
         DNUM(I)=ZP4(I)*TNCOEF
         DDEN(I)=ZP4(I)*TDCOEF
  200 CONTINUE
      TNCOEF=ANUM(2)*(TDIV2**3.)
      TDCOEF=ADEN(2)*(TDIV2**3.)
      DO 210 I=1,5
         CALL PMPY(DTMP,IDTMP,ZM1,IZM,ZP3,IZP)
         DNUM(I)=DNUM(I)+(DTMP(I)*TNCOEF)
         DDEN(I)=DDEN(I)+(DTMP(I)*TDCOEF)
  210 CONTINUE
      TNCOEF=ANUM(3)*(TDIV2**2.)
      TDCOEF=ADEN(3)*(TDIV2**2.)
      DO 220 I=1,5
         CALL PMPY(DTMP,IDTMP,ZM2,IZM,ZP2,IZP)
         DNUM(I)=DNUM(I)+(DTMP(I)*TNCOEF)
         DDEN(I)=DDEN(I)+(DTMP(I)*TDCOEF)
  220 CONTINUE
      TNCOEF=ANUM(4)*(TDIV2)
      TDCOEF=ADEN(4)*(TDIV2)
      DO 230 I=1,5
         CALL PMPY(DTMP,IDTMP,ZM3,IZM,ZP1,IZP)
         DNUM(I)=DNUM(I)+(DTMP(I)*TNCOEF)
         DDEN(I)=DDEN(I)+(DTMP(I)*TDCOEF)
  230 CONTINUE
      TNCOEF=ANUM(5)
      TDCOEF=ADEN(5)
      DO 240 I=1,5
         DNUM(I)=DNUM(I)+(ZM4(I)*TNCOEF)
         DDEN(I)=DDEN(I)+(ZM4(I)*TDCOEF)
  240 CONTINUE
      DO 250 I=1,5
         DNUM(I)=DNUM(I)/DDEN(5)
         DDEN(I)=DDEN(I)/DDEN(5)
  250 CONTINUE
      RHO=RHO*DNUM(5)
      DO 260 I=1,5
         DNUM(I)=DNUM(I)/DNUM(5)
  260 CONTINUE
      WRITE(4,310)RHO
      WRITE(4,320)DNUM(5),DNUM(4),DNUM(3),DNUM(2),DNUM(1)
      DO 305 I=1,5
```

```
          DNUM(I)=DNUM(I)*RHO
  305 CONTINUE
      WRITE(4,330)DNUM(5),DNUM(4),DNUM(3),DNUM(2),DNUM(1)
      WRITE(4,340)DDEN(5),DDEN(4),DDEN(3),DDEN(2),DDEN(1)
  310 FORMAT('1','SECTION 3 OUTPUT',//,
     &' EQUIVALENT DIGITAL FILTER TRANSFER FUNCTION',//,
     &' NUMERATOR COEFFICIENT = ',E12.5,/)
  320 FORMAT(' NUMERATOR (NORMALIZED)'
     &,/,1X,E12.5,' + ',/,1X,E12.5,' Z**-1 +',/,1X,E12.5,' Z**-2 +',
     &/,1X,E12.5,' Z**-3 +',/,1X,E12.5,' Z**-4',//)
  330 FORMAT(' NUMERATOR (UN-NORMALIZED)'
     &,/,1X,E12.5,' + ',/,1X,E12.5,' Z**-1 +',/,1X,E12.5,' Z**-2 +',
     &/,1X,E12.5,' Z**-3 +',/,1X,E12.5,' Z**-4',//)
  340 FORMAT(' DENOMINATOR'
     &,/,1X,E12.5,' + ',/,1X,E12.5,' Z**-1 +',/,1X,E12.5,' Z**-2 +',
     &/,1X,E12.5,' Z**-3 +',/,1X,E12.5,' Z**-4',////)
C***********************************************************************
C                                                                      *
C                             SECTION 3A                               *
C                                                                      *
C     CALCULATE THE COMPLEX POLES AND ZEROS OF THE DIGITAL FILTER      *
C     AND THE RADIUS OF THE COMPLEX POLE AND ZERO VECTORS RELATIVE     *
C     TO THE UNIT CIRCLE.                                              *
C                                                                      *
C***********************************************************************
      IPDEG = 4
      DO 400 I=1,5
          DNMINV(I)=DNUM(6-I)
          DDNINV(I)=DDEN(6-I)
  400 CONTINUE
      CALL ZRPOLY(DNMINV,IPDEG,DZERO,IERR)
      CALL ZRPOLY(DDNINV,IPDEG,DPOLE,IERR)
      DO 404 I=1,4
          RZ(I)=SQRT((REAL(DZERO(I)))**2.+(AIMAG(DZERO(I)))**2.)
          RP(I)=SQRT((REAL(DPOLE(I)))**2.+(AIMAG(DPOLE(I)))**2.)
  404 CONTINUE
      WRITE(4,410)
      WRITE(4,420)DZERO(1),RZ(1),DZERO(2),RZ(2),DZERO(3),RZ(3),
     &DZERO(4),RZ(4)
      WRITE(4,430)DPOLE(1),RP(1),DPOLE(2),RP(2),DPOLE(3),RP(3),
     &DPOLE(4),RP(4)
  410 FORMAT(' SECTION 3A OUTPUT',//,
     &' DIGITAL FILTER COMPLEX POLES AND ZEROS AND RADIUS',//)
  420 FORMAT(' ZERO LOCATIONS (REAL,IMAG) AND RADIUS',/,
     &4(E12.5,2X,E12.5,2X,E12.5,/),/)
  430 FORMAT(' POLE LOCATIONS (REAL,IMAG) AND RADIUS',/,
     &4(E12.5,2X,E12.5,2X,E12.5,/),///)
C***********************************************************************
C                                                                      *
C                             SECTION 4                                *
C                                                                      *
C     COMPUTE THE POLYNOMIAL COEFFICIENTS OF QUADRATIC FACTORS FOR     *
C     THE POLES AND ZEROS PREVIOUSLY DETERMINED.                       *
C                                                                      *
C***********************************************************************
```

```
      I4=4
      I2=2
      DO 500 I=1,4
         DZ4(I)=CMPLX(REAL(DZERO(I)),AIMAG(DZERO(I)))
         DP4(I)=CMPLX(REAL(DPOLE(I)),AIMAG(DPOLE(I)))
  500 CONTINUE
      DO 502 I=1,2
         DZ21(I)=DZ4(I)
         DP21(I)=DP4(I)
  502 CONTINUE
      DO 504 I=3,4,1
         DZ22(I-2)=DZ4(I)
         DP22(I-2)=DP4(I)
  504 CONTINUE
      CALL MAKPOL(I4,DZ4,DN4)
      CALL MAKPOL(I4,DP4,DD4)
      DN45(5)=CMPLX(1.,0.)
      DD45(5)=CMPLX(1.,0.)
      DO 512 I=1,4
         DN45(I)=DN4(I)
         DD45(I)=DD4(I)
  512 CONTINUE
      WRITE(4,514)
  514 FORMAT('1','SECTION 4 OUTPUT',//,
     &' REASSEMBLE COEFFICIENTS FROM POLES AND ZEROS',//)
      WRITE(4,515)
      WRITE(4,520)DN45(5),DN45(4),DN45(3),DN45(2),DN45(1)
      WRITE(4,530)DD45(5),DD45(4),DD45(3),DD45(2),DD45(1)
  515 FORMAT(/' SINGLE FOURTH ORDER TRANSFER FUNCTION COEFFICIENTS',//)
  520 FORMAT(' FOURTH ORDER NUMERATOR COEFFICIENTS (NORMALIZED)',/,
     &5(1X,E12.5,2X,E12.5,/))
  530 FORMAT(' FOURTH ORDER DENOMINATOR COEFFICIENTS (NORMALIZED)',/,
     &5(1X,E12.5,2X,E12.5,/),//)
      CALL MAKPOL(I2,DZ21,DN21)
      CALL MAKPOL(I2,DP21,DD21)
      DN213(3)=CMPLX(1.,0.)
      DD213(3)=CMPLX(1.,0.)
      DO 612 I=1,2
         DN213(I)=DN21(I)
         DD213(I)=DD21(I)
  612 CONTINUE
      WRITE(4,614)
  614 FORMAT(' CASCADED SECOND ORDER TRANSFER FUNCTION COEFFICIENTS',/)
      WRITE(4,615)
      WRITE(4,620)DN213(3),DN213(2),DN213(1)
      WRITE(4,630)DD213(3),DD213(2),DD213(1)
  615 FORMAT(/' FIRST STAGE QUADRATIC FUNCTION COEFFICIENTS',/)
  620 FORMAT(' COMPLEX (REAL,IMAG) NUMERATOR COEFFICIENTS',/,
     &1X,E12.5,2X,E12.5,
     &'  Z**2 +',/,1X,E12.5,2X,E12.5,'  Z +',/,1X,E12.5,2X,E12.5,/)
  630 FORMAT(' COMPLEX (REAL,IMAG) DENOMINATOR COEFFICIENTS',/,
     &1X,E12.5,2X,E12.5,
     &'  Z**2 +',/,1X,E12.5,2X,E12.5,'  Z +',/,1X,E12.5,2X,E12.5,/)
      CALL MAKPOL(I2,DZ22,DN22)
      CALL MAKPOL(I2,DP22,DD22)
```

```
      DN223(3)=CMPLX(1.,0.)
      DD223(3)=CMPLX(1.,0.)
      DO 712 I=1,2
         DN223(I)=DN22(I)
         DD223(I)=DD22(I)
  712 CONTINUE
      WRITE(4,715)
      WRITE(4,720)DN223(3),DN223(2),DN223(1)
      WRITE(4,730)DD223(3),DD223(2),DD223(1)
  715 FORMAT(/' SECOND STAGE QUADRATIC FUNCTION COEFFICIENTS',//)
  720 FORMAT(' COMPLEX (REAL,IMAG) NUMERATOR COEFFICIENTS',/,
     &1X,E12.5,2X,E12.5,
     &'  Z**2 +',/,1X,E12.5,2X,E12.5,'  Z +',/,1X,E12.5,2X,E12.5,/)
  730 FORMAT(' COMPLEX (REAL,IMAG) DENOMINATOR COEFFICIENTS',/,
     &1X,E12.5,2X,E12.5,
     &'  Z**2 +',/,1X,E12.5,2X,E12.5,'  Z +',/,1X,E12.5,2X,E12.5,/)
      STOP
      END
C
C     ..............................................................
C
C        SUBROUTINE PMPY
C
C        PURPOSE
C           MULTIPLY TWO POLYNOMIALS
C
C        USAGE
C           CALL PMPY(Z,IDIMZ,X,IDIMX,Y,IDIMY)
C
C        DESCRIPTION OF PARAMETERS
C           Z     - VECTOR OF RESULTANT COEFFICIENTS, ORDERED FROM
C                   SMALLEST TO LARGEST POWER
C           IDIMZ - DIMENSION OF Z (CALCULATED)
C           X     - VECTOR OF COEFFICIENTS FOR FIRST POLYNOMIAL, ORDERED
C                   FROM SMALLEST TO LARGEST POWER
C           IDIMX - DIMENSION OF X (DEGREE IS IDIMX-1)
C           Y     - VECTOR OF COEFFICIENTS FOR SECOND POLYNOMIAL,
C                   ORDERED FROM SMALLEST TO LARGEST POWER
C           IDIMY - DIMENSION OF Y (DEGREE IS IDIMY-1)
C
C        REMARKS
C           Z CANNOT BE IN THE SAME LOCATION AS X
C           Z CANNOT BE IN THE SAME LOCATION AS Y
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           DIMENSION OF Z IS CALCULATED AS IDIMX+IDIMY-1
C           THE COEFFICIENTS OF Z ARE CALCULATED AS SUM OF PRODUCTS
C           OF COEFFICIENTS OF X AND Y , WHOSE EXPONENTS ADD UP TO THE
C           CORRESPONDING EXPONENT OF Z.
C
C     ..............................................................
C
```

```
      SUBROUTINE PMPY(Z,IDIMZ,X,IDIMX,Y,IDIMY)
      DIMENSION Z(1),X(1),Y(1)
C
      IF(IDIMX*IDIMY)10,10,20
   10 IDIMZ=0
      GO TO 50
   20 IDIMZ=IDIMX+IDIMY-1
      DO 30 I=1,IDIMZ
   30 Z(I)=0.
      DO 40 I=1,IDIMX
      DO 40 J=1,IDIMY
      K=I+J-1
   40 Z(K)=X(I)*Y(J)+Z(K)
   50 RETURN
      END
C
C-----------------------------------------------------------------------
C
C       SUBROUTINE MAKPOL
C
C       PURPOSE
C          TO COMPUTE THE COMPLEX COEFFICIENTS OF AN N-TH DEGREE POLY-
C          NOMIAL GIVEN N COMPLEX ROOTS OF THE POLYNOMIAL
C
C       USAGE
C          CALL MAKPOL(N,R,C)
C
C       DESCRIPTION OF PARAMETERS
C          N - NUMBER OF ROOTS GIVEN AND DEGREE OF POLYNOMIAL.  THE
C              COEFFICIENT OF THE HIGHEST POWER OF THE UNKNOWN IS ALWAYS
C              UNITY, AND IS NOT COMPUTED BY "MAKPOL".
C          R - DOUBLE PRECISION COMPLEX ARRAY CONTAINING THE COMPLEX
C              ROOTS
C          C - DOUBLE PRECISION COMPLEX ARRAY CONTAINING THE COMPLEX
C              COEFFICIENTS
C
C       REMARKS
C          ARRAYS R AND C ARE TYPED COMPLEX*16
C
      SUBROUTINE MAKPOL(N,R,C)
      COMPLEX*16 R(N),C(N)
      IF(N.LE.0) RETURN
      DO 10 I=1,N
   10 C(I)=R(I)
      K=N
      M=N-1
      DO 20 L=1,M
      DO 30 I=2,K
   30 C(I)=C(I)+C(I-1)
      K=K-1
      DO 20 I=1,K
      J=I+L
      C(I)=R(J)*C(I)
   20 CONTINUE
      K=N/2
```

FILE: ABPDBP   FORTRAN A1

```
      K=2*K/(N-K)                                                       ABP05510
      DO 40 I=K,N,2                                                     ABP05520
   40 C(I)=-C(I)                                                        ABP05530
      RETURN                                                            ABP05540
      END                                                               ABP05550
```

ABPDBP OUTPUT

SECTION 1 OUTPUT

INPUT AND DERIVED PARAMETERS FOR FURTHER CALCULATIONS

```
A = 21.164001
B =  0.787152
C =  0.842554
D =  1.035090
E = 30.507828
A1 =  1.463835
F0 =   600.000
W0 =  3769.911
Q = 12.000000
K =  1.000000
K1 =  1.000000
K2  1.000000
W0(DIG) =  3712.266
F0(DIG) =   590.825
```

ELLIPTIC ANALOG BPF COMPONENT VALUES

FIRST STAGE

```
C11 = .996E-08
C12 = .995E-08
R17 = .267E+05
R11 = .393E+07
R12 = .785E+06
R13 = .257E+05
R14 = .134E+06
R15 = .945E+05
R16 = .258E+05
```

SECOND STAGE

```
C21 = .103E-07
C22 = .103E-07
R27 = .267E+05
R21 = .408E+07
R22 = .813E+06
R23 = .267E+05
R24 = .134E+06
R25 = .182E+06
R26 = .266E+05
```

SECTION 2 OUTPUT

ANALOG ELLIPTIC BANDPASS FILTER TRANSFER FUNCTION

NUMERATOR COEFFICIENT = RHO = 0.039811

```
NUMERATOR POLYNOMIAL (NORMALIZED)
0.10000E+01 S**4 +
0.0         S**3 +
0.29587E+08 S**2 +
0.0         S    +
0.18991E+15


NUMERATOR POLYNOMIAL (UN-NORMALIZED)
0.39811E-01 S**4 +
0.0         S**3 +
0.11779E+07 S**2 +
0.0         S    +
0.75606E+13


DENOMINATOR POLYNOMIAL (NORMALIZED)
0.10000E+01 S**4 +
0.24351E+03 S**3 +
0.27642E+08 S**2 +
0.33558E+10 S    +
0.18991E+15
```

SECTION 2A OUTPUT

ANALOG FILTER COMPLEX POLES AND ZEROS

```
ANALOG FILTER ZEROS
-0.45776E-03   0.30683E+04
-0.45776E-03  -0.30683E+04
 0.45776E-03   0.44914E+04
 0.45776E-03  -0.44914E+04

ANALOG FILTER POLES
-0.58188E+02   0.35859E+04
-0.58188E+02  -0.35859E+04
-0.63567E+02   0.38421E+04
-0.63567E+02  -0.38421E+04
```

SECTION 3 OUTPUT

EQUIVALENT DIGITAL FILTER TRANSFER FUNCTION

NUMERATOR COEFFICIENT = 0.39516E-01

NUMERATOR (NORMALIZED)
```
 0.10000E+01 +
-0.36279E+01 Z**-1 +
 0.52861E+01 Z**-2 +
-0.36279E+01 Z**-3 +
 0.10000E+01 Z**-4
```

NUMERATOR (UN-NORMALIZED)
```
 0.39516E-01 +
-0.14336E+00 Z**-1 +
 0.20888E+00 Z**-2 +
-0.14336E+00 Z**-3 +
 0.39516E-01 Z**-4
```

DENOMINATOR
```
 0.10000E+01 +
-0.36251E+01 Z**-1 +
 0.52586E+01 Z**-2 +
-0.35768E+01 Z**-3 +
 0.97353E+00 Z**-4
```

SECTION 3A OUTPUT

DIGITAL FILTER.COMPLEX POLES AND ZEROS AND RADIUS

ZERO LOCATIONS (REAL, IMAG) AND RADIUS
```
0.87496E+00   0.48595E+00   0.10009E+01
0.87496E+00  -0.48595E+00   0.10009E+01
0.93896E+00   0.34135E+00   0.99909E+00
0.93896E+00  -0.34135E+00   0.99909E+00
```

POLE LOCATIONS (REAL, IMAG) AND RADIUS
```
0.90781E+00   0.40813E+00   0.99533E+00
0.90781E+00  -0.40813E+00   0.99533E+00
0.90475E+00   0.40493E+00   0.99123E+00
0.90475E+00  -0.40493E+00   0.99123E+00
```

```
SECTION 4 OUTPUT

REASSEMBLE COEFFICIENTS FROM POLES AND ZEROS


SINGLE FOURTH ORDER TRANSFER FUNCTION COEFFICIENTS


FOURTH ORDER NUMERATOR COEFFICIENTS (NORMALIZED)
 0.10000E+01    0.0
-0.36279E+01    0.0
 0.52861E+01    0.0
-0.36279E+01    0.0
 0.99988E+00    0.0

FOURTH ORDER DENOMINATOR COEFFICIENTS (NORMALIZED)
 0.10000E+01    0.0
-0.36251E+01    0.0
 0.52586E+01    0.0
-0.35766E+01   -0.18041E-15
 0.97339E+00    0.0


CASCADED SECOND ORDER TRANSFER FUNCTION COEFFICIENTS


FIRST STAGE QUADRATIC FUNCTION COEFFICIENTS

COMPLEX (REAL,IMAG) NUMERATOR COEFFICIENTS
 0.10000E+01    0.0          Z**2 +
-0.17499E+01    0.0          Z +
 0.10017E+01    0.0

COMPLEX (REAL,IMAG) DENOMINATOR COEFFICIENTS
 0.10000E+01    0.0          Z**2 +
-0.18156E+01    0.0          Z +
 0.99068E+00    0.0


SECOND STAGE QUADRATIC FUNCTION COEFFICIENTS


COMPLEX (REAL,IMAG) NUMERATOR COEFFICIENTS
 0.10000E+01    0.0          Z**2 +
-0.18779E+01    0.0          Z +
 0.99817E+00    0.0

COMPLEX (REAL,IMAG) DENOMINATOR COEFFICIENTS
 0.10000E+01    0.0          Z**2 +
-0.18095E+01    0.0          Z +
 0.98255E+00    0.0
```

FILE: ABPFR   FORTRAN A1

```
C************************************************************************
C                                                                       *
C                            APPENDIX D                                 *
C                                                                       *
C                     FORTRAN PROGRAM ABPFR                             *
C                                                                       *
C                                                                       *
C    PROGRAM TO PLOT ANALOG BAND-PASS FILTER FREQUENCY AND PHASE        *
C    RESPONSE OF THE ELLIPTIC FILTER TRANSFER FUNCTION                  *
C                                                                       *
C************************************************************************
C     TYPE DECLARATIONS
      IMPLICIT REAL(A-H,O-Z),INTEGER(I-N)
      REAL W(1001),HMAG(1001),HMAGN(1001),HMAGDB(1001),WX,F(1001)
      REAL A(5),B(5),HPHASE(1001)
      COMPLEX S,H
C     NORMALIZED ANALOG TRANSFER FUNCTION COEFFICIENTS
      A(1) = 1.
      A(2) = 0.0
      A(3) = 0.30513E+8
      A(4) = 0.0
      A(5) = 0.20198E+15
      B(1) = 1.0
      B(2) = 0.24729E+3
      B(3) = 0.28507E+8
      B(4) = 0.35145E+10
      B(5) = 0.20198E+15
C     CONSTANTS
      PI = 3.1415927
C     EVALUATE MAGNITUDE AND PHASE OF H(JW)
      DO 10 I = 1,1001
         F(I) = FLOAT(I-1)
         W(I) = (2.*PI*F(I))
         S = CMPLX(0.,W(I))
         H = (A(1)*(S**4)+A(2)*(S**3)+A(3)*(S**2)+A(4)*S+A(5))/
     &(B(1)*(S**4)+B(2)*(S**3)+B(3)*(S**2)+B(4)*S+B(5))
         HMAG(I) = CABS(H)
         X = REAL(H)
         Y = AIMAG(H)
         HPHASE(I) = ATAN(Y/X)*180./PI
   10 CONTINUE
C     NORMALIZE MAGNITUDE
      AMAX = 0.0
      DO 20 I = 1,1001
         IF(HMAG(I).GT.AMAX) AMAX = HMAG(I)
   20 CONTINUE
      DO 30 I = 1,1001
         HMAGN(I) = HMAG(I)/AMAX
         HMAGDB(I) = 20.0 * ALOG10(HMAG(I))
   30 CONTINUE
      DO 40 I = 1,1001
         WRITE (4,50)I,F(I),W(I),HMAG(I),HMAGN(I),HMAGDB(I),HPHASE(I)
   50 FORMAT(I4,6(1X,E10.3))
   40 CONTINUE
C-----------------------------------------------------------------------
```

```
C GRAPHICS PARAMETERS FOR MAGNITUDE VS FREQUENCY
C-----------------------------------------------------------------------
      CALL LRGBUF
      CALL COMPRS
C     CALL TEK618
C     CALL VRSTEC(0,0,0)
C .....     SETUP THE PLOTTING AREA
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....  LABEL THE X & Y AXES
      CALL XNAME('FREQUENCY (HZ)$',100)
      CALL YNAME('AMPLITUDES$',100)
      CALL HEADIN('ANALOG ELLIPTIC BPF FREQUENCY RESPONSE$',100,1.6,2)
      CALL HEADIN('AMPLITUDE VS FREQ (F0=600 HZ)$',100,1.,2)
C .....   DEFINE THE AXES
      CALL GRAF(0.0,'SCALE',1000.,-0.5,'SCALE',+1.5)
C..... DRAW THE CURVES
C     CALL THKCRV(0.02)
      CALL MARKER(15)
      CALL CURVE(F,HMAGN,1001,0)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
C-----------------------------------------------------------------------
C GRAPHICS PARAMETERS FOR MAGNITUDE IN DBS VS FREQUENCY
C-----------------------------------------------------------------------
      CALL LRGBUF
      CALL COMPRS
C     CALL TEK618
C     CALL VRSTEC(0,0,0)
C .....     SETUP THE PLOTTING AREA
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....  LABEL THE X & Y AXES
      CALL XNAME('FREQUENCY (HZ)$',100)
      CALL YNAME('AMPLITUDE (DB)$',100)
      CALL HEADIN('ANALOG ELLIPTIC BPF FREQUENCY RESPONSE$',100,1.6,2)
      CALL HEADIN('AMPLITUDE (DB) VS FREQ (F0=600 HZ)$',100,1.,2)
C .....   DEFINE THE AXES
      CALL GRAF(0.0,'SCALE',1000.,-60.0,'SCALE',30.0)
C..... DRAW THE CURVES
C     CALL THKCRV(0.02)
      CALL MARKER(15)
      CALL CURVE(F,HMAGDB,1001,0)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
C-----------------------------------------------------------------------
C GRAPHICS PARAMETERS FOR PHASE VS FREQUENCY
C-----------------------------------------------------------------------
      CALL LRGBUF
      CALL COMPRS
C     CALL TEK618
C     CALL VRSTEC(0,0,0)
C .....     SETUP THE PLOTTING AREA
```

FILE: ABPFR FORTRAN A1

```
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....  LABEL THE X & Y AXES
      CALL XNAME('FREQUENCY (HZ)$',100)
      CALL YNAME('PHASE (DEGREES)$',100)
      CALL HEADIN('ANALOG ELLIPTIC BPF PHASE RESPONSE$',100,1.6,2)
      CALL HEADIN ('PHASE VS FREQ (F0=600 HZ)$',100,1.,2)
C .....   DEFINE THE AXES
      CALL GRAF(0.0,'SCALE',1000.,-100.,'SCALE',100.)
C..... DRAW THE CURVES
C     CALL THKCRV(0.02)
      CALL MARKER(15)
      CALL CURVE(F,HPHASE,1001,0)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
      CALL DONEPL
      STOP
      END
```

```
      DN223(1)=0.998169
      DD223(3)=1.
      DD223(2)=-1.809446
      DD223(1)=0.982544
C PRINT SECOND STAGE TRANSFER FUNCTION
      WRITE(4,82)DN223(3),DN223(2),DN223(1),DD223(3),DD223(2),DD223(1)
   82 FORMAT(' SECOND STAGE TRANSFER FUNCTION (2920 EQUIVALENT)',//,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &'   -----------------------------------------------------------',/,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',///)
C INITIALIZE VARIABLES
      PI=3.1415927
      SCALS1=0.
      SCALS2=0.
C SAMPLE PERIOD = T = 1.1521152 X 10**-4 SECONDS
      T=4.*192./6.666E6
C COMPUTE SCALE SUM FOR STAGES TO LIMIT INPUT AMPLITUDE
      DO 6 K=1,3
         SCALS1=SCALS1+ABS(DN213(K))
         SCALS1=SCALS1+ABS(DD213(K))
         SCALS2=SCALS2+ABS(DN223(K))
         SCALS2=SCALS2+ABS(DD223(K))
    6 CONTINUE
C ENSURE SCALE SUM FACTORS WILL LIMIT OUTPUT TO LESS THAN ONE
      SCALS1=SCALS1+1.E-6
      SCALS2=SCALS2+1.E-6
C COMPUTE SIMULATED INPUT MAGNITUDE LIMIT
      SINMAG=1./(SCALS1*SCALS2)
C PRINT STAGE SCALE SUMS AND INPUT MAGNITUDE LIMIT
      WRITE(4,85)SCALS1,SCALS2,SINMAG
   85 FORMAT(//,' SCALE FACTORS AND INPUT MAGNITUDE LIMIT',//,
     &' FIRST STAGE SCALE SUM = ',E14.6,/,
     &' SECOND STAGE SCALE SUM = ',E14.6,/,
     &' INPUT AMPLITUDE LIMITED TO +/- ',E14.6,///)
      WRITE(4,86)
   86 FORMAT('1')
C BEGIN SIMULATION FOR SPECIFIED FREQUENCIES GIVEN BY F(L)
C*******************************
C ADJUST AS NECESSARY
      DO 20 L=1,6
C*******************************
C COMPUTE SIMULATION RUN INPUT CONSTANT FOR EACH FREQUENCY
         TWOPIF=2.*PI*F(L)
C INITIALIZE STAGE INPUTS AND OUTPUTS
         IMOUT=0
         MOUT=0.
         DO 5 I=1,3
            X1(I)=0.
            X2(I)=0.
            Y1(I)=0.
            Y2(I)=0.
    5    CONTINUE
C PRINT SIMULATION HEADINGS
         WRITE(4,98)F(L)
   98    FORMAT(' FILTER FREQUENCY RESPONSE FOR F = ',F5.0,' HZ',/)
```

```
C**********************************************************************
C                                                                     *
C                          APPENDIX G                                 *
C                                                                     *
C                     FORTRAN PROGRAM S22F                            *
C                                                                     *
C                                                                     *
C      THIS PROGRAM SIMULATES THE EXECUTION OF A 2920 PROGRAM         *
C      FOR A 600 HZ BANDPASS FILTER TRANSFER FUNCTION WHICH IS THE    *
C      PRODUCT OF TWO SECOND ORDER FILTER SECTIONS                    *
C                                                                     *
C               Y(Z)   DN213(Z)   DN223(Z)                            *
C        H(Z) = ---- = -------- X --------                            *
C               X(Z)   DD213(Z)   DD223(Z)                            *
C                                                                     *
C      THE SIMULATION IS PERFORMED OVER A RANGE OF DIFFERENT INPUT    *
C      FREQUENCIES ABOUT THE TARGET CENTER FREQUENCY OF 600 HZ        *
C                                                                     *
C**********************************************************************
C VARIABLE DECLARATIONS
      INTEGER IMOUT
      REAL TX,INO,OUTO,F(9)
      REAL X1(3),X2(3),Y1(3),Y2(3)
      REAL SCALS1,SCALS2,SINMAG,MOUT
      REAL DN213(3),DD213(3),DN223(3),DD223(3),T,PI,TWOPIF
C INPUT FREQUENCIES
      F(1)=500.
      F(2)=575.
      F(3)=590.825
      F(4)=600.
      F(5)=625.
      F(6)=700.
C     F(7)=
C     F(8)=
C     F(9)=
      WRITE(4,80)
   80 FORMAT(' PROGRAM S22F OUTPUT',//,
     &' FOURTH ORDER FILTER FREQUENCY RESPONSE',/,
     &'   (CASCADED SECOND ORDER SECTIONS)',///)
C FIRST SECOND ORDER STAGE COEFFICIENTS
      DN213(3)=1.
      DN213(2)=-1.749875
      DN213(1)=1.001585
      DD213(3)=1.
      DD213(2)=-1.81555
      DD213(1)=0.990601
C PRINT FIRST STAGE TRANSFER FUNCTION
      WRITE(4,81)DN213(3),DN213(2),DN213(1),DD213(3),DD213(2),DD213(1)
   81 FORMAT(' FIRST STAGE TRANSFER FUNCTION (2920 EQUIVALENT)',//,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &'    -----------------------------------------------------------',/,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',///)
C SECOND SECOND ORDER STAGE COEFFICIENTS
      DN223(3)=1.
      DN223(2)=-1.877805
```

FILE: S221G   FORTRAN A1

```
      CALL ENDPL(0)                                                     S2202210
      CALL DONEPL                                                       S2202220
      STOP                                                              S2202230
      END                                                               S2202240
```

```
      CALL CROSS
      CALL GRAF(0.,100.,1100.,0.0,0.5,2.5)
C..... DRAW THE BAR CURVES
      CALL BARPAT(16)
      CALL BARWID(0.02)
      CALL VBARS(XKHZ,Z,YMAG,128)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
C----------------------------------------------------------------------
C GRAPHICS PARAMETERS FOR YPHASE VS K
C----------------------------------------------------------------------
      CALL LRGBUF
C     CALL TEK618
      CALL COMPRS
C .....     SETUP THE PLOTTING AREA
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....   LABEL THE X & Y AXES
      CALL XNAME('FREQUENCY (HZ) $',100)
      CALL YNAME('PHASE (RADS)$',100)
      CALL HEADIN ('DFT OF DIGITAL FILTER IMPULSE RESPONSE$',100,1.6,2)
      CALL HEADIN ('PHASE VS FREQUENCY$',100,1.,2)
C .....    DEFINE THE AXES
      CALL CROSS
      CALL GRAF(0.,100.,1100.,-2.0,0.5,2.0)
C..... DRAW THE PHASE CURVE
      CALL THKCRV(0.01)
      CALL MARKER(15)
      CALL CURVE(XKHZ,YPH,128,0)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
C----------------------------------------------------------------------
C GRAPHICS PARAMETERS FOR IMPULSE RESPONSE VS N
C----------------------------------------------------------------------
      CALL LRGBUF
C     CALL TEK618
      CALL COMPRS
C .....     SETUP THE PLOTTING AREA
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....   LABEL THE X & Y AXES
      CALL XNAME('ITERATION (N) $',100)
      CALL YNAME('MAGNITUDE$',100)
      CALL HEADIN ('DIGITAL FILTER IMPULSE RESPONSE$',100,1.6,3)
      CALL HEADIN ('OUTPUT VS ITERATION $',100,1.,3)
      CALL HEADIN ('INPUT = .017 (SINMAG) AT T = 0$',100,1.,3)
C .....    DEFINE THE AXES
      CALL GRAF(0.,64.,1024.,-.03,.01,.03)
C..... DRAW THE IMPULSE RESPONSE CURVE
      CALL THKCRV(0.01)
      CALL MARKER(15)
      CALL CURVE(XK,XN,1024,0)
C..... TERMINATE THIS PLOT
```

```
C PRINT PARAMETERS FOR EACH SIMULATION ITERATION
C************************************
            IM1=I-1
            WRITE(4,100)IM1,TX,X1(1),Y2(1)
  100       FORMAT(1X,I4,2X,3(F13.6,2X))
C USE THIS OUTPUT FORMAT FOR EASYPLOT ROUTINE
C           WRITE(4,100)TX,X1(1),Y2(1)
C 100       FORMAT(1X,3(F15.8,2X))
C************************************
C PERFORM SIMULATION SHIFT DELAY
            DO 15 J=2,3,1
               Y1(5-J)=Y1(4-J)
               X1(5-J)=X1(4-J)
               Y2(5-J)=Y2(4-J)
               X2(5-J)=X2(4-J)
   15       CONTINUE
   10    CONTINUE
   20 CONTINUE
C     THIS PROGRAM CALCULATES THE DFT OF THE IMPULSE RESPONSE
C     OVER 1024 VALUES.  THIS IMPLIES M = 9 (2**9=1024).  IWK IS AN
C     INTEGER VECTOR FOR FFT2C CALCULATION OF LENGTH M+1 = 10
      M=10
      DO 30 I=1,1024
         A(I)=CONJG(A(I))
   30 CONTINUE
      CALL FFT2C(A,M,IWK)
      WRITE(4,38)
   38 FORMAT(///,1X,'DFT IMPULSE RESPONSE OUTPUT OVER 1024 ITERATIONS'
     &,/,'        K              YMAG                   YPHASE',/)
      DO 40 I=1,1024
         Z(I-1)=0.
         A(I)=CONJG(A(I))
         XK(I)=FLOAT(I-1)
         XKHZ(I)=(1./T)*(XK(I)/1024.)
         YMAG(I)=CABS(A(I))
         YPH(I)=ATAN(AIMAG(A(I))/REAL(A(I)))
         WRITE(4,39)XK(I),YMAG(I),YPH(I)
   39    FORMAT(1X,F8.0,4X,F12.4,4X,F12.4)
   40 CONTINUE
C-----------------------------------------------------------------------
C GRAPHICS PARAMETERS FOR YMAG VS K
C-----------------------------------------------------------------------
      CALL LRGBUF
C     CALL TEK618
      CALL COMPRS
C .....     SETUP THE PLOTTING AREA
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....  LABEL THE X & Y AXES
      CALL XNAME('FREQUENCY (HZ)$',100)
      CALL YNAME('MAGNITUDE$',100)
      CALL HEADIN ('DFT OF DIGITAL FILTER IMPULSE RESPONSE$',100,1.6,2)
      CALL HEADIN ('MAGNITUDE VS FREQUENCY$',100,1.,2)
C .....   DEFINE THE AXES
```

```
      SCALS1=0.
      SCALS2=0.
C SAMPLE PERIOD = T = 1.1521152 X 10**-4 SECONDS
      T=4.*192./6.666E6
C COMPUTE SCALE SUM FOR STAGES TO LIMIT INPUT AMPLITUDE
      DO 6 K=1,3
         SCALS1=SCALS1+ABS(DN213(K))
         SCALS1=SCALS1+ABS(DD213(K))
         SCALS2=SCALS2+ABS(DN223(K))
         SCALS2=SCALS2+ABS(DD223(K))
    6 CONTINUE
C ENSURE SCALE SUM FACTORS WILL LIMIT OUTPUT TO LESS THAN ONE
      SCALS1=SCALS1+1.E-6
      SCALS2=SCALS2+1.E-6
C COMPUTE SIMULATED INPUT MAGNITUDE LIMIT
      SINMAG=1./(SCALS1*SCALS2)
C PRINT STAGE SCALE SUMS AND INPUT MAGNITUDE LIMIT
      WRITE(4,85)SCALS1,SCALS2,SINMAG
   85 FORMAT(//,'SCALE FACTORS AND INPUT MAGNITUDE LIMIT',//,
     &'FIRST STAGE SCALE SUM = ',E14.6,/,
     &'SECOND STAGE SCALE SUM = ',E14.6,/,
     &'INPUT AMPLITUDE LIMITED TO +/- ',E14.6,///)
C PERFORM IMPULSE RESPONSE SIMULATION
C INITIALIZE STAGE INPUTS AND OUTPUTS
      DO 5 I=1,3
         X1(I)=0.
         X2(I)=0.
         Y1(I)=0.
         Y2(I)=0.
    5 CONTINUE
C PRINT SIMULATION HEADINGS
         WRITE(4,98)
   98    FORMAT(///,'FILTER IMPULSE RESPONSE',//)
         WRITE(4,99)
   99    FORMAT(/,'    I      TIME            IN1               OUT2',//)
C COMPUTE SIMULATED FILTER RESPONSE OVER INDICATED NUMBER OF SAMPLES (I)
C******************************
C ADJUST AS NECESSARY
         DO 10 I=1,1024
C******************************
C IN1 = X1(1) = IMPULSE INPUT AT T=0.
            IF (I.EQ.1) X1(1)=SINMAG
            IF (I.NE.1) X1(1)=0.
C TX = TOTAL ELAPSED SAMPLE TIME
            TX=T*FLOAT(I-1)
C OUT1 = Y1(1) = FIRST STAGE OUTPUT = IN2 = X2(1) = SECOND STAGE INPUT
            Y1(1)=DN213(3)*X1(1)+DN213(2)*X1(2)+DN213(1)*X1(3)-
     &      DD213(2)*Y1(2)-DD213(1)*Y1(3)
            X2(1)=Y1(1)
C OUT2 = Y2(1) = SECOND STAGE OUTPUT = FILTER OUTPUT
            Y2(1)=DN223(3)*X2(1)+DN223(2)*X2(2)+DN223(1)*X2(3)-
     &      DD223(2)*Y2(2)-DD223(1)*Y2(3)
            XN(I)=Y2(1)
C FORM COMPLEX ARRAY OF IMPULSE VALUES FOR DFT
            A(I)=CMPLX(Y2(1),0.)
```

FILE: S221G   FORTRAN. A1

```
C*********************************************************************
C                                                                    *
C      THIS PROGRAM SIMULATES THE EXECUTION OF A 2920 PROGRAM        *
C      FOR A 600 HZ BANDPASS FILTER TRANSFER FUNCTION WHICH IS THE   *
C      PRODUCT OF TWO SECOND ORDER FILTER SECTIONS                   *
C                                                                    *
C              Y(Z)   DN213(Z)   DN223(Z)                            *
C        H(Z) = ---- = -------- X --------                           *
C              X(Z)   DD213(Z)   DD223(Z)                            *
C                                                                    *
C      THE SIMULATION IS PERFORMED FOR AN IMPULSE INPUT EQUAL TO THE *
C      GREATEST ALLOWABLE INPUT (SINMAG) AT T=0.  WE THEN EXAMINE    *
C      THE FAST FOURIER TRANSFORM OF THE IMPULSE SEQUENCE OVER 1024  *
C      SAMPLES TO CONFIRM THE FREQUENCY RESPONSE OF THE SYSTEM.      *
C                                                                    *
C*********************************************************************
C VARIABLE DECLARATIONS
      INTEGER IM1,M,IWK(11)
      REAL X1(3),X2(3),Y1(3),Y2(3)
      REAL SCALS1,SCALS2,SINMAG
      REAL DN213(3),DD213(3),DN223(3),DD223(3),T,TX
      COMPLEX A(1024)
      REAL XK(1024),XKHZ(1024),XN(1024),YMAG(1024),YPH(1024),Z(1024)
C PRINT OUTPUT HEADING
      WRITE(4,80)
   80 FORMAT('EIGHTH ORDER FILTER IMPULSE RESPONSE',/,
     &'    (CASCADED SECOND ORDER SECTIONS)',/,
     &'(SECOND FOURTH ORDER BLOCK OF EIGHTH ORDER FILTER)',//)
C FIRST SECOND ORDER STAGE COEFFICIENTS
      DN213(3)=1.
      DN213(2)=-1.749875
      DN213(1)=1.001585
      DD213(3)=1.
      DD213(2)=-1.81555
      DD213(1)=0.990601
C PRINT FIRST STAGE TRANSFER FUNCTION
      WRITE(4,81)DN213(3),DN213(2),DN213(1),DD213(3),DD213(2),DD213(1)
   81 FORMAT('SECOND FIRST STAGE TRANSFER FUNCTION',//,
     &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &' ------------------------------------------------------------',/,
     &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',///)
C SECOND SECOND ORDER STAGE COEFFICIENTS
      DN223(3)=1.
      DN223(2)=-1.877805
      DN223(1)=0.998169
      DD223(3)=1.
      DD223(2)=-1.809446
      DD223(1)=0.982544
C PRINT SECOND STAGE TRANSFER FUNCTION
      WRITE(4,82)DN223(3),DN223(2),DN223(1),DD223(3),DD223(2),DD223(1)
   82 FORMAT('SECOND SECOND STAGE TRANSFER FUNCTION',//,
     &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &' ------------------------------------------------------------',/,
     &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',///)
C INITIALIZE VARIABLES
```

S221 PROGRAM OUTPUT

FOURTH ORDER FILTER IMPULSE RESPONSE
(CASCADED SECOND ORDER SECTIONS)

FIRST STAGE TRANSFER FUNCTION (2920 COEFFICIENTS)

```
0.100000E+01 +  -0.174988E+01 Z**-1 +   0.100159E+01 Z**-2
-----------------------------------------------------------
0.100000E+01 +  -0.181555E+01 Z**-1 +   0.990601E+00 Z**-2
```

SECOND STAGE TRANSFER FUNCTION (2920 COEFFICIENTS)

```
0.100000E+01 +  -0.187780E+01 Z**-1 +   0.998169E+00 Z**-2
-----------------------------------------------------------
0.100000E+01 +  -0.180945E+01 Z**-1 +   0.982554E+00 Z**-2
```

SCALE FACTORS AND INPUT MAGNITUDE LIMIT

FIRST STAGE SCALE SUM =   0.755761E+01
SECOND STAGE SCALE SUM =   0.766797E+01
INPUT AMPLITUDE LIMITED TO +/-   0.172558E-01

FILTER IMPULSE RESPONSE

```
MAX AMPLITUDE OF IMPULSE RESPONSE OVER   512 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  1024 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  1536 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  2048 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  2560 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  3072 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  3584 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  4096 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  4608 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  5120 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  5632 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  6144 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  6656 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  7168 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  7680 ITERATIONS =  0.011935
MAX AMPLITUDE OF IMPULSE RESPONSE OVER  8192 ITERATIONS =  0.011935
```

MAX AMPLITUDE OCCURRED AT ITERATION   150

```
     &   D2(2,2)*Y2(2)-D2(2,1)*Y2(3)
         IF ((MXAMP.LT.(ABS(Y2(1)))).AND.(I.NE.1)) GO TO 30
            GO TO 40
   30       MXAMP=ABS(Y2(1))
            IMXAMP=I
   40       CONTINUE
C PRINT PARAMETERS FOR EACH SIMULATION ITERATION
C*************************************
C        WRITE(4,100)IM1,TX,X1(1),Y2(1)
C 100    FORMAT(1X,I4,2X,3(F13.6,2X))
C USE THIS OUTPUT FORMAT FOR EASYPLOT ROUTINE
C        WRITE(4,100)TX,X1(1),Y2(1)
C 100    FORMAT(1X,3(F15.8,2X))
C*************************************
C PERFORM SIMULATION SHIFT DELAY
         DO 15 J=2,3,1
            Y1(5-J)=Y1(4-J)
            X1(5-J)=X1(4-J)
            Y2(5-J)=Y2(4-J)
            X2(5-J)=X2(4-J)
   15    CONTINUE
         IF (MOD(I,512))10,14,10
   14    WRITE(4,16)I,MXAMP
   16    FORMAT(' MAX AMPLITUDE OF IMPULSE RESPONSE OVER ',I5,
     &' ITERATIONS = ',F9.6)
   10 CONTINUE
      WRITE(4,18)IMXAMP
   18 FORMAT(/,' MAX AMPLITUDE OCCURRED AT ITERATION ',I5)
      STOP
      END
```

```
   82 FORMAT(' SECOND STAGE TRANSFER FUNCTION (2920 COEFFICIENTS)',//,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &'   -----------------------------------------------------------',/,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',///)
C INITIALIZE VARIABLES
      SCALS1=0.
      SCALS2=0.
C SAMPLE PERIOD = T = 1.1521152 X 10**-4 SECONDS
      T=4.*192./6.666E6
C COMPUTE SCALE SUM FOR STAGES TO LIMIT INPUT AMPLITUDE
      DO 6 K=1,3
         SCALS1=SCALS1+ABS(N2(1,K))
         SCALS1=SCALS1+ABS(D2(1,K))
         SCALS2=SCALS2+ABS(N2(2,K))
         SCALS2=SCALS2+ABS(D2(2,K)).
    6 CONTINUE
C ENSURE SCALE SUM FACTORS WILL LIMIT OUTPUT TO LESS THAN ONE
      SCALS1=SCALS1+1.E-6
      SCALS2=SCALS2+1.E-6
C COMPUTE SIMULATED INPUT MAGNITUDE LIMIT
      SINMAG=1./(SCALS1*SCALS2)
C PRINT STAGE SCALE SUMS AND INPUT MAGNITUDE LIMIT
      WRITE(4,85)SCALS1,SCALS2,SINMAG
   85 FORMAT(//,' SCALE FACTORS AND INPUT MAGNITUDE LIMIT',///,
     &' FIRST STAGE SCALE SUM = ',E14.6,/,
     &' SECOND STAGE SCALE SUM = ',E14.6,/,
     &' INPUT AMPLITUDE LIMITED TO +/- ',E14.6,////)
C PERFORM IMPULSE RESPONSE SIMULATION
C INITIALIZE STAGE INPUTS AND OUTPUTS
      DO 5 I=1,3
         X1(I)=0.
         X2(I)=0.
         Y1(I)=0.
         Y2(I)=0.
    5 CONTINUE
      MXAMP=0.
      IMXAMP=0
C PRINT SIMULATION HEADINGS
      WRITE(4,98)
   98 FORMAT(' FILTER IMPULSE RESPONSE',//)
C     WRITE(4,99)
C  99 FORMAT(/,'    I      TIME              IN1              OUT2',//)
C COMPUTE SIMULATED FILTER RESPONSE OVER REQUIRED ITERATIONS (NUMIT)
      DO 10 I=1,NUMIT
C IN1 = X1(1) = IMPULSE INPUT AT T=0.
         IF (I.EQ.1) X1(1)=SINMAG
         IF (I.NE.1) X1(1)=0.
C TX = TOTAL ELAPSED SAMPLE TIME
         TX=T*FLOAT(I-1)
C OUT1 = Y1(1) = FIRST STAGE OUTPUT = IN2 = X2(1) = SECOND STAGE INPUT
         Y1(1)=N2(1,3)*X1(1)+N2(1,2)*X1(2)+N2(1,1)*X1(3)-
     &   D2(1,2)*Y1(2)-D2(1,1)*Y1(3)
         X2(1)=Y1(1)
C OUT2 = Y2(1) = SECOND STAGE OUTPUT = XN(I)
         Y2(1)=N2(2,3)*X2(1)+N2(2,2)*X2(2)+N2(2,1)*X2(3)-
```

FILE: S221   FORTRAN A1

```
C**********************************************************************
C                                                                     *
C                             APPENDIX F                              *
C                                                                     *
C                        FORTRAN PROGRAM S221                         *
C                                                                     *
C                                                                     *
C      THIS PROGRAM SIMULATES THE EXECUTION OF A 2920 PROGRAM         *
C      FOR A FOURTH ORDER ELLIPTIC 600 HZ BANDPASS FILTER TRANSFER    *
C      TRANSFER FUNCTION WHICH IS THE PRODUCT OF TWO SECOND ORDER     *
C      FILTER SECTIONS                                                *
C                                                                     *
C                        Y(Z)    N2(1,Z)    N2(2,Z)                   *
C                H(Z) = ---- = ------- X -------                      *
C                        X(Z)    D2(1,Z)    D2(2,Z)                   *
C                                                                     *
C      THE SIMULATION IS PERFORMED FOR AN IMPULSE INPUT EQUAL TO THE  *
C      GREATEST ALLOWABLE INPUT (SINMAG) AT T=0 AND ALLOWED TO RUN    *
C      OVER NUMIT ITERATIONS TO CHECK FOR STABILITY.                  *
C                                                                     *
C**********************************************************************
C VARIABLE DECLARATIONS
      INTEGER NUMIT,IMXAMP
      REAL X1(3),X2(3),Y1(3),Y2(3)
      REAL SCALS1,SCALS2,SINMAG,MXAMP
      REAL N2(2,3),D2(2,3),T,TX
C DECLARE NUMBER OF SIMULATION ITERATIONS
      NUMIT=8192
C PRINT OUTPUT HEADING
      WRITE(4,80)
   80 FORMAT(' S221 PROGRAM OUTPUT',//,
     &' FOURTH ORDER FILTER IMPULSE RESPONSE',/,
     &'   (CASCADED SECOND ORDER SECTIONS)',///)
C FIRST SECOND ORDER STAGE COEFFICIENTS
      N2(1,3)=1.
      N2(1,2)=-1.749875
      N2(1,1)=1.001585
      D2(1,3)=1.
      D2(1,2)=-1.81555
      D2(1,1)=0.990601
C PRINT FIRST STAGE TRANSFER FUNCTION
      WRITE(4,81)N2(1,3),N2(1,2),N2(1,1),D2(1,3),D2(1,2),D2(1,1)
   81 FORMAT(' FIRST STAGE TRANSFER FUNCTION (2920 COEFFICIENTS)',//,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &'   --------------------------------------------------------',/,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',///)
C SECOND SECOND ORDER STAGE COEFFICIENTS
      N2(2,3)=1.
      N2(2,2)=-1.877805
      N2(2,1)=0.998169
      D2(2,3)=1.
      D2(2,2)=-1.809446
      D2(2,1)=0.982554
C PRINT SECOND STAGE TRANSFER FUNCTION
      WRITE(4,82)N2(2,3),N2(2,2),N2(2,1),D2(2,3),D2(2,2),D2(2,1)
```

```
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....  LABEL THE X & Y AXES
      CALL XNAME('FREQUENCY (HZ)$',100)
      CALL YNAME('PHASE (DEGREES)$',100)
      CALL HEADIN('DIGITAL ELLIPTIC BPF PHASE RESPONSE$',100,1.6,2)
      CALL HEADIN('PHASE (DEGREES) VS FREQ (F0=590 HZ)$',100,1.,2)
C .....   DEFINE THE AXES
      CALL GRAF(0.0,'SCALE',1200.,-100.,'SCALE',100.)
C..... DRAW THE CURVES
      CALL THKCRV(0.02)
      CALL MARKER(15)
      CALL CURVE(FREQ,HPHASE,201,0)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
      CALL DONEPL
      STOP
      END
```

FILE: DBPFR   FORTRAN A1

```
C GRAPHICS PARAMETERS FOR MAGNITUDE VS FREQUENCY (IN HZ)
C-----------------------------------------------------------------------
      CALL LRGBUF
C     CALL COMPRS
      CALL TEK618
C     CALL VRSTEC(0,0,0)
C .....     SETUP THE PLOTTING AREA
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....  LABEL THE X & Y AXES
      CALL XNAME('FREQUENCY (HZ)$',100)
      CALL YNAME('AMPLITUDES$',100)
      CALL HEADIN('DIGITAL ELLIPTIC BPF FREQUENCY RESPONSES$',100,1.6,2)
      CALL HEADIN('NORMALIZED AMPLITUDE VS FREQ (F0=590 HZ)$',100,1.,2)
C .....   DEFINE THE AXES
      CALL GRAF(0.0,'SCALE',1200.,-0.5,'SCALE',1.5)
C..... DRAW THE CURVES
      CALL THKCRV(0.02)
      CALL MARKER(15)
      CALL CURVE(FREQ,HMAGN,201,0)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
C-----------------------------------------------------------------------
C GRAPHICS PARAMETERS FOR MAGNITUDE (IN DB) VS FREQUENCY (IN HZ)
C-----------------------------------------------------------------------
      CALL LRGBUF
C     CALL COMPRS
      CALL TEK618
C     CALL VRSTEC(0,0,0)
C .....     SETUP THE PLOTTING AREA
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....  LABEL THE X & Y AXES
      CALL XNAME('FREQUENCY (HZ)$',100)
      CALL YNAME('AMPLITUDE (DB)$',100)
      CALL HEADIN('DIGITAL ELLIPTIC BPF FREQUENCY RESPONSES$',100,1.6,2)
      CALL HEADIN('AMPLITUDE (DB) VS FREQ (F0=590 HZ)$',100,1.,2)
C .....   DEFINE THE AXES
      CALL GRAF(0.0,'SCALE',1200.,20.0,'SCALE',1[illegible].0)
C..... DRAW THE CURVES
      CALL THKCRV(0.02)
      CALL MARKER(15)
      CALL CURVE(FREQ,HMAGDB,201,0)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
C-----------------------------------------------------------------------
C GRAPHICS PARAMETERS FOR PHASE VS FREQUENCY
C-----------------------------------------------------------------------
      CALL LRGBUF
      CALL COMPRS
C     CALL TEK618
C     CALL VRSTEC(0,0,0)
C .....     SETUP THE PLOTTING AREA
```

FILE: DBPFR   FORTRAN A1

```
C***********************************************************************
C                                                                      *
C                              APPENDIX E                              *
C                                                                      *
C                        FORTRAN PROGRAM DBPFR                         *
C                                                                      *
C                                                                      *
C    PROGRAM TO PLOT DIGITAL BAND-PASS FILTER FREQUENCY AND PHASE      *
C    RESPONSE OF THE ELLIPTIC FILTER TRANSFER FUNCTION                 *
C                                                                      *
C***********************************************************************
C    TYPE DECLARATIONS
      IMPLICIT REAL(A-H,O-Z),INTEGER(I-N)
      REAL OMEGA(201),HMAG(201),HPHASE(201),HMAGN(201),HMAGDB(201)
      REAL F(201),FREQ(201),FS,FSDIV2,TS
      COMPLEX Z,H
C    NORMALIZED TRANSFER FUNCTION COEFFICIENTS
      A0 = 1.
      A1 =-3.6279
      A2 = 5.2861
      A3 = A2
      A4 = A1
      B0 = 1.
      B1 =-3.6251
      B2 = 5.2586
      B3 =-3.5768
      B4 = 0.97353
C    CONSTANTS
      PI = 3.1415927
      FS = 6.666E6/(4.*192.)
      TS=1./FS
      FSDIV2 = FS/2.
C    EVALUATE MAGNITUDE AND PHASE OF H(EXP(J*OMEGA*T))
      DO 10 I = 1,201
         F(I) = FLOAT(I-1)
         FREQ(I) = 6.*F(I)
         OMEGA(I) = (2.*PI*FREQ(I)*TS)
         Z = CMPLX(COS(OMEGA(I)),SIN(OMEGA(I)))
         H = (A0+A1*Z**(-1)+A2*Z**(-2)+A3*Z**(-3)+A4*Z**(-4))/(B0+
     &B1*Z**(-1)+B2*Z**(-2)+B3*Z**(-3)+B4*Z**(-4))
         HMAG(I) = CABS(H)
         X = REAL(H)
         Y = AIMAG(H)
         HPHASE(I) = ATAN(Y/X)*180./PI
   10 CONTINUE
C    NORMALIZE MAGNITUDE
      AMAX = 0.0
      DO 20 I = 1,201
         IF(HMAG(I).GT.AMAX) AMAX = HMAG(I)
   20 CONTINUE
      DO 30 I = 1,201
         HMAGN(I) = HMAG(I)/AMAX
         HMAGDB(I) = 20.0 * ALOG10(HMAG(I))
   30 CONTINUE
C----------------------------------------------------------------------
```

```
C        WRITE(4,99)
C  99    FORMAT(/,'    I      TIME       IN1      OUT1=IN2      OUT2',//)
C COMPUTE SIMULATED FILTER RESPONSE OVER INDICATED NUMBER OF SAMPLES (I)
C******************************
C ADJUST AS NECESSARY
         DO 10 I=1,2048
C******************************
C TX = TOTAL ELAPSED SAMPLE TIME
            TX=T*FLOAT(I-1)
C IN1 = X1(1) = FILTER FIRST STAGE INPUT VALUE (LIMITED BY SINMAG)
            X1(1)=SINMAG*SIN(TWOPIF*TX)
C OUT1 = Y1(1) = FIRST STAGE OUTPUT = IN2 = X2(1) = SECOND STAGE INPUT
            Y1(1)=DN213(3)*X1(1)+DN213(2)*X1(2)+DN213(1)*X1(3)-
     &      DD213(2)*Y1(2)-DD213(1)*Y1(3)
            X2(1)=Y1(1)
C OUT2 = Y2(1) = SECOND STAGE OUTPUT = FILTER OUTPUT
            Y2(1)=DN223(3)*X2(1)+DN223(2)*X2(2)+DN223(1)*X2(3)-
     &      DD223(2)*Y2(2)-DD223(1)*Y2(3)
C PRINT PARAMETERS FOR EACH SIMULATION ITERATION
C************************************
C           WRITE(4,100)I,TX,X1(1),X2(1),Y2(1)
C 100       FORMAT(1X,I4,2X,4(F10.3,2X))
C USE THIS OUTPUT FORMAT FOR EASYPLOT ROUTINE
C           WRITE(4,100)TX,X1(1),Y2(1)
C 100       FORMAT(1X,3(F15.8,2X))
C************************************
C REMEMBER MAXIMUM AMPLITUDE IN EACH FREQUENCY SIMULATION TRIAL
            IF (ABS(Y2(1))-MOUT) 11,11,14
   14          MOUT=ABS(Y2(1))
               IMOUT=I
C PERFORM SIMULATION SHIFT DELAY
   11       DO 15 J=2,3,1
               Y1(5-J)=Y1(4-J)
               X1(5-J)=X1(4-J)
               Y2(5-J)=Y2(4-J)
               X2(5-J)=X2(4-J)
   15       CONTINUE
   10    CONTINUE
C PRINT MAXIMUM OUTPUT AMPLITUDE FOR EACH FREQUENCY SIMULATION RUN
         WRITE(4,89)MOUT,IMOUT
   89    FORMAT('    MAXIMUM OUTPUT AMPLITUDE = ',F15.8,/,
     &'    THIS OCCURRED AT SIMULATION ITERATION ',I5,///)
   20 CONTINUE
      STOP
      END
```

PROGRAM S22F OUTPUT

FOURTH ORDER FILTER FREQUENCY RESPONSE
(CASCADED SECOND ORDER SECTIONS)

FIRST STAGE TRANSFER FUNCTION (2920 EQUIVALENT)

```
0.100000E+01 +  -0.174988E+01 Z**-1 +   0.100159E+01 Z**-2
-----------------------------------------------------------
0.100000E+01 +  -0.181555E+01 Z**-1 +   0.990601E+00 Z**-2
```

SECOND STAGE TRANSFER FUNCTION (2920 EQUIVALENT)

```
0.100000E+01 +  -0.187780E+01 Z**-1 +   0.998169E+00 Z**-2
-----------------------------------------------------------
0.100000E+01 +  -0.180945E+01 Z**-1 +   0.982544E+00 Z**-2
```

SCALE FACTORS AND INPUT MAGNITUDE LIMIT

FIRST STAGE SCALE SUM =   0.755761E+01
SECOND STAGE SCALE SUM =   0.766796E+01
INPUT AMPLITUDE LIMITED TO +/-   0.172558E-01

FILTER FREQUENCY RESPONSE FOR F = 500. HZ

MAXIMUM OUTPUT AMPLITUDE = 0.10085225
THIS OCCURRED AT SIMULATION ITERATION 180

FILTER FREQUENCY RESPONSE FOR F = 575. HZ

MAXIMUM OUTPUT AMPLITUDE = 1.59343243
THIS OCCURRED AT SIMULATION ITERATION 515

FILTER FREQUENCY RESPONSE FOR F = 591. HZ

MAXIMUM OUTPUT AMPLITUDE = 1.62147427
THIS OCCURRED AT SIMULATION ITERATION 500

FILTER FREQUENCY RESPONSE FOR F = 600. HZ

MAXIMUM OUTPUT AMPLITUDE = 0.89969766
THIS OCCURRED AT SIMULATION ITERATION 284

FILTER FREQUENCY RESPONSE FOR F = 625. HZ

MAXIMUM OUTPUT AMPLITUDE = 0.30743510
THIS OCCURRED AT SIMULATION ITERATION 135

FILTER FREQUENCY RESPONSE FOR F = 700. HZ

MAXIMUM OUTPUT AMPLITUDE = 0.07697707
THIS OCCURRED AT SIMULATION ITERATION 150

FILE: S22FG   FORTRAN A1

```
C***********************************************************************
C                                                                      *
C       THIS PROGRAM SIMULATES THE EXECUTION OF A 2920 PROGRAM         *
C       FOR A 600 HZ BANDPASS FILTER TRANSFER FUNCTION WHICH IS THE    *
C       PRODUCT OF TWO SECOND ORDER FILTER SECTIONS                    *
C                                                                      *
C                    Y(Z)    DN213(Z)    DN223(Z)                      *
C           H(Z) = ---- = -------- X --------                          *
C                    X(Z)    DD213(Z)    DD223(Z)                      *
C                                                                      *
C       THE SIMULATION IS PERFORMED OVER A RANGE OF DIFFERENT INPUT    *
C       FREQUENCIES ABOUT THE TARGET CENTER FREQUENCY OF 600 HZ.       *
C       AFTER SIMULATION THE FREQUENCY RESPONSE IS PLOTTED FOR         *
C       GRAPHICAL REVIEW.                                              *
C                                                                      *
C***********************************************************************
C VARIABLE DECLARATIONS
      INTEGER IMOUT
      REAL TX,F(9)
      REAL NUM(1024)
      REAL IN(1024)
      REAL OUT(1024)
      REAL X1(3),X2(3),Y1(3),Y2(3)
      REAL SCALS1,SCALS2,SINMAG,MOUT
      REAL DN213(3),DD213(3),DN223(3),DD223(3),T,PI,TWOPIF
C INPUT FREQUENCIES
      F(1)=700
C     F(2)=
C     F(3)=
C     F(4)=
C     F(5)=
C     F(6)=
C     F(7)=
C     F(8)=
C     F(9)=
      WRITE(4,80)
   80 FORMAT('FOURTH ORDER FILTER FREQUENCY RESPONSE',/,
     &'    (CASCADED SECOND ORDER SECTIONS)',//)
C FIRST SECOND ORDER STAGE COEFFICIENTS
      DN213(3)=1.
      DN213(2)=-1.749875
      DN213(1)=1.001585
      DD213(3)=1.
      DD213(2)=-1.81555
      DD213(1)=0.990601
C PRINT FIRST STAGE TRANSFER FUNCTION
      WRITE(4,81)DN213(3),DN213(2),DN213(1),DD213(3),DD213(2),DD213(1)
   81 FORMAT('FIRST STAGE TRANSFER FUNCTION (2920 EQUIVALENT)',//,
     &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &' ------------------------------------------------------------',/,
     &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',///)
C SECOND SECOND ORDER STAGE COEFFICIENTS
      DN223(3)=1.
      DN223(2)=-1.877805
      DN223(1)=0.998169
```

```
      DD223(3)=1.
      DD223(2)=-1.809446
      DD223(1)=0.982544
C PRINT SECOND STAGE TRANSFER FUNCTION
      WRITE(4,82)DN223(3),DN223(2),DN223(1),DD223(3),DD223(2),DD223(1)
   82 FORMAT('SECOND STAGE TRANSFER FUNCTION (2920 EQUIVALENT)',//,
     &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &' ------------------------------------------------------------',/,
     &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',///)
C INITIALIZE VARIABLES
      PI=3.1415927
      SCALS1=0.
      SCALS2=0.
C SAMPLE PERIOD = T = 1.1521152 X 10**-4 SECONDS
      T=4.*192./6.666E6
C COMPUTE SCALE SUM FOR STAGES TO LIMIT INPUT AMPLITUDE
      DO 6 K=1,3
         SCALS1=SCALS1+ABS(DN213(K))
         SCALS1=SCALS1+ABS(DD213(K))
         SCALS2=SCALS2+ABS(DN223(K))
         SCALS2=SCALS2+ABS(DD223(K))
    6 CONTINUE
C ENSURE SCALE SUM FACTORS WILL LIMIT OUTPUT TO LESS THAN ONE
      SCALS1=SCALS1+1.E-6
      SCALS2=SCALS2+1.E-6
C COMPUTE SIMULATED INPUT MAGNITUDE LIMIT
      SINMAG=1./(SCALS1*SCALS2)
C PRINT STAGE SCALE SUMS AND INPUT MAGNITUDE LIMIT
      WRITE(4,85)SCALS1,SCALS2,SINMAG
   85 FORMAT(//,'SCALE FACTORS AND INPUT MAGNITUDE LIMIT',//,
     &'FIRST STAGE SCALE SUM = ',E14.6,/,
     &'SECOND STAGE SCALE SUM = ',E14.6,/,
     &'INPUT AMPLITUDE LIMITED TO +/- ',E14.6,///)
C BEGIN SIMULATION FOR SPECIFIED FREQUENCIES GIVEN BY F(L)
C*******************************
C ADJUST AS NECESSARY
      DO 20 L=1,1
C*******************************
C COMPUTE SIMULATION RUN INPUT CONSTANT FOR EACH FREQUENCY
         TWOPIF=2.*PI*F(L)
C INITIALIZE STAGE INPUTS AND OUTPUTS
         IMOUT=0
         MOUT=0.
         DO 5 I=1,3
            X1(I)=0.
            X2(I)=0.
            Y1(I)=0.
            Y2(I)=0.
    5    CONTINUE
C PRINT SIMULATION HEADINGS
         WRITE(4,98)F(L)
   98    FORMAT(///,'FILTER FREQUENCY RESPONSE FOR F = ',F5.0,' HZ',//)
         WRITE(4,99)
   99    FORMAT(/,'    I      TIME       IN1      OUT1=IN2     OUT2',//)
C COMPUTE SIMULATED FILTER RESPONSE OVER INDICATED NUMBER OF SAMPLES (I)
```

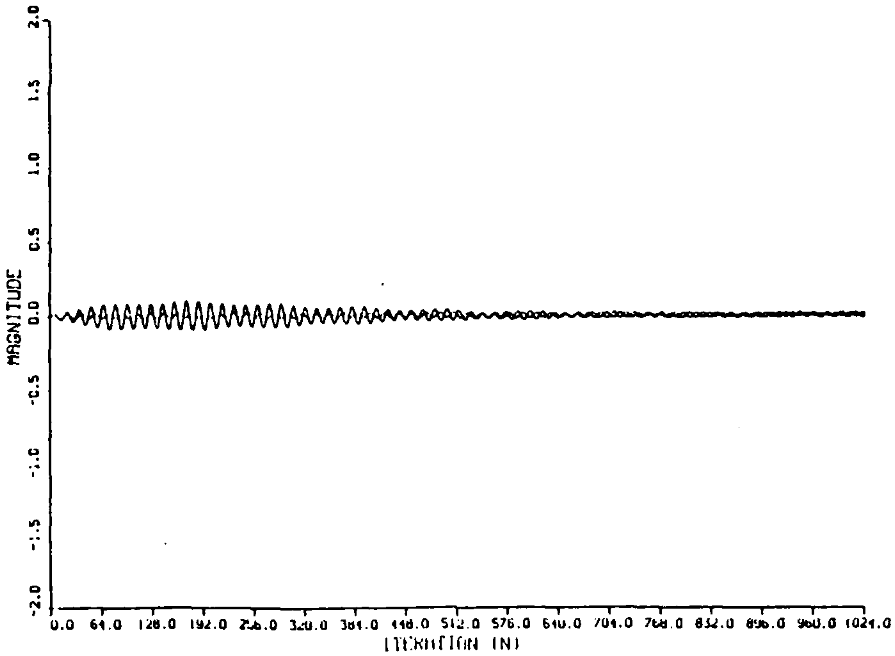
```
C*******************************
C ADJUST AS NECESSARY
         DO 10 I=1,1024
C*******************************
C TX = TOTAL ELAPSED SAMPLE TIME
            TX=T*FLOAT(I-1)
C IN1 = X1(1) = FILTER FIRST STAGE INPUT VALUE (LIMITED BY SINMAG)
            X1(1)=SINMAG*SIN(TWOPIF*TX)
            NUM(I)=FLOAT(I)
            IN(I)=X1(1)
C OUT1 = Y1(1) = FIRST STAGE OUTPUT = IN2 = X2(1) = SECOND STAGE INPUT
            Y1(1)=DN213(3)*X1(1)+DN213(2)*X1(2)+DN213(1)*X1(3)-
     &      DD213(2)*Y1(2)-DD213(1)*Y1(3)
            X2(1)=Y1(1)
C OUT2 = Y2(1) = SECOND STAGE OUTPUT = FILTER OUTPUT
            Y2(1)=DN223(3)*X2(1)+DN223(2)*X2(2)+DN223(1)*X2(3)-
     &      DD223(2)*Y2(2)-DD223(1)*Y2(3)
            OUT(I)=Y2(1)
C PRINT PARAMETERS FOR EACH SIMULATION ITERATION
C************************************
C           WRITE(4,100)I,TX,X1(1),X2(1),Y2(1)
C 100       FORMAT(1X,I4,2X,4(F10.3,2X))
C USE THIS OUTPUT FORMAT FOR EASYPLOT ROUTINE
C           WRITE(4,100)TX,X1(1),Y2(1)
C 100       FORMAT(1X,3(F15.8,2X))
C************************************
C REMEMBER MAXIMUM AMPLITUDE IN EACH FREQUENCY SIMULATION TRIAL
            IF (ABS(Y2(1))-MOUT) 11,11,14
   14          MOUT=ABS(Y2(1))
               IMOUT=I
C PERFORM SIMULATION SHIFT DELAY
   11       DO 15 J=2,3,1
               Y1(5-J)=Y1(4-J)
               X1(5-J)=X1(4-J)
               Y2(5-J)=Y2(4-J)
               X2(5-J)=X2(4-J)
   15       CONTINUE
   10    CONTINUE
C PRINT MAXIMUM OUTPUT AMPLITUDE FOR EACH FREQUENCY SIMULATION RUN
         WRITE(4,89)F(L),MOUT,IMOUT
   89    FORMAT(' MAXIMUM OUTPUT AMPLITUDE FOR ',F5.0,' HZ = ',F15.8,/,
     &' THIS OCCURRED AT SIMULATION ITERATION ',I5)
   20 CONTINUE
C---------------------------------------------------------------------
C GRAPHICS PARAMETERS FOR FREQUENCY RESPONSE OUTPUT VS INPUT
C---------------------------------------------------------------------
      CALL LRGBUF
C     CALL TEK618
      CALL COMPRS
C .....     SETUP THE PLOTTING AREA
      CALL PAGE (11.0,8.5)
      CALL NOBRDR
      CALL AREA2D(9.0,6.5)
C .....  LABEL THE X & Y AXES
      CALL XNAME('ITERATION (N) $',100)
```

```
      CALL YNAME('MAGNITUDES$',100)
      CALL HEADIN ('DIGITAL FILTER FREQUENCY RESPONSES',100,1.6,3)
      CALL HEADIN ('SIMULATION INPUT/OUTPUT VS ITERATIONS$',100,1.,3)
      CALL HEADIN ('FREQUENCY = 700 HZ$',100,1.,3)
C .....   DEFINE THE AXES
      CALL GRAF(0.,64.,1024.,-2.0,.5,2.0)
C..... DRAW THE INPUT CURVE
      CALL THKCRV(0.01)
      CALL MARKER(15)
      CALL CURVE(NUM,IN,1024,0)
C..... DRAW THE OUTPUT CURVE
      CALL THKCRV(0.01)
      CALL MARKER(15)
      CALL CURVE(NUM,OUT,1024,0)
C..... TERMINATE THIS PLOT
      CALL ENDPL(0)
      CALL DONEPL
      STOP
      END
```

DIGITAL FILTER FREQUENCY RESPONSE

SIMULATION INPUT/OUTPUT VS ITERATION

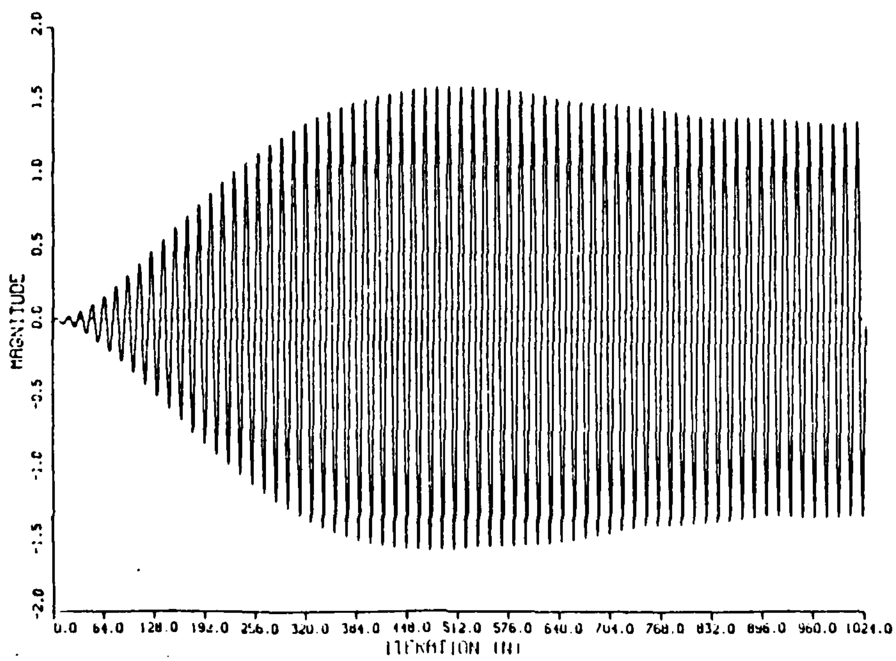
FREQUENCY - 500 HZ



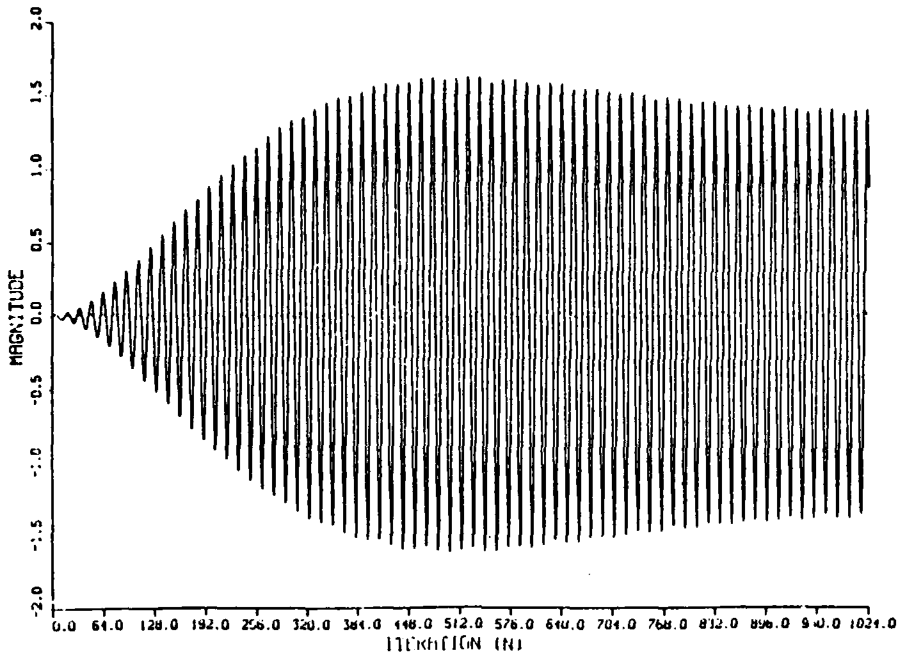
DIGITAL FILTER FREQUENCY RESPONSE

SIMULATION INPUT/OUTPUT VS ITERATION

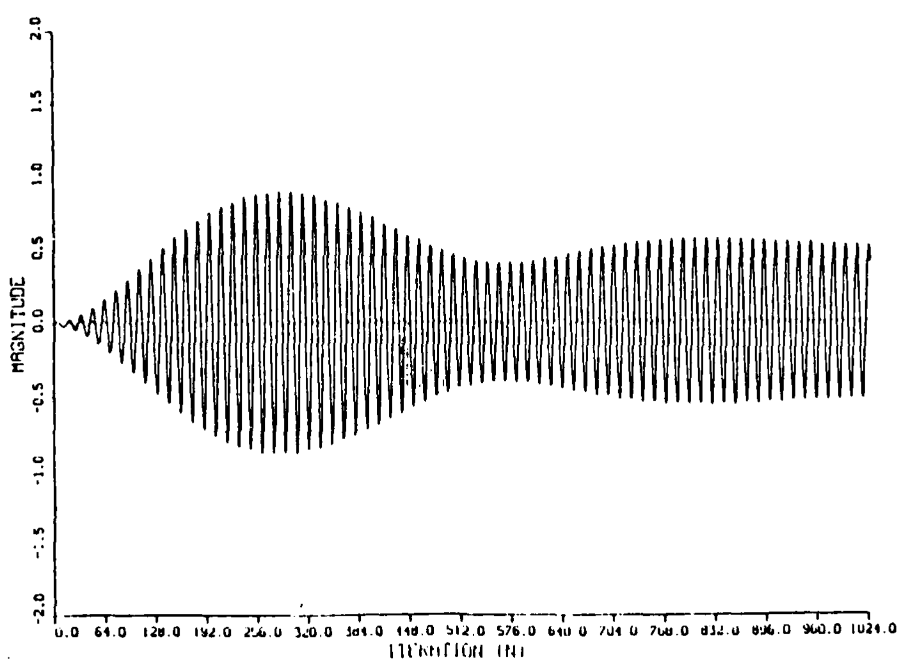
FREQUENCY - 575 HZ

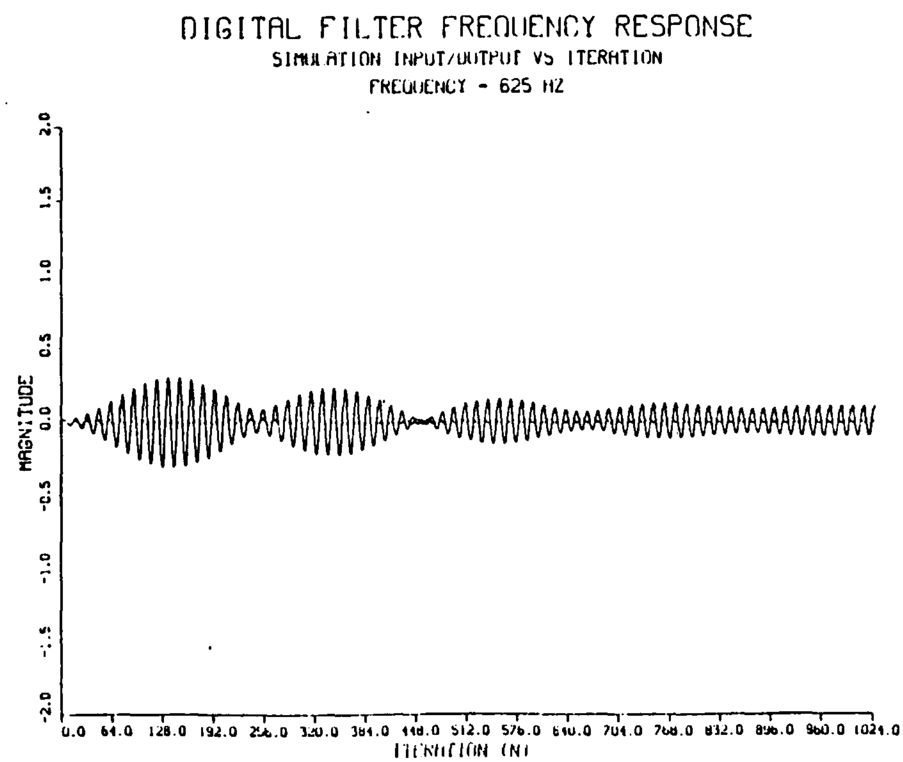


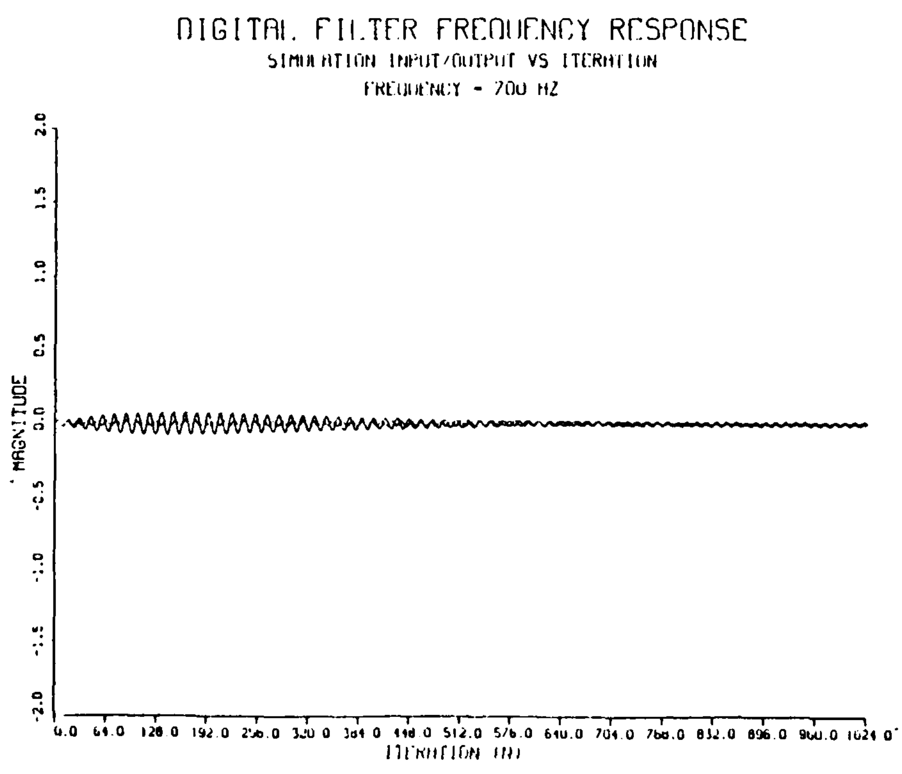
DIGITAL FILTER FREQUENCY RESPONSE
SIMULATION INPUT/OUTPUT VS ITERATION
FREQUENCY = 590.825 HZ



DIGITAL FILTER FREQUENCY RESPONSE
SIMULATION INPUT/OUTPUT VS ITERATION
FREQUENCY = 600 HZ




DIGITAL FILTER FREQUENCY RESPONSE
SIMULATION INPUT/OUTPUT VS ITERATION
FREQUENCY - 625 HZ
MAGNITUDE
ITERATION (N)


DIGITAL FILTER FREQUENCY RESPONSE
SIMULATION INPUT/OUTPUT VS ITERATION
FREQUENCY - 700 HZ
MAGNITUDE
ITERATION (N)

ASSEMBLER INVOKED BY: AS2920 SM2X4

```
LINE  LOC OBJECT SOURCE STATEMENT

                  ;
                  ;        APPENDIX H
                  ;
                  ; 2920 ASSEMBLY LANGUAGE PROGRAM SM2X2
                  ; 600 HZ CENTER FREQUENCY BAND-PASS FILTER
                  ; FOURTH ORDER ELLIPTIC FILTER
                  ; AUTHOR: LT D. W. JORDAN
                  ;
                  ; CLEAR DAR REGISTER
       0 4006EF      LDA DAR,KP0
                  ;
                  ; INPUT ANALOG SAMPLE TO SAMPLE/HOLD
                  ; INPUT MUST BE LESS THAN 1.0 VOLTS
       1 0000EF      IN0
       2 0000EF      IN0
       3 0000EF      IN0
       4 0000EF      IN0
       5 4000EF      NOP
       6 4000EF      NOP
       7 4000EF      NOP
                  ;
                  ; BEGIN ANALOG TO DIGITAL CONVERSION
                  ; DIGITAL SAMPLE WILL RESIDE IN DAR REGISTER
       8 6000EF      CVTS
       9 4000EF      NOP
      10 4000EF      NOP
      11 4000EF      NOP
      12 7100EF      CVT7
      13 4000EF      NOP
      14 4000EF      NOP
      15 4000EF      NOP
      16 6100EF      CVT6
      17 4000EF      NOP
      18 4000EF      NOP
      19 4000EF      NOP
      20 5100EF      CVT5
      21 4000EF      NOP
      22 4000EF      NOP
      23 4000EF      NOP
      24 4100EF      CVT4
      25 4000EF      NOP
      26 4000EF      NOP
      27 4000EF      NOP
      28 3100EF      CVT3
      29 4000EF      NOP
      30 4000EF      NOP
      31 4000EF      NOP
      32 2100EF      CVT2
      33 4000EF      NOP
      34 4000EF      NOP
      35 4000EF      NOP
      36 1100EF      CVT1
```

```
LINE  LOC OBJECT SOURCE STATEMENT

      37 4000EF      NOP
      38 4000EF      NOP
      39 4000EF      NOP
      40 0100EF      OUT0
      41 4000EF      NOP
      42 4000EF      NOP
      43 4000EF      NOP
               ;
               ; SCALE DOWN DIGITAL INPUT BY A FACTOR OF 64
      44 4066AE      LDA DAR,DAR,R06
               ;
               ; LOAD SCALED INPUT FROM DAR INTO X11
      45 4022EF      LDA X11,DAR
               ;
               ; INITIALIZE Y11
      46 4082FF      LDA Y11,KP0
               ;
               ; PERFORM FIRST STAGE DIFFERENCE EQUATION COMPUTATION
               ; Y11=N13*X11-N12*X12+N11*X13+D12*Y12-D11*Y13
               ;
               ; Y11=N13*X11
               ; (N13=1.000000)
               ;
      47 4400EF      LDA FRD,X11
               ;
      48 4200FD      ADD Y11,FRD
               ;
               ; Y11=Y11-N12*X12
               ; (N12=1.749875)
               ;
      49 4608EF      LDA FRD,X12,R00
      50 46080C      ADD FRD,X12,R01
      51 46084C      ADD FRD,X12,R03
      52 46086C      ADD FRD,X12,R04
      53 46088C      ADD FRD,X12,R05
      54 4608AC      ADD FRD,X12,R06
      55 4608CC      ADD FRD,X12,R07
      56 4608EC      ADD FRD,X12,R08
      57 46080D      ADD FRD,X12,R09
      58 46082D      ADD FRD,X12,R10
      59 46084D      ADD FRD,X12,R11
      60 46086D      ADD FRD,X12,R12
      61 46088D      ADD FRD,X12,R13
               ;
      62 4200FB      SUB Y11,FRD
               ;
               ; Y11=Y11+N11*X13
               ; (N11=1.001595)
               ;
      63 4000EF      LDA FRD,X13,R00
      64 40002D      ADD FRD,X13,R10
      65 40004D      ADD FRD,X13,R11
      66 40008D      ADD FRD,X13,R13
               ;
```

```
LINE  LOC OBJECT SOURCE STATEMENT

      67 4200FD     ADD Y11,FPD
                  ;
                  ; Y11=Y11+D12*Y12
                  ; (D12=1.815550)
                  ;
      68 4C08EF     LDA FPD,Y12,R00
      69 4C080C     ADD FPD,Y12,R01
      70 4C082C     ADD FPD,Y12,R02
      71 4C086C     ADD FPD,Y12,R04
      72 4C080D     ADD FPD,Y12,R09
      73 4C082D     ADD FPD,Y12,R10
      74 4C088D     ADD FPD,Y12,R13
                  ;
      75 4200FD     ADD Y11,FPD
                  ;
                  ; Y11=Y11-D11*Y13
                  ; (D11=0.990601)
                  ;
      76 4E000E     LDA FPD,Y13,R01
      77 4E002C     ADD FPD,Y13,R02
      78 4E004C     ADD FPD,Y13,R03
      79 4E006C     ADD FPD,Y13,R04
      80 4E008C     ADD FPD,Y13,R05
      81 4E00AC     ADD FPD,Y13,R06
      82 4E00EC     ADD FPD,Y13,R08
      83 4E000D     ADD FPD,Y13,R09
      84 4E006D     ADD FPD,Y13,R12
      85 4E008D     ADD FPD,Y13,R13
                  ;
      86 4200FB     SUB Y11,FPD
                  ;
                  ; INITIALIZE Y21
      87 4492FF     LDA Y21,KP0
                  ;
                  ; PERFORM SECOND STAGE DIFFERENCE EQUATION COMPUTATION
                  ; Y21=N23*Y11-N22*X22+N21*X23+D22*Y22-D21*Y23
                  ;
                  ; Y21=N23*Y11
                  ; (N23=1.000000)
                  ;
      88 4408EF     LDA FPD,Y11
                  ;
      89 4610FD     ADD Y21,FPD
                  ;
                  ; Y21=Y21-N22*X22
                  ; (N22=1.877905)
                  ;
      90 4420EF     LDA FPD,X22,R00
      91 44200C     ADD FPD,X22,R01
      92 44202C     ADD FPD,X22,R02
      93 44204C     ADD FPD,X22,R03
      94 44200D     ADD FPD,X22,R09
      95 44204D     ADD FPD,X22,R11
      96 44206D     ADD FPD,X22,R12
```

```
LINE  LOC OBJECT SOURCE STATEMENT

       97 44208D      ADD FRD,X22,R13
                  ;
       98 4610FB      SUB Y21,FRD
                  ;
                  ; Y21=Y21+N21*X23
                  ; (N21=0.998169)
                  ;
       99 44280E      LDA FRD,X23,R01
      100 44282C      ADD FRD,X23,R02
      101 44284C      ADD FRD,X23,R03
      102 44286C      ADD FRD,X23,R04
      103 44288C      ADD FRD,X23,R05
      104 4428AC      ADD FRD,X23,R06
      105 4428CC      ADD FRD,X23,R07
      106 4428EC      ADD FRD,X23,R08
      107 44280D      ADD FRD,X23,R09
      108 44288D      ADD FRD,X23,R13
                  ;
      109 4610FD      ADD Y21,FRD
                  ;
                  ; Y21=Y21-D22*Y22
                  ; (D22=1.809446)
                  ;
      110 4620EF      LDA FRD,Y22,R00
      111 46200C      ADD FRD,Y22,R01
      112 46202C      ADD FRD,Y22,R02
      113 46208C      ADD FRD,Y22,R05
      114 4620AC      ADD FRD,Y22,R06
      115 4620CC      ADD FRD,Y22,R07
      116 4620EC      ADD FRD,Y22,R08
      117 46204D      ADD FRD,Y22,R11
      118 46206D      ADD FRD,Y22,R12
      119 46208D      ADD FRD,Y22,R13
                  ;
      120 4610FD      ADD Y21,FRD
                  ;
                  ; Y21=Y21-D21*Y23
                  ; (D21=0.982544)
                  ;
      121 46280E      LDA FRD,Y23,R01
      122 46282C      ADD FRD,Y23,R02
      123 46284C      ADD FRD,Y23,R03
      124 46286C      ADD FRD,Y23,R04
      125 46288C      ADD FRD,Y23,R05
      126 4628CC      ADD FRD,Y23,R07
      127 4628EC      ADD FRD,Y23,R08
      128 46280D      ADD FRD,Y23,R09
      129 46288D      ADD FRD,Y23,R13
                  ;
      130 4610FB      SUB Y21,FRD
                  ;
                  ; LOAD Y21 INTO DAR FOR OUTPUT
      131 4A4CEF      LDA DAR,Y21,R00
                  ;
```

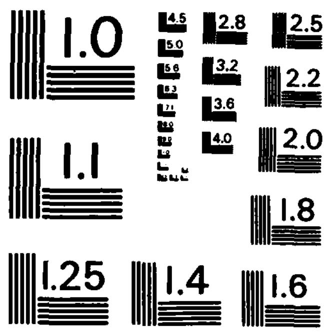
AD-A158 960 A MATCHED FILTER ALGORITHM FOR ACOUSTIC SIGNAL DETECTION(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D W JORDAN JUN 85 3/3

UNCLASSIFIED F/G 17/1 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
LINE  LOC OBJECT SOURCE STATEMENT

                 ; MULTIPLY OUTPUT BY A FACTOR OF 4
      132 4066AF    LDA DAR,DAR,L02
      133 4000EF    NOP
      134 4000EF    NOP
      135 4000EF    NOP
      136 4000EF    NOP
      137 4000EF    NOP
      138 4000EF    NOP
                 ;
                 ; OUTPUT VALUE IN DAR TO CHANNEL 0
      139 8000EF    OUT0
      140 8000EF    OUT0
      141 8000EF    OUT0
      142 8000EF    OUT0
                 ;
                 ; SERIAL REGISTER SHIFT FOR NEXT PROGRAM PASS
      143 4660FF    LDA Y23,Y22,R00
      144 4E48EF    LDA Y22,Y21,R00
      145 4C18EF    LDA Y13,Y12,R00
      146 4018FF    LDA Y12,Y11,R00
      147 4060FF    LDA X23,X22,R00
      148 4048EF    LDA X22,Y11,R00
      149 4218EF    LDA X13,X12,R00
      150 4400FF    LDA X12,X11,R00
      151 4000EF    NOP
      152 4000EF    NOP
      153 4000EF    NOP
      154 4000EF    NOP
      155 4000EF    NOP
      156 4000EF    NOP
      157 4000EF    NOP
      158 4000EF    NOP
      159 4000EF    NOP
      160 4000EF    NOP
      161 4000EF    NOP
      162 4000EF    NOP
      163 4000EF    NOP
      164 4000EF    NOP
      165 4000EF    NOP
      166 4000EF    NOP
      167 4000EF    NOP
      168 4000EF    NOP
      169 4000EF    NOP
      170 4000EF    NOP
      171 4000EF    NOP
      172 4000EF    NOP
      173 4000EF    NOP
      174 4000EF    NOP
      175 4000EF    NOP
      176 4000EF    NOP
      177 4000EF    NOP
      178 4000EF    NOP
      179 4000EF    NOP
      180 4000EF    NOP
```

```
 LINE  LOC OBJECT SOURCE STATEMENT

  269  181 4000EF      NOP
  270  182 4000EF      NOP
  271  183 4000EF      NOP
  272  184 4000EF      NOP
  273  185 4000EF      NOP
  274  186 4000EF      NOP
  275  187 4000EF      NOP
  276                ;
  277                ; THIS IS THE FINAL FOUR INSTRUCTION SEGMENT
  278  188 5000EF      EOP
  279  189 4000EF      NOP
  280  190 4000EF      NOP
  281  191 4000EF      NOP
  282                  END

SYMBOL:                          VALUE:

X11                                  0
Y11                                  1
PRD                                  2
X12                                  3
X13                                  4
Y12                                  5
Y13                                  6
Y21                                  7
X22                                  8
X23                                  9
Y22                                 10
Y23                                 11

ASSEMBLY COMPLETE
ERRORS    =     0
WARNINGS  =     0
RAMSIZE   =    12
ROMSIZE   =   192
```

```
C*********************************************************************
C                                                                    *
C                             APPENDIX I                             *
C                                                                    *
C                      FORTRAN PROGRAM CTRANS2                       *
C                                                                    *
C                                                                    *
C      THIS PROGRAM PERFORMS A TRANSFORMATION OF THE COEFFICIENTS    *
C      OF TWO SECOND ORDER POLYNOMIAL STAGES FOR 2920 IMPLEMENTATION *
C                                                                    *
C*********************************************************************
C VARIABLE DECLARATIONS
      REAL N2(2,3),D2(2,3),N2TX,D2TX,TRIAL,TWOVAL,FKM1
      REAL N2B(2,3),D2B(2,3),N2P(2,3),D2P(2,3)
      INTEGER N2T(2,3,14),D2T(2,3,14)
      INTEGER JM1,JX,KM1
C ACTUAL FIRST STAGE COEFFICIENTS TO BE TRANSFORMED
      N2(1,3)=1.
      N2(1,2)=-1.7499
      N2(1,1)=1.0017
      D2(1,3)=1.
      D2(1,2)=-1.8156
      D2(1,1)=0.99068
C ACTUAL SECOND STAGE COEFFICIENTS TO BE TRANSFORMED
      N2(2,3)=1.
      N2(2,2)=-1.8779
      N2(2,1)=0.99817
      D2(2,3)=1.
      D2(2,2)=-1.8095
      D2(2,1)=0.98255
C INITIALIZE BINARY COEFFICIENT MATRIX
      DO 10 I=1,2
         DO 12 J=1,3
            N2B(I,J)=0.
            D2B(I,J)=0.
            N2P(I,J)=0.
            D2P(I,J)=0.
            DO 14 K=1,13
               N2T(I,J,K)=0
               D2T(I,J,K)=0
   14       CONTINUE
   12    CONTINUE
   10 CONTINUE
C PERFORM COEFFICIENT TRANSFORMATION TO BINARY 2920 REPRESENTATION
C NUMERATOR TERMS
      DO 20 I=1,2
         DO 22 J=1,3
            N2TX=ABS(N2(I,J))
            IF (N2TX-1.0) 221,222,223
  222          N2T(I,J,1)=1
               N2B(I,J)=1.0
               GO TO 22
  223          N2T(I,J,1)=1
               N2B(I,J)=1.0
               N2TX=N2TX-1.
```

```
  221          DO 24 K=2,14
                  FKM1=FLOAT(K-1)
                  TWOVAL=1./(2.**FKM1)
                  TRIAL=N2TX-TWOVAL
                  IF (TRIAL) 25,242,243
  242                N2T(I,J,K)=1
                     N2B(I,J)=N2B(I,J)+TWOVAL
                     GO TO 20
  243                N2T(I,J,K)=1
                     N2B(I,J)=N2B(I,J)+TWOVAL
                     N2TX=TRIAL
                     GO TO 24
   25                IF (K.EQ.14) N2T(I,J,K)=0
   24          CONTINUE
   22    CONTINUE
   20 CONTINUE
C DENOMINATOR TERMS
      DO 30 I=1,2
         DO 32 J=1,3
            D2TX=ABS(D2(I,J))
            IF (D2TX-1.0) 321,322,323
  322          D2T(I,J,1)=1
               D2B(I,J)=1.0
               GO TO 32
  323          D2T(I,J,1)=1
               D2B(I,J)=1.0
               D2TX=D2TX-1.
  321          DO 34 K=2,14
                  FKM1=FLOAT(K-1)
                  TWOVAL=1./(2.**FKM1)
                  TRIAL=D2TX-TWOVAL
                  IF (TRIAL) 35,342,343
  342                D2T(I,J,K)=1
                     D2B(I,J)=D2B(I,J)+TWOVAL
                     GO TO 30
  343                D2T(I,J,K)=1
                     D2B(I,J)=D2B(I,J)+TWOVAL
                     D2TX=TRIAL
                     GO TO 34
   35                IF (K.EQ.14) D2T(I,J,K)=0
   34          CONTINUE
   32    CONTINUE
   30 CONTINUE
C PRINT OUTPUT HEADING
      WRITE(4,80)
   80 FORMAT(' PROGRAM CTRANS2 OUTPUT',//,
     &' FOURTH ORDER DIGITAL FILTER 2920 BINARY EQUIVALENTS',/,
     &'    (TWO CASCADED SECOND ORDER SECTIONS)',///)
C PRINT FIRST STAGE TRANSFER FUNCTION
      WRITE(4,81)N2(1,3),N2(1,2),N2(1,1),D2(1,3),D2(1,2),D2(1,1)
   81 FORMAT(' FIRST STAGE TRANSFER FUNCTION',//,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
     &'   -----------------------------------------------------------',/,
     &' ',E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2')
      WRITE(4,810)
```

```
 810 FORMAT(///,' BINARY REPRESENTATION OF NUMERATOR COEFFICIENTS',//,
    &'          ROO R01 R02 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13'
    &,/)
      DO 811 J=1,3
         JX=4-J
         N2P(1,JX)=N2B(1,JX)/ABS(N2(1,JX))
         WRITE(4,812)JX,N2(1,JX)
 812     FORMAT(' N2(1,',I1,') = ',F9.6)
         WRITE(4,814)N2T(1,JX,1),N2T(1,JX,2),N2T(1,JX,3),N2T(1,JX,4),
    &N2T(1,JX,5),N2T(1,JX,6),N2T(1,JX,7),N2T(1,JX,8),N2T(1,JX,9),
    &N2T(1,JX,10),N2T(1,JX,11),N2T(1,JX,12),N2T(1,JX,13),N2T(1,JX,14),
    &N2B(1,JX),N2P(1,JX)
 814     FORMAT(9X,14(I1,3X),/,' ABSOLUTE BINARY EQUIVALENT = '
    &,F9.6,/,' (THIS IS ',F9.6,' OF THE ACTUAL VALUE)',/)
 811 CONTINUE
      WRITE(4,815)
 815 FORMAT(//,' BINARY REPRESENTATION OF DENOMINATOR COEFFICIENTS',//,
    &'          ROO R01 R02 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13'
    &,/)
      DO 816 J=1,3
         JX=4-J
         D2P(1,JX)=D2B(1,JX)/ABS(D2(1,JX))
         WRITE(4,817)JX,D2(1,JX)
 817     FORMAT(' D2(1,',I1,') = ',F9.6)
         WRITE(4,819)D2T(1,JX,1),D2T(1,JX,2),D2T(1,JX,3),D2T(1,JX,4),
    &D2T(1,JX,5),D2T(1,JX,6),D2T(1,JX,7),D2T(1,JX,8),D2T(1,JX,9),
    &D2T(1,JX,10),D2T(1,JX,11),D2T(1,JX,12),D2T(1,JX,13),D2T(1,JX,14),
    &D2B(1,JX),D2P(1,JX)
 819     FORMAT(9X,14(I1,3X),/,' ABSOLUTE BINARY EQUIVALENT = '
    &,F9.6,/,' (THIS IS ',F9.6,' OF THE ACTUAL VALUE)',//)
 816 CONTINUE
C PRINT SECOND STAGE TRANSFER FUNCTION
      WRITE(4,82)N2(2,3),N2(2,2),N2(2,1),D2(2,3),D2(2,2),D2(2,1)
  82 FORMAT('1',' SECOND STAGE TRANSFER FUNCTION',//,
    &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2',/,
    &' ---------------------------------------------------------',/,
    &E14.6,' + ',E14.6,' Z**-1 + ',E14.6,' Z**-2')
      WRITE(4,820)
 820 FORMAT(//,' BINARY REPRESENTATION OF NUMERATOR COEFFICIENTS',//,
    &'          ROO R01 R02 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13'
    &,/)
      DO 821 J=1,3
         JX=4-J
         N2P(2,JX)=N2B(2,JX)/ABS(N2(2,JX))
         WRITE(4,822)JX,N2(1,JX)
 822     FORMAT(' N2(2,',I1,') = ',F9.6)
         WRITE(4,824)N2T(2,JX,1),N2T(2,JX,2),N2T(2,JX,3),N2T(2,JX,4),
    &N2T(2,JX,5),N2T(2,JX,6),N2T(2,JX,7),N2T(2,JX,8),N2T(2,JX,9),
    &N2T(2,JX,10),N2T(2,JX,11),N2T(2,JX,12),N2T(2,JX,13),N2T(2,JX,14),
    &N2B(2,JX),N2P(2,JX)
 824     FORMAT(9X,14(I1,3X),/,' ABSOLUTE BINARY EQUIVALENT = '
    &,F9.6,/,' (THIS IS ',F9.6,' OF THE ACTUAL VALUE)',//)
 821 CONTINUE
      WRITE(4,825)
 825 FORMAT(/,' BINARY REPRESENTATION OF DENOMINATOR COEFFICIENTS',//,
```

```
     &'         R00 R01 R02 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13'
     &,/)
      DO 826 J=1,3
         JX=4-J
         D2P(2,JX)=D2B(2,JX)/ABS(D2(2,JX))
         WRITE(4,827)JX,D2(2,JX)
 827     FORMAT(' D2(2,',I1,') = ',F9.6)
         WRITE(4,829)D2T(2,JX,1),D2T(2,JX,2),D2T(2,JX,3),D2T(2,JX,4),
     &D2T(2,JX,5),D2T(2,JX,6),D2T(2,JX,7),D2T(2,JX,8),D2T(2,JX,9),
     &D2T(2,JX,10),D2T(2,JX,11),D2T(2,JX,12),D2T(2,JX,13),D2T(2,JX,14),
     &D2B(2,JX),D2P(2,JX)
 829     FORMAT(9X,14(I1,3X),/,' ABSOLUTE BINARY EQUIVALENT = '
     &,F9.6,/,' (THIS IS ',F9.6,' OF THE ACTUAL VALUE)',//)
 826  CONTINUE
      STOP
      END
```

```
PROGRAM CTRANS2 OUTPUT

FOURTH ORDER DIGITAL FILTER 2920 BINARY EQUIVALENTS
  (TWO CASCADED SECOND ORDER SECTIONS)


FIRST STAGE TRANSFER FUNCTION

  0.100000E+01 +  -0.174990E+01 Z**-1 +   0.100170E+01 Z**-2
  ----------------------------------------------------------
  0.100000E+01 +  -0.181560E+01 Z**-1 +   0.990680E+00 Z**-2


BINARY REPRESENTATION OF NUMERATOR COEFFICIENTS

       R00 R01 R02 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13

N2(1,3) =  1.000000
        1   0   0   0   0   0   0   0   0   0   0   0   0   0
ABSOLUTE BINARY EQUIVALENT =  1.000000
(THIS IS  1.000000 OF THE ACTUAL VALUE)

N2(1,2) = -1.749900
        1   1   0   1   1   1   1   1   1   1   1   1   1   1
ABSOLUTE BINARY EQUIVALENT =  1.749875
(THIS IS  0.999986 OF THE ACTUAL VALUE)

N2(1,1) =  1.001700
        1   0   0   0   0   0   0   0   0   0   1   1   0   1
ABSOLUTE BINARY EQUIVALENT =  1.001585
(THIS IS  0.999885 OF THE ACTUAL VALUE)


BINARY REPRESENTATION OF DENOMINATOR COEFFICIENTS

       R00 R01 R02 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13

D2(1,3) =  1.000000
        1   0   0   0   0   0   0   0   0   0   0   0   0   0
ABSOLUTE BINARY EQUIVALENT =  1.000000
(THIS IS  1.000000 OF THE ACTUAL VALUE)


D2(1,2) = -1.815600
        1   1   1   0   1   0   0   0   0   1   1   0   0   1
ABSOLUTE BINARY EQUIVALENT =  1.815550
(THIS IS  0.999972 OF THE ACTUAL VALUE)


D2(1,1) =  0.990680
        0   1   1   1   1   1   1   0   1   1   0   0   1   1
ABSOLUTE BINARY EQUIVALENT =  0.990601
(THIS IS  0.999920 OF THE ACTUAL VALUE)
```

```
SECOND STAGE TRANSFER FUNCTION

0.100000E+01 +  -0.187790E+01 Z**-1 +   0.998170E+00 Z**-2
----------------------------------------------------------
0.100000E+01 +  -0.180950E+01 Z**-1 +   0.982550E+00 Z**-2

BINARY REPRESENTATION OF NUMERATOR COEFFICIENTS

       R00 R01 R02 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13

N2(2,3) =  1.000000
         1   0   0   0   0   0   0   0   0   0   0   0   0   0
ABSOLUTE BINARY EQUIVALENT =  1.000000
(THIS IS  1.000000 OF THE ACTUAL VALUE)

N2(2,2) = -1.749900
         1   1   1   1   0   0   0   0   0   1   0   1   1   1
ABSOLUTE BINARY EQUIVALENT =  1.877805
(THIS IS  0.999949 OF THE ACTUAL VALUE)

N2(2,1) =  1.001700
         0   1   1   1   1   1   1   1   1   1   0   0   0   1
ABSOLUTE BINARY EQUIVALENT =  0.998169
(THIS IS  0.999999 OF THE ACTUAL VALUE)


BINARY REPRESENTATION OF DENOMINATOR COEFFICIENTS

       R00 R01 R02 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13

D2(2,3) =  1.000000
         1   0   0   0   0   0   0   0   0   0   0   0   0   0
ABSOLUTE BINARY EQUIVALENT =  1.000000
(THIS IS  1.000000 OF THE ACTUAL VALUE)

D2(2,2) = -1.809500
         1   1   1   0   0   1   1   1   1   0   0   1   1   1
ABSOLUTE BINARY EQUIVALENT =  1.809446
(THIS IS  0.999970 OF THE ACTUAL VALUE)

D2(2,1) =  0.982550
         0   1   1   1   1   1   0   1   1   1   0   0   0   1
ABSOLUTE BINARY EQUIVALENT =  0.982544
(THIS IS  0.999994 OF THE ACTUAL VALUE)
```

FILE: FIR4P   FORTRAN A1

```
C**********************************************************************
C                                                                     *
C                              APPENDIX J                             *
C                                                                     *
C                         FORTRAN PROGRAM FIR4                        *
C                       ADAPTIVE TRANSVERSAL FILTER                   *
C                                                                     *
C  THIS PROGRAM WILL OBTAIN THE OPTIMAL FILTER WEIGHTS FOR THE FIR    *
C  FILTER OF ORDER FOUR.  THE ALGORITHM BEGINS WITH A TRIAL REGION    *
C  (-40,40) FOR EACH OF THE WEIGHTS AND DOES A SUCCESSIVE ITERATION   *
C  OF THE ERROR FUNCTION WHILE TRANSFORMING THE WEIGHTS TO OPTIMAL    *
C  VALUES.  WHEN THE WEIGHTS CONVERGE TO OPTIMAL VALUES THEN THE      *
C  ITERATION STOPS.  THEN WE MAY TEST THE ADEQUACY OF THE WEIGHTS     *
C  SO OBTAINED THROUGH SUBSEQUENT SIMULATION IN THE FORTRAN PROGRAM   *
C  FIR4SIM WHICH FOLLOWS AS APPPENDIX K.                              *
C                                                                     *
C**********************************************************************
      REAL WS(4),WU(4),WL(4),R,JE
      INTEGER NTA,NPR,NAV,NV,IP
C  WS(I) IS THE STARTING GUESS
      WS(1)=.1
      WS(2)=1.
      WS(3)=1.
      WS(4)=1.
C  WL(I) IS THE LOWER LIMIT FOR THE I'TH VARIABLE
C  WU(I) IS THE UPPER LIMIT FOR THE I'TH VARIABLE
      WL(1)=-14.
      WU(1)=14.
      WL(2)=-14.
      WU(2)=14.
      WL(3)=-14.
      WU(3)=14.
      WL(4)=-14.
      WU(4)=14.
C  A DESCRIPTION OF THE FOLLOWING PARAMETERS IS DISCUSSED IN BOXPLX
      R=9./13.
      NTA=1400
      NPR=100
      NAV=0
      NV=4
      IP=0
C  PERFORM ITERATION ROUTINE FOR WEIGHT OPTIMIZATION
      CALL BOXPLX(NV,NAV,NPR,NTA,R,WS,IP,WU,WL,YMN,IER)
      WRITE (6,25)
25    FORMAT(1X,' OPTIMAL GAINS',/)
      DO 30 I=1,4
30       WRITE(6,40)I,WS(I)
40    FORMAT(1X,'W(',I2,')=',F14.7)
      STOP
      END
C**********************************************************************
      SUBROUTINE FIR(XX)
C  SUBROUTINE FIR(XX) SIMULATES THE FIR FILTER
      COMMON J
      REAL*8 J,W0,W1,W2,W3,X1,X2,X3,INPUT,OUTPUT
```

.FILE: FIR4P   FORTRAN A1

```
      DIMENSION XX(4),DESIRE(105)
C  INITIAL CONDITIONS
      ETIME=100.
      T=0.0
      ICOUNT=2
C  INITIALIZE THE COST (CUMULATIVE ERROR) FUNCTION
      J=0.0
C  GAIN COEFFICIENTS TO BE OPTIMIZED
      W0=XX(1)
      W1=XX(2)
      W2=XX(3)
      W3=XX(4)
C  SHIFT REGISTERS
      X1=0.0
      X2=0.0
      X3=0.0
C  SIMULATE DESIRED OUTPUT SIGNAL
      DO 15 I=1,105
15       DESIRE(I)=-1.0
      DO 16 I=1,11
16       DESIRE(I+44)=1.0
C
      K=1
C  OUTPUT HEADING
C     WRITE(6,99)
C  99 FORMAT(' ','FIR TRANSVERSAL FILTER SIMULATION RESULTS',///,
C    &'  TIME        INPUT       SIMULATED OUTPUT       DESIRED OUTPUT',/)
C  LOOP FOR 100 SAMPLE ITERATIONS
  200 CONTINUE
C  SIMULATED INPUT SIGNAL
      INPUT=SIN(.1*T)*COS(.1*T)*(2.+COS(.1*T))
C     INPUT=-.0004*T**2+.04*T
C  SIMULATED OUTPUT SIGNAL FROM FIR FILTER
      OUTPUT=W0*INPUT+W1*X1+W2*X2+W3*X3
C  WHEN TO PRINTOUT
      IF (ICOUNT.EQ. 2) GO TO 50
      GO TO 300
C  PRINTOUT
   50 CONTINUE
C  EASYPLOT OUTPUT OPTION
C     WRITE (6,100) T, [illegible] DESIRE(K)
C 100 FORMAT(2X,F8.4,[illegible]F8.4,[illegible]3X,F8.4)
C  SCREEN OUTPUT OPTION
C     WRITE (6,100) T, [illegible] DESIRE(K)
C 100 FORMAT(1X,'TIME=[illegible]INPUT=',F8.4,5X,'OUTPUT=',F8.4,5X,
C    &'DESIRED OUTPUT[illegible]
      ICOUNT=1
C  TEST IF WANT TO STOP
 300  IF (T.GE.ETIME) [illegible]
C  JE=ERROR FUNCTION
      JE=(OUTPUT-DESIRE(K))**2
C  J=COST FUNCTION (CUMULATIVE ERROR)
      J=J+JE
C  STEP SIZE DELT
      DELT=1.0
```

```
      T=T+DELT
      K=K+1
      ICOUNT=ICOUNT+1
      X3=X2
      X2=X1
      X1=INPUT
      GO TO 200
C  OUTPUT OPTIMAL WEIGHTS
 400  WRITE(6,500) JE,W0,W1,W2,W3
 500  FORMAT(' ',1X,'  J =',E15.9,2X,
     1 'W0=',F15.7,2X,'W1=',F15.7,2X,'W2=',F15.7,2X,'W3=',F15.7)
      RETURN
      FND
C     ..........................................................
C
C     SUBROUTINE BOXPLX              (CATEGORY H0)
C
C     PURPOSE
C
C        BOXPLX IS A SUBROUTINE USED TO SOLVE THE PROBLEM OF LOCATING
C        A MINIMUM (OR MAXIMUM) OF AN ARBITRARY OBJECTIVE FUNCTION
C        SUBJECT TO ARBITRARY EXPLICIT AND/OR IMPLICIT CONSTRAINTS BY
C        THE COMPLEX METHOD OF M.J. BOX.  EXPLICIT CONSTRAINTS ARE
C        DEFINED AS UPPER AND LOWER BOUNDS ON THE INDEPENDENT VARIABLES.
C        IMPLICIT CONSTRAINTS MAY BE ARBITRARY FUNCTIONS OF THE VAR-
C        IABLES.  TWO FUNCTION SUBPROGRAMS TO EVALUATE THE OBJECTIVE
C        FUNCTION AND IMPLICIT CONSTRAINTS, RESPECTIVELY, MUST BE
C        SUPPLIED BY THE USER (SEE EXAMPLE BELOW).  BOXPLX ALSO HAS
C        THE OPTION TO PERFORM INTEGER PROGRAMMING, WHERE THE VALUES
C        OF THE INDEPENDENT VARIABLES ARE RESTRICTED TO INTEGERS.
C
C     USAGE
C
C        CALL BOXPLX (NV,NAV,NPR,NTA,R,XS,IP,XU,XL,YMN,IER)
C
C
C     DESCRIPTION OF PARAMETERS
C
C        NV    AN INTEGER INPUT DEFINING THE NUMBER OF INDEPENDENT
C              VARIABLES OF THE OBJECTIVE FUNCTION TO BE MINIMIZED.
C              NOTE:  MAXIMUM NV + NAV IS PRESENTLY 50.  MAXIMIM NV IS
C              25.  IF THESE LIMITS MUST BE EXCEEDED, PUNCH A SOURCE
C              DECK IN THE USUAL MANNER, AND CHANGE THE DIMENSION
C              STATEMENTS.
C
C        NAV   AN INTEGER INPUT DEFINING THE NUMBER OF AUXILIARY VAR-
C              IABLES THE USER WISHES TO DEFINE FOR HIS OWN CONVENIENCE.
C              TYPICALLY HE MAY WISH TO DEFINE THE VALUE OF EACH IMPLICIT
C              CONSTRAINT FUNCTION AS AN AUXILIARY VARIABLE.  IF THIS
C              IS DONE, THE OPTIONAL OUTPUT FEATURE OF BOXPLX CAN BE
C              USED TO OBSERVE THE VALUES OF THOSE CONSTRAINTS AS THE
C              SOLUTION PROGRESSES.  AUXILIARY VARIABLES, IF USED,
C              SHOULD BE EVALUATED IN FUNCTION KE (DEFINED BELOW).
C              NAV MAY BE ZERO.
C
```

```
C         NPR  INPUT INTEGER CONTROLLING THE FREQUENCY OF OUTPUT DESIRED
C              FOR DIAGNOSTIC PURPOSES.  IF NPR .LE. 0, NO OUTPUT WILL BE
C              PRODUCED BY BOXPLX.  OTHERWISE, THE CURRENT COMPLEX OF
C              K= 2*NV VERTICES AND THEIR CENTROID WILL BE OUTPUT AFTER
C              EACH NPR PERMISSIBLE TRIALS.  THE NUMBER OF TOTAL TRIALS,
C              NUMBER OF FEASIBLE TRIALS, NUMBER OF FUNCTION EVALUATIONS
C              AND NUMBER OF IMPLICIT CONSTRAINT EVALUATIONS ARE IN-
C              CLUDED IN THE OUTPUT.
C              ADDITIONALLY, (WHEN NPR .GT. 0) THE SAME INFORMATION
C              WILL BE OUTPUT:
C
C                1) IF THE INITIAL POINT IS NOT FEASIBLE,
C                2) AFTER THE FIRST COMPLETE COMPLEX IS GENERATED,
C                3) IF A FEASIBLE VERTEX CANNOT BE FOUND AT SOME TRIAL,
C                4) IF THE OBJECTIVE VALUE OF A VERTEX CANNOT BE MADE
C                   NO-LONGER-WORST.
C                5) IF THE LIMIT ON TRIALS (NTA) IS REACHED AND,
C                6) WHEN THE OBJECTIVE FUNCTION HAS BEEN UNCHANGED FOR
C                   2*NV TRIALS, INDICATING A LOCAL MINIMUM HAS BEEN
C                   FOUND.
C
C              IF THE USER WISHES TO TRACE THE PROGRESS OF A SOLUTION,
C              A CHOICE OF NPR = 25, 50 OR 100 IS RECOMMENDED.
C
C         NTA  INTEGER INPUT OF LIMIT ON THE NUMBER OF TRIALS ALLOWED
C              IN THE CALCULATION.  IF THE USER INPUTS NTA .LE. 0, A
C              DEFAULT VALUE OF 2000 IS USED.  WHEN THIS LIMIT IS REACHED
C              CONTROL RETURNS TO THE CALLING PROGRAM WITH THE BEST
C              ATTAINED OBJECTIVE FUNCTION VALUE IN YMN, AND THE BEST
C              ATTAINED SOLUTION POINT IN XS.
C
C         R    A REAL NUMBER INPUT TO DEFINE THE FIRST RANDOM NUMBER
C              USED IN DEVELOPING THE INITIAL COMPLEX OF 2*NV VERTICIES.
C              (0. .GT. R .LT. 1.) IF R IS NOT WITHIN THESE BOUNDS,
C              IT WILL BE REPLACED BY 1./3. .
C
C         XS   INPUT REAL ARRAY DIMENSIONED AT LEAST NV+NAV. THE FIRST
C              NV MUST CONTAIN A FEASIBLE ORIGIN FOR STARTING THE CAL-
C              CULATION.  THE LAST NAV NEED NOT BE INITIALIZED.  UPON
C              RETURN FROM BOXPLX, THE FIRST NV ELEMENTS OF THE ARRAY
C              CONTAIN THE COORDINATES OF THE MINIMUM OBJECTIVE FUNCTION,
C              AND THE REMAINING NAV (NAV .GE. 0) CONTAIN THE VALUES OF
C              THE CORRESPONDING AUXILIARY VARIABLES.
C
C         IP   INTEGER INPUT FOR OPTIONAL INTEGER PROGRAMMING.  IF IP=1,
C              THE VALUES OF THE INDEPENDENT VARIABLES WILL BE REPLACED
C              WITH INTEGER VALUES (STILL STORED AS REAL*4).
C
C         XU   A REAL ARRAY DIMENSIONED AT LEAST NV INPUTTING THE UPPER
C              BOUND ON EACH INDEPENDENT VARIABLE, (EACH EXPLICIT CON-
C              STRAINT).  INPUT VALUES ARE SLIGHTLY ALTERED BY BOXPLX.
C
C         XL   A REAL ARRAY DIMENSIONED AT LEAST NV INPUTTING THE LOWER
C              BOUND ON EACH INDEPENDENT VARIABLE, (EACH EXPLICIT CON-
C              STRAINT).  NOTE:  FOR BOTH XU AND XL CHOOSE REASONABLE
```

FILE: FIR4SIMP FORTRAN A1

```
      K=1
C  OUTPUT HEADING
      WRITE(6,99)
   99 FORMAT(' ','FIR TRANSVERSAL FILTER SIMULATION RESULTS',///,
     &'  TIME        INPUT        SIMULATED OUTPUT       DESIRED OUTPUT',/)
C  LOOP FOR 100 SAMPLE ITERATIONS
  200 CONTINUE
C  SIMULATED INPUT SIGNAL (600 HZ + 1200 HZ + 1800 HZ)
      INPUT=SIN(.1*T)*COS(.1*T)*(2.+COS(.1*T))
C  SIMULATED OUTPUT SIGNAL FROM FIR FILTER
      OUTPUT=W0*INPUT+W1*X1+W2*X2+W3*X3
C  WHEN TO PRINTOUT
      IF (ICOUNT.EQ. 2) GO TO 50
      GO TO 300
C  PRINTOUT
   50 CONTINUE
C  EASYPLOT OUTPUT OPTION
      WRITE (6,100) T,INPUT,OUTPUT,DESIRE(K)
  100 FORMAT(2X,F8.4,1X,F8.4,3X,F8.4,13X,F8.4)
C  SCREEN OUTPUT OPTION
C     WRITE (6,100) T,INPUT,OUTPUT,DESIRE(K)
C 100 FORMAT(1X,'TIME=',F7.3,5X,'INPUT=',F8.4,5X,'OUTPUT=',F8.4,5X,
C    &'DESIRED OUTPUT=',F8.4)
      ICOUNT=1
C  TEST IF WANT TO STOP
 300  IF (T.GE.ETIME) GO TO 400
C  JE=ERROR FUNCTION
      JE=(OUTPUT-DESIRE(K))**2
C  J=COST FUNCTION (CUMULATIVE ERROR)
      J=J+JE
C  STEP SIZE DELT
      DELT=1.0
      T=T+DELT
      K=K+1
      ICOUNT=ICOUNT+1
      X3=X2
      X2=X1
      X1=INPUT
      GO TO 200
  400 RETURN
      END
```

FILE: FIR4SIMP FORTRAN A1

```
C***********************************************************************
C                                                                      *
C                            APPENDIX K                                *
C                                                                      *
C                       FORTRAN PROGRAM FIR4SIM                        *
C                 ADAPTIVE TRANSVERSAL FILTER SIMULATION               *
C                                                                      *
C  TO PERFORM THE SIMULATION WE EMPLOY THE PROGRAM FIR WHICH           *
C  WAS USED BY FIR4 TO CALCULATE THE FILTER OUTPUT VALUES WHEN         *
C  CALCULATING THE OPTIMAL FILTER WEIGHTS.  WE ACCOMPLISH THIS         *
C  BY CHANGING WS(*) TO THE ACTUAL W(*) AND ALSO DELETING THE          *
C  INITIAL TRIAL BOUNDS REPRESENTED BY WU(*) AND WL(*).                *
C                                                                      *
C***********************************************************************
      REAL W(4),R,JE
C  W(I) IS THE CALCULATED OPTIMAL GAIN
      W(1)=-7.5060358
      W(2)=7.5403662
      W(3)=4.7097464
      W(4)=-5.3987589
      WRITE (6,25)
25    FORMAT(1X,' OPTIMAL GAINS',//)
      DO 30 I=1,4
30       WRITE(6,40)I,W(I)
40    FORMAT(1X,'W(',I2,')=',F14.7)
      CALL FIR(W)
      STOP
      END
C***********************************************************************
      SUBROUTINE FIR(XX)
C  SUBROUTINE FIR(XX) SIMULATES THE FIR ADAPTIVE TRANSVERSAL FILTER
      COMMON J
      REAL*8 J,W0,W1,W2,W3,X1,X2,X3,INPUT,OUTPUT
      DIMENSION XX(4),DESIRE(105)
C  INITIAL CONDITIONS
      ETIME=100.
      T=0.0
      ICOUNT=2
C  INITIALIZE THE COST (CUMULATIVE ERROR) FUNCTION
      J=0.0
C  GAIN COEFFICIENTS TO BE OPTIMIZED
      W0=XX(1)
      W1=XX(2)
      W2=XX(3)
      W3=XX(4)
C  SHIFT REGISTERS
      X1=0.0
      X2=0.0
      X3=0.0
C  SIMULATE DESIRED OUTPUT SIGNAL
      DO 15 I=1,105
15       DESIRE(I)=-1.0
      DO 16 I=1,11
16       DESIRE(I+44)=1.0
C
```

```
      KE=0                                                              FIR08260
      RETURN                                                            FIR08270
      END                                                               FIR08280
```

```
     1 '  NEW MIN IS ',E15.7)
   56 FORMAT ('OMIN OBJECTIVE FUNCTION IS ',E15.7)
      END
      SUBROUTINE FBV (K,FUN,M)
      DIMENSION FUN(50)
      M = 1
C
      DO 1 I=2,K
      IF (FUN(M).LE.FUN(I)) GO TO 1
      M = I
    1 CONTINUE
C
      RETURN
      END
      SUBROUTINE BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FN,C,IK)
      DIMENSION V(50,50), FN(50), C(25)
      WRITE (6,4) NT,NPT,NFE,NCE
C
      DO 1 I=1,K
      WRITE (6,5) FN(I),(V(J,I),J=1,NV)
      IF (NVT.LE.NV) GO TO 1
      NVP = NV+1
      WRITE (6,6) (V(J,I),J=NVP,NVT)
    1 CONTINUE
C
      IF (IK.NE.0) GO TO 2
C
      WRITE (6,7) (C(I),I=1,NV)
      RETURN
    2 IF (IK.GE.0) GO TO 3
      WRITE (6,8) (C(I),I=1,NV)
      RETURN
    3 WRITE (6,9) IK,(C(I),I=1,NV)
      RETURN
C
    4 FORMAT ('ONO. TOTAL TRIALS = ',I5,4X,'NO. FEASIBLE TRIALS = ',
     1I5,4X,'NO. FUNCTION EVALUATIONS = ',I5,4X,'NO. CONSTRAINT EVALUATI
     2ONS = ',I5/'0      FUNCTION VALUE',6X,'INDEPENDENT VARIABLES/DEPEND
     3ENT OR IMPLICIT CONSTRAINTS')
    5 FORMAT (1H ,E18.7,2X,7E14.7/(21X,7E14.7))
    6 FORMAT (21X,7E14.7)
    7 FORMAT (10HOCENTROID 11X,7E14.7/(21X,7E14.7))
    8 FORMAT ('0  BEST VERTEX',7X,7E14.7/(21X,7E14.7))
    9 FORMAT ('OCENTROID LESS VX',I2,2X,7E14.7/(21X,7E14.7))
      END
       FUNCTION FE(X)
       DIMENSION X(4)
       REAL J
       COMMON J
       CALL FIR(X)
       FE=J
       RETURN
       END
       FUNCTION KE(X)
       DIMENSION X(4)
```

```
C   IF NOT, GO TO NEW TRIAL.                                            FIR07160
   40 IF (NT.GE.NTA) GO TO 41                                           FIR07170
C                                                                       FIR07180
C  NEXT-TO-WORST VERTEX NOW BECOMES WORST.                              FIR07190
      J = JN                                                            FIR07200
      GO TO 17                                                          FIR07210
   41 IER = 3                                                           FIR07220
      IF (NPR.GT.0) WRITE (6,54)                                        FIR07230
C                                                                       FIR07240
C  COLLECTOR POINT FOR ALL ENDINGS.                                     FIR07250
C   1)  CANNOT DEVELOP FEASIBLE VERTEX.           IER = 1               FIR07260
C   2)  CANNOT DEVELOP A NO-LONGER-WORST VERTEX.  IER = 2               FIR07270
C   3)  FUNCTION VALUE UNCHANGED FOR K TRIALS.    IER = 0               FIR07280
C   4)  LIMIT ON TRIALS REACHED.                  IER = 3               FIR07290
C   5)  CANNOT FIND FEASIBLE VERTEX AT START.     IER = -1              FIR07300
   42 CONTINUE                                                          FIR07310
C                                                                       FIR07320
C  FIND BEST VERTEX.                                                    FIR07330
      CALL FBV (K,FUN,M)                                                FIR07340
      IF (IER.GE.3) GO TO 44                                            FIR07350
C                                                                       FIR07360
C  RESTART IF THIS SOLUTION IS SIGNIFICANTLY BETTER THAN THE PREVIOUS,  FIR07370
C  OR IF THIS IS THE FIRST TRY.                                         FIR07380
      IF (NPR.LE.0) GO TO 43                                            FIR07390
      WRITE (6,55) (M,YMN,FUN(M))                                       FIR07400
   43 IF (FUN(M).GE.YMN) GO TO 47                                       FIR07410
      IF (ABS(FUN(M)-YMN).LE.AMAX1(EP,EP*YMN)) GO TO 47                 FIR07420
C                                                                       FIR07430
C  GIVE IT ANOTHER TRY UNLESS LIMIT ON TRIALS REACHED.                  FIR07440
   44 YMN = FUN(M)                                                      FIR07450
      FUN(1) = FUN(M)                                                   FIR07460
C                                                                       FIR07470
      DO 45 I=1,NV                                                      FIR07480
      CEN(I) = V(I,M)                                                   FIR07490
      SUM(I) = V(I,M)                                                   FIR07500
   45 V(I,1) = V(I,M)                                                   FIR07510
C                                                                       FIR07520
      DO 46 I=1,NVT                                                     FIR07530
   46 XS(I) = V(I,M)                                                    FIR07540
C                                                                       FIR07550
      IF (IER.LT.3) GO TO 6                                             FIR07560
   47 IF (NPR.LE.0) GO TO 48                                            FIR07570
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,V(1,M),-1)               FIR07580
      WRITE (6,56) FUN(M)                                               FIR07590
   48 RETURN                                                            FIR07600
C                                                                       FIR07610
   49 FORMAT (50H0INDEX AND DIRECTION OF OUTLYING VARIABLE AT STARTI5)  FIR07620
   50 FORMAT (50H0IMPLICIT CONSTRAINT VIOLATED AT START. DEAD END. )    FIR07630
   51 FORMAT ('0CANNOT FIND FEASIBLE',I4,'TH VERTEX OR CENTROID AT STARTFIR07640
     1.')                                                               FIR07650
   52 FORMAT (10H0AT TRIAL I4,54H CANNOT FIND FEASIBLE VERTEX WHICH IS NFIR07660
     1O LONGER WORST,I4,15X,'RESTART FROM BEST VERTEX.')                FIR07670
   53 FORMAT (40H0FUNCTION HAS BEEN ALMOST UNCHANGED FOR I5,7H TRIALS)  FIR07680
   54 FORMAT (27H0LIMIT ON TRIALS EXCEEDED. )                           FIR07690
   55 FORMAT ('0BEST VERTEX IS NO.',I3,'  OLD MIN WAS ',E15.7,           FIR07700
```

```
      IF (IP.EQ.1) VT = AINT(VT+.5)
   29 V(I,J) = AMAX1(AMIN1(VT,BU(I)),BL(I))
C
      GO TO 32
C
   30 DO 31 I=1,NV
      VT = .5*(CEN(I)+V(I,J))
      IF (IP.EQ.1) VT = AINT(VT+.5)
      V(I,J) = VT
   31 CONTINUE
C
   32 IF (LIMT.LT.NLIM) GO TO 33
C
C  CANNOT MAKE THE 'J'TH VERTEX NO LONGER WORST BY DISPLACING TOWARD
C  THE CENTROID OR BY OVER-REFLECTING THRU THE BEST VERTEX.
      IER = 2
      IF (NPR .LE. 0)  GO TO 42
      WRITE (6,52)  NT, J
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,J)
      GO TO 42
   33 NT = NT+1
      GO TO 20
C
C  SUCCESS:  WE HAVE A REPLACEMENT FOR VERTEX J.
   34 FUN(J) = FUNTRY
      FUNOLD = FUNTRY
      NPT = NPT+1
C
C  EVERY 100'TH PERMISSIBLE TRIAL, RECOMPUTE CENTROID SUMMATION TO
C  AVOID CREEPING ERROR.
      IF (MOD(NPT,100).NE.0) GO TO 37
C
      DO 36 I=1,NV
      SUM(I) = 0.
C
      DO 35 N=1,K
   35 SUM(I) = SUM(I)+V(I,N)
C
      CEN(I) = SUM(I)/FK
   36 CONTINUE
C
      LC = 0
      GO TO 39
C
   37 DO 38 I=1,NV
   38 SUM(I) = SUM(I)+V(I,J)
C
      LC = J
C
   39 IF (NPR.LE.0) GO TO 40
      IF (MOD(NPT,NPR).NE.0) GO TO 40
C
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,LC)
C
C  HAS THE MAX. NUMBER OF TRIALS BEEN REACHED WITHOUT CONVERGENCE?
```

```
      GO TO 24
C
C   CONSTRAINT VIOLATION:  MOVE NEW POINT TOWARD CENTROID.
C
   22 DO 23 I=1,NV
      VT = .5*(CEN(I)+V(I,J))
      IF (IP.EQ.1) VT = AINT(VT+.5)
      V(I,J) = VT
   23 CONTINUE
C
   24 NT = NT+1
   25 CONTINUE
C
      IER = 1
C
C   CANNOT GET FEASIBLE VERTEX BY MOVING TOWARD CENTROID,
C   OR BY OVER-REFLECTING THRU THE BEST VERTEX.
      IF (NPR.LE.0) GO TO 42
      WRITE (6,52) NT,J
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,J)
      GO TO 42
C
C   FEASIBLE VERTEX FOUND, EVALUATE THE OBJECTIVE FUNCTION.
   26 NFE = NFE+1
      FUNTRY = FE(V(1,J))
C
C   TEST TO SEE IF FUNCTION VALUE HAS NOT CHANGED.
      AFO = ABS(FUNTRY-FUNOLD)
      AMX = AMAX1(ABS(EP*FUNOLD),EP)
C
C   ACTIVATE THE FOLLOWING TWO STATEMENTS FOR DIAGNOSTIC PURPOSES ONLY.
C     WRITE (6,99) J,AFO,AMX,FUNTRY,FUNOLD,FUN(J),FUN(JN),NTFS,N
C  99 FORMAT (1X,I3,6E15.7,2I5)
      IF (AFO.GT.AMX) GO TO 27
      NTFS = NTFS+1
      IF (NTFS.LT.NCT) GO TO 28
      IER = 0
      IF (NPR.LE.0) GO TO 42
      WRITE (6,53) K
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,0)
      GO TO 42
   27 NTFS = 0
C
C   IS THE NEW VERTEX NO LONGER WORST?
   28 IF (FUNTRY.LT.FUN(JN)) GO TO 34
C
C   TRIAL VERTEX IS STILL WORST; ADJUST TOWARD CENTROID.
C   EVERY 'KV'TH TIME, OVER-REFLECT THE OFFENDING VERTEX THROUGH THE
C   BEST VERTEX.
      LIMT = LIMT+1
      IF (MOD(LIMT,KV).NE.0) GO TO 30
      CALL FBV (K,FUN,M)
C
      DO 29 I=1,NV
      VT = BETA*V(I,M)-ALPHA*V(I,J)
```

```
C
C   FIND THE WORST VERTEX, THE 'J'TH.
      J = 1
C
      DO 16 I=2,K
      IF (FUN(J).GE.FUN(I)) GO TO 16
      J = I
   16 CONTINUE
C
C   BASIC LOOP.  ELIMINATE EACH WORST VERTEX IN TURN.  IT MUST BECOME
C   NO LONGER WORST, NOT MERELY IMPROVED.  FIND NEXT-TO-WORST VERTEX,
C   THE 'JN'TH ONE.
   17 JN = 1
      IF (J.EQ.1) JN = 2
C
      DO 18 I=1,K
      IF (I.EQ.J) GO TO 18
      IF (FUN(JN).GE.FUN(I)) GO TO 18
      JN = I
   18 CONTINUE
C
C   LIMT = NUMBER OF MOVES DURING THIS TRIAL TOWARD THE CENTROID
C     DUE TO FUNCTION VALUE.
      LIMT = 1
C
C   COMPUTE CENTROID AND OVER REFLECT WORST VERTEX.
C
      DO 19 I=1,NV
      VT = V(I,J)
      SUM(I) = SUM(I)-VT
      CEN(I) = SUM(I)/FKM
      VT = BETA*CEN(I)-ALPHA*VT
      IF (IP.EQ.1) VT = AINT(VT+.5)
C
C   INSURE THE EXPLICIT CONSTRAINTS ARE OBSERVED.
   19 V(I,J) = AMAX1(AMIN1(VT,BU(I)),BL(I))
C
      NT = NT+1
C
C   CHECK FOR IMPLICIT CONSTRAINT VIOLATION.
C
   20 DO 25 N=1,NLIM
      NCE = NCE+1
      IF (KE(V(1,J)).EQ.0) GO TO 26
C
C   EVERY 'KV'TH TIME, OVER-REFLECT THE OFFENDING VERTEX THROUGH THE
C   BEST VERTEX.
      IF (MOD(N,KV).NE.0) GO TO 22
      CALL FBV (K,FUN,M)
C
      DO 21 I=1,NV
      VT = BETA*V(I,M)-ALPHA*V(I,J)
      IF (IP.EQ.1) VT = AINT(VT+.5)
   21 V(I,J) = AMAX1(AMIN1(VT,BU(I)),BL(I))
C
```

```
      FUNOLD = FUN(1)
C
      DO 15 I=2,K
      FI = FI+1.
      LIMT = 0
    7 LIMT = LIMT+1
C
C   END CALCULATION IF FEASIBLE CENTROID CANNOT BE FOUND.
      IF (LIMT.GE.NLIM) GO TO 11
C
      DO 8 J=1,NV
C
C   RANDOM NUMBER GENERATOR  (RANDU)
      IQR = IQR*65539
      IF (IQR.LT.0) IQR = IQR+2147483647+1
      RQX = IQR
      RQX = RQX*.4656613E-9
      V(J,I) = BL(J)+RQX*(BU(J)-BL(J))
      IF (IP.EQ.1) V(J,I)=AINT(V(J,I)+.5)
    8 CONTINUE
C
      DO 10 L=1,NLIM
      NCE = NCE+1
      IF (KE(V(1,I)).EQ.0) GO TO 13
C
      DO 9 J=1,NV
      VT = .5*(V(J,I)+CEN(J))
      IF (IP.EQ.1) VT = AINT(VT+.5)
      V(J,I) = VT
    9 CONTINUE
C
   10 CONTINUE
C
   11 IF (NPR.LE.0) GO TO 12
      WRITE (6,51) I
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,I,FUN,CEN,I)
   12 IER = -1
      GO TO 48
C
   13 DO 14 J=1,NV
      SUM(J) = SUM(J)+V(J,I)
   14 CEN(J) = SUM(J)/FI
C
C    TRY TO ASSURE FEASIBLE CENTROID FOR STARTING.
      NCE = NCE+1
      IF (KE(CEN).EQ.0) GO TO 60
      SUM(J) = SUM(J) -V(J,I)
      GO TO 7
   60 NFE = NFE+1
      FUN(I) = FE(V(1,I))
   15 CONTINUE
C
C    END OF LOOP SETTING OF INITIAL COMPLEX.
      IF (NPR.LE.0) GO TO 17
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,0)
```

```
      NPT = 0
C           CURRENT NO. OF PERMISSIBLE TRIALS
      NTFS = 0
C           CURRENT NO. OF TIMES F HAS BEEN ALMOST UNCHANGED
C
C             CHECK FEASIBILITY OF START POINT
C
      DO 4 I=1,NV
      VT = XS(I)
      IF (BL(I).LE.VT) GO TO 1
      II = -I
      VT = BL(I)
      GO TO 2
    1 IF (BU(I).GE.VT) GO TO 3
      II = I
      VT = BU(I)
    2 IF (NPR.GT.0) WRITE (6,49) II
    3 V(I,1) = VT
      CEN(I) = VT
      IF (IP.EQ.1) GO TO 4
      BL(I) = BL(I)+AMAX1(EP,EP*ABS(BL(I)))
      BU(I) = BU(I)-AMAX1(EP,EP*ABS(BU(I)))
    4 SUM(I) = VT
C
C
      NCE = 1
C           NUMBER OF CONSTRAINT EVALUATIONS
      I = 1
      IF (KE(V(1,1)).EQ.0) GO TO 5
      IF (NPR.LE.0) GO TO 12
      WRITE (6,50)
      GO TO 12
    5 NFE = 1
C
C   NUMBER OF VERTICES (K) = 2 TIMES NO. OF VARIABLES.
      K = 2*NV
C
C   NUMBER OF DISPLACEMENTS ALLOWED.
      NLIM = 5*NV+10
C
C   NUMBER OF CONSECUTIVE TRIALS WITH UNCHANGED FE TO TERMINATE.
      NCT = NLIM+NV
      ALPHA = 1.3
      FK = K
      FKM = FK-1.
      BETA = ALPHA+1.
C
C   INSURE SEED OF RANDOM NUMBER GENERATOR IS ODD.
      IQR = R*1.E7
      IF (MOD(IQR,2).EQ.0) IQR=IQR+101
C
C                      SET UP INITIAL VERTICES
      FUN(1) = FE(V(1,1))
      YMN = FUN(1)
    6 FI = 1.
```

```
C        SUBROUTINE 'BOUT' AND FUNCTION 'FBV' ARE INTEGRAL PARTS OF
C        THE BOXPLX PACKAGE.
C
C        TWO FUNCTIONS MUST BE SUPPLIED BY THE USER.  THE FIRST, KE(X),
C        IS USED TO EVALUATE THE IMPLICIT CONSTRAINT.   SET KE=0 AT THE
C        BEGINING OF THE FUNCTION, THEN EVALUATE THE IMPLICIT CON-
C        STRAINTS.  IN THE EXAMPLE ABOVE, THE FIRST CONSTRAINT, X(3),
C        MUST BE WITHIN THE RANGE (0. .LE. X(3) .LE. 6.).  THE SECOND
C        CONSTRAINT X(4), MUST BE .GE. 0. .  IF EITHER CONSTRAINT IS
C        NOT WITHIN THESE BOUNDS, CONTROL IS TRANSFERRED TO STATEMENT 1,
C        AND KE IS SET TO "1" AND CONTROL IS RETURNED TO BOXPLX.
C
C        THE SECOND FUNCTION THE USER MUST PROVIDE EVALUATES THE OB-
C        JECTIVE FUNCTION.  IT IS CALLED FE(X) AS SHOWN IN THE EXAM-
C        PLE ABOVE, AND FE MUST BE SET TO THE VALUE OF THE OBJECTIVE
C        FUNCTION CORRESPONDING TO CURRENT VALUES OF THE NV INDEPENDENT
C        VARIABLES IN ARRAY 'X'.
C
C     REFERENCES
C
C        BOX, M. J., "A NEW METHOD OF CONSTRAINED OPTIMIZATION AND A
C        COMPARISON WITH OTHER METHODS", COMPUTER JOURNAL, 8 APR. '65,
C        PP. 45-52.
C
C        BEVERIDGE G., AND SCHECHTER R., "OPTIMIZATION: THEORY AND
C        PRACTICE", MCGRAW-HILL, 1970.
C
C     PROGRAMMER
C
C        R.R. HILLEARY 1/1966.
C        REVISED FOR SYSTEM 360 4/1967
C        CORRECTED  1/1969
C        REVISED/EXTENDED BY L.NOLAN/R.HILLEARY  2/1975
C        CORRECTED  8/1976
C
C
C     ...............................................................
C
C
      SUBROUTINE BOXPLX (NV,NAV,NPR,NTZ,RZ,XS,IP,BU,BL,YMN,IER)
C
      DIMENSION V(50,50), FUN(50), SUM(25), CEN(25), XS(NV), BU(NV), BL(
     1NV)
C
      KV = 5
      EP = 1.E-6
      NTA = 2000
      IF (NTZ.GT.0) NTA = NTZ
      R = RZ
      IF (R.LE.0..OR.R.GE.1.) R=1./3.
      NVT = NV+NAV
C
C           TOTAL VARS, EXPLICIT PLUS IMPLICIT
      NT = 0
C           CURRENT TRIAL NO.
```

```
C         THE REPLACED VERTEX AND CENTROID OF ALL OTHER VERTICES.)
C
C         WHEN AN OVER-REFLECTION IS NOT FEASIBLE OR REMAINS WORST, IT
C         IS CONSIDERED NOT-PERMISSIBLE AND IS DISPLACED HALFWAY TOWARD
C         THE CENTROID.  AFTER FOUR SUCH ATTEMPTS ARE MADE UNSUCCESSFULLY
C         EVERY FIFTH ATTEMPT IS MADE BY REFLECTING THE OFFENDING VERTEX
C         THROUGH THE PRESENT BEST VERTEX, INSTEAD OF THROUGH THE CEN-
C         TROID.  IF 5*N+10 DISPLACEMENTS AND OVER-REFLECTIONS OCCUR
C         WITHOUT A SUCCESSFUL (PERMISSIBLE) RESULT, THE CURRENT BEST
C         VERTEX IS TAKEN AS AN INITIAL FEASIBLE POINT FOR A RESTART
C         RUN OF THE COMPLETE PROCESS.  RESTARTING IS ALSO UNDERTAKEN
C         WHEN 6*NV+10 CONSECUTIVE TRIALS HAVE BEEN MADE WITH NO SIGNIF-
C         ICANT CHANGE IN THE VALUE OF THE OBJECTIVE FUNCTION.  IN ALL
C         CASES, RESTARTING IS INHIBITED IF THE LAST RESTART DID NOT
C         PRODUCE A SIGNIFICANT IMPROVEMENT IN THE MINIMUM ATTAINED.
C
C         IT IS RECOMMENDED THAT THE USER READ THE REFERENCE FOR
C         FURTHER USEFUL INFORMATION.  IT SHOULD BE NOTED THAT THE
C         ALGORITHM DEFINED THERE HAS BEEN ALTERED TO FIND THE
C         CONSTRAINED MINIMUM, RATHER THAN THE MAXIMUM.
C
C
C
C      REMARKS
C
C         THE INTEGER PROGRAMMING OPTION WAS ADDED TO THIS PROGRAM
C         AS SUGGESTED IN REFERENCE (2).  A MIXED INTEGER/CONTINUOUS
C         VARIABLE VERSION OF BOXPLX WOULD BE EASY TO CREATE BY DE-
C         CLARING "IP" TO BE AN ARRAY OF NV CONTROL VARIABLES WHERE IP
C         (I)=1 WOULD INDICATE THAT THE I-TH VARIABLE IS TO BE CONFINED
C         TO INTEGER VALUES.  EACH STATEMENT OF THE FORM 'IF (IP .EQ.
C         1)' ETC. WOULD THEN NEED TO BE ALTERED TO 'IF (IP(I) .EQ. 1)'
C         ETC., WHERE THE SUBSCRIPT IS APPROPRIATELY CHOSEN.  NORMALLY,
C         XU AND XL VALUES ARE ALTERED TO BE AN EPSILON 'WITHIN' ACTUAL
C         VALUES DECLARED BY THE USER.  THIS ADJUSTMENT IS NOT MADE
C         WHEN IP=1.
C
C         NOTE:  NO NON-LINEAR PROGRAMMING ALGORITHM CAN GUARANTEE THAT
C         THE ANSWER FOUND IS THE GLOBAL MINIMUM, RATHER THAN JUST A
C         LOCAL MINIMUM.  HOWEVER, ACCORDING TO REF.2, THE COMPLEX
C         METHOD HAS AN ADVANTAGE IN THAT IT TENDS TO FIND THE GLOBAL
C         MINIMUM MORE FREQUENTLY THAN MANY OTHER NON-LINEAR PROGRAM-
C         MING ALGORITHMS.
C
C         IT SHOULD BE NOTED THAT THE AUXILIARY VARIABLE FEATURE CAN
C         ALSO BE USED TO DEAL WITH PROBLEMS CONTAINING EQUALITY CON-
C         STRAINTS.  ANY EQUALITY CONSTRAINT IMPLIES THAT A GIVEN VAR-
C         IABLE IS NOT TRULY INDEPENDENT.  THEREFORE, IN GENERAL, ONE
C         VARIABLE INVOLVED IN AN EQUALITY CONSTRAINT CAN BE RENUMBERED
C         FROM THE SET OF NV INDEPENDENT VARIABLES AND ADDED TO THE SET
C         OF NAV AUXILIARY VARIABLES.  THIS USUALLY INVOLVES RENUMBERING
C         THE INDEPENDENT VARIABLES OF THE GIVEN PROBLEM.
C
C      SUBROUTINES AND FUNCTIONS REQUIRED
C
```

```
CC
C      FUNCTION  KE(X)
CC  EVALUATE CONSTRAINTS.  SET KE=0 IF NO IMPLICIT CONSTRAINT IS
CC  VIOLATED, OR SET KE=1 IF ANY IMPLICIT CONSTRAINT IS VIOLATED.
C      DIMENSION X(4)
C      X1 = X(1)
C      X2 = X(2)
C      KE = 0
C      X(3) = X1 + 1.732051*X2
C      IF (X(3) .LT. 0. .OR. X(3) .GT. 6.) GO TO 1
C      X(4) = X1/1.732051 -X2
C      IF (X(4) .GE. 0.)  RETURN
CC
C    1 KE = 1
C      RETURN
C      END
CC
CC
C      FUNCTION  FE(X)
C      DIMENSION X(4)
CC
CC  THIS IS THE OBJECTIVE FUNCTION.
C      FE= -(X(2)**3 *(9.-(X(1)-3.)**2)/(46.76538))
C      RETURN
C      END
C
C      METHOD
C
C         THE COMPLEX METHOD IS AN EXTENSION AND ADAPTION OF THE SIM-
C         PLEX METHOD OF LINEAR PROGRAMMING.  STARTING WITH ANY ONE
C         FEASIBLE POINT IN N-DIMENSION SPACE A "COMPLEX" OF 2*N
C         VERTICES IS CONSTRUCTED BY SELECTING RANDOM POINTS WITHIN THE
C         FEASIBLE REGION.  FOR THIS PURPOSE N COORDINATES ARE FIRST
C         RANDOMLY CHOSEN WITHIN THE SPACE BOUNDED BY EXPLICIT CON-
C         STRAINTS.  THIS DEFINES A TRIAL INITIAL VERTEX.  IT IS THEN
C         CHECKED FOR POSSIBLE VIOLATION OF IMPLICIT CONSTRAINTS.  IF
C         ONE OR MORE ARE VIOLATED, THE TRIAL INITIAL VERTEX IS DISPLACED
C         HALF OF ITS DISTANCE FROM THE CENTROID OF PREVIOUSLY SELECTED
C         INITIAL VERTICES.  IF NECESSARY THIS DISPLACEMENT PROCESS IS
C         REPEATED UNTIL THE VERTEX HAS BECOME FEASIBLE.  IF THIS FAILS
C         TO HAPPEN AFTER 5*N+10 DISPLACEMENTS, THE SOLUTION IS ABAND-
C         ONED.  AFTER EACH VERTEX IS ADDED TO THE COMPLEX, THE CURRENT
C         CENTROID IS CHECKED FOR FEASIBILITY.  IF IT IS INFEASIBLE,
C         THE LAST TRIAL VERTEX IS ABANDONED AND AN EFFORT TO GENERATE
C         AN ALTERNATIVE TRIAL VERTEX IS MADE.  IF 5*N+10 VERTICES ARE
C         ABANDONED CONSECUTIVELY, THE SOLUTION IS TERMINATED.
C
C         IF AN INITIAL COMPLEX IS ESTABLISHED, THE BASIC COMPUTATION
C         LOOP IS INITIATED.  THESE INSTRUCTIONS FIND THE CURRENT WORST
C         VERTEX, THAT IS, THE VERTEX WITH THE LARGEST CORRESPONDING
C         VALUE FOR THE OBJECTIVE FUNCTION, AND REPLACE THAT VERTEX BY
C         ITS OVER-REFLECTION THROUGH THE CENTROID OF ALL OTHER VERTICES.
C         (IF THE VERTEX TO BE REPLACED IS CONSIDERED AS A VECTOR IN
C         N-SPACE, ITS OVER-REFLECTION IS OPPOSITE IN DIRECTION, IN-
C         CREASED IN LENGTH BY THE FACTOR 1.3, AND COLLINEAR WITH
```

```
C              VALUES IF NONE ARE GIVEN, NOT VALUES WHICH ARE MAGNITUDES
C              ABOVE OR BELOW THE EXPECTED SOLUTION.  INPUT VALUES ARE
C              SLIGHTLY ALTERED BY BOXPLX.
C
C         YMN  THIS OUTPUT IS THE VALUE (REAL*4) OF THE OBJECTIVE FUNC-
C              TION, CORRESPONDING TO THE SOLUTION POINT OUTPUT IN XS.
C
C         IER  INTEGER ERROR RETURN.  TO BE INTERROGATED UPON RETURN
C              FROM BOXPLX.  IER WILL BE ONE OF THE FOLLOWING:
C
C                =-1  CANNOT FIND FEASIBLE VERTEX OR FEASIBLE CENTROID
C                     AT THE START OR A RESTART (SEE 'METHOD' BELOW).
C                =0   FUNCTION VALUE UNCHANGED FOR 'N' TRIALS.  (WHERE
C                     N=6*NV+10)  THIS IS THE NORMAL RETURN PARAMETER.
C                =1   CANNOT DEVELOP FEASIBLE VERTEX.
C                =2   CANNOT DEVELOP A NO-LONGER-WORST VERTEX.
C                =3   LIMIT ON TRIALS REACHED.  (NTA EXCEEDED)
C                NOTE:  VALID RESULTS MAY BE RETURNED IN ANY OF THE
C                     ABOVE CASES.
C
C     EXAMPLE OF USAGE
C
C        THIS EXAMPLE MINIMIZES THE OBJECTIVE FUNCTION SHOWN IN THE
C        EXTERNAL FUNCTION FE(X).  THERE ARE TWO INDEPENDENT VAR-
C        IABLES X(1) & X(2), AND TWO IMPLICIT CONSTRAINT FUNCTIONS
C        X(3) & X(4) WHICH ARE EVALUATED AS AUXILIARY VARIABLES (SEE
C        EXTERNAL FUNCTION KE(X) ).
C
C     DIMENSION  XS(4),XU(2),XL(2)
CC
CC  STARTING GUESS
C     XS(1) = 1.0
C     XS(2) = 0.5
CC  UPPER LIMITS
C     XU(1) = 6.0
C     XU(2) = 6.0
CC  LOWER LIMITS
C     XL(1) = 0.0
C     XL(2) = 0.0
CC
C     R = 9./13.
C     NTA = 5000
C     NPR = 50
C     NAV = 2
C     NV = 2
C     IP = 0
CC
C      CALL BOXPLX (NV,NAV,NPR,NTA,R,XS,IP,XU,XL,YMN,IER)
C     WRITE(6,1) ((XS(I),I=1,4),YMN,IER)
C   1 FORMAT (////,'    THE POINT IS LOCATED AT (XS(I)=) ',4(E13.7,5X),
C    1//,'    AND THE FUNCTION VALUE IS ',E13.7,' IER = ',I5)
CC
C     STOP
C     END
CC
```

LIST OF REFERENCES

1. Get Away Special (GAS) Small Self-Contained Payloads Experimenter Handbook, pp. 7-9, National Aeronautics and Space Administration, Goddard Space Flight Center, 1984.

2. AIAA/SAE Joint Propulsion Conference Report 78-1007, Space Shuttle Orbiter Auxiliary Power Unit Configuration and Performance, by J. R. Baughman and M. W. Reck, pp. 2-3, 25 July 1978.

3. NASA DATE Report 003, Payload Bay Acoustic and Vibration Data from STS-2 Flight, pp. 1-1 through 1-4 and 1-11 through 1-23, January 1982.

4. NASA DATE Report 005, Payload Bay Acoustic and Vibration Data from STS-4 Flight, pp. 1-1 through 1-4 and 1-11 through 1-23, December 1982.

5. Chen, C-T., One Dimensional Digital Signal Processing, pp. 141-326, Marcel Dekker, 1979.

6. Johnson, D. E., Johnson, J. R., and Moore, H. P., A Handbook of Active Filters, pp. 11-68, Prentice-Hall, 1980.

7. Intel Corp., The 2920 Analog Signal Processor Design Handbook, Intel, Santa Clara, 1979.

8. Gold, B., and Rabiner, L. R., Theory and Application of Digital Signal Processing, pp. 219-224, Prentice-Hall, 1975.

9. Intel Corp. AR-81, Single Chip N-MOS Microcomputer Processes Signals in Real Time, by M. E. Hoff and M. Townsend, pp. 1-8, March 1979.

10. Intel Corp., SDK-2920 System Design Kit User's Guide, Intel, Santa Clara, 1983.

11. Intel Corp., 2920 Assembly Language Manual, Intel, Santa Clara, 1979.

12. Naval Undersea Center TP 530, Principles and Applications of Adaptive Filters: A Tutorial Review, by J. M. McCool, pp. 1-11, March 1977.

13. Parker, S. R., An Integrated Approach to Discrete Signal Processing for Scientists and Engineers (Pre-publication notes), Naval Postgraduate School, Monterey, California, 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Commander, Naval Space Command Dahlgren, Virginia 22448	1
4.	Professor Rudolf Panholzer, Code 62Pz Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5100	3
5.	Professor Sydney R. Parker, Code 62Px Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5100	1
6.	Dr. Bharat Madan, Code 62Px Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5100	1
7.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
8.	LCDR Dorsett W. Jordan Naval Research Laboratory (Code 9110-52) 4555 Overlook Ave. S. W. Washington, D. C. 20375	2
9.	Mr. Donald E. Jordan 630 Steamboat Road Apartment 4E-N Greenwich, Connecticut 06830	1

10. Mr. Dorsett M. Jordan 1
2041 Shillingwood Drive
Kennesaw, Georgia 30144

11. Mrs. Victoria Soler 1
16 Weld Street
West Roxbury, Massachusetts 02131

END

FILMED

11-85

DTIC