/A NUMERICAL PROCEDURE FOR COMPUTING ERRORS IN THE MEASUREMENT OF PULSE TIME-OF-ARRIVAL AND PULSE-WIDTH/

by

LONNIE A. HADEN

B.S., Kansas State University, 1984

A MASTER'S REPORT

Submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1985

Approved by:

Donald R. Hummels

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

THIS BOOK CONTAINS NUMEROUS PAGES THAT ARE CUT OFF

THIS IS AS RECEIVED FROM THE CUSTOMER

I. INTRODUCTION

In many communication systems, measuring the width of received pulses is of great importance. Examples of these types of systems include radar and intercept receivers. In systems of this type, a key performance measure is the RMS error in the pulse-width measurement. An inexpensive way to evaluate the performance of a pulse-width measurement system is by a computer solution of the system equations. In a computer solution, many factors related to receiver performance, such as pre-filter and post-filter parameters, can be varied and hopefully, optimized. The objective of this report is to develop the necessary relationships needed to compute the RMS pulse-width measurement error, and then implement these relationships in the form of a computer program which may be used to predict performance for a measurement system. The report is intended to serve as a user's guide for the program. It begins with a review of the error analysis which is the basis for the computer program. The error analysis is then followed by a detailed description of the computer program. The description demonstrates program inputs and responses in some typical applications.

II. ERROR ANALYSIS

The main concern of the following analysis is the development of a numerical procedure which can be used to predict the performance of a pulse-width measurement system. This procedure takes into account such factors as the shape of the pulse and the transfer functions of the pre-filter and post filter of the receiver. Specifically expressions for pulse-width and RMS pulse-width error will be developed. The system of interest, commonly known as a square-law receiver is shown in Figure 1. An estimate of the width of a received pulse is obtained as the difference between the time-of-arrival (TOA) and the time-of-departure (TOD) of the pulse. The TOA event is defined as the time when the lending edge of the pulse at the output of the receiver crosses a predetermined threshold. The TOD event is defined as the time when the trailing edge of the output pulse falls below the threshold (see Figure 2). Assuming that TOA and TOD are statistically independent, a good assumption in most applications, then the RMS pulse-width error is

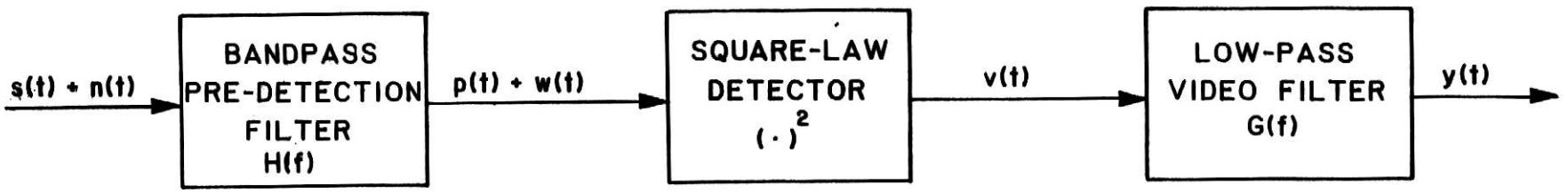
$$\sigma_p = \left|\sigma_{ta}^2 + \sigma_{td}^2\right|^{1/2} , \tag{1}$$

where σ_{ta}, σ_{td}, and σ_p are the RMS errors in TOA, TOD, and pulse-width respectively.

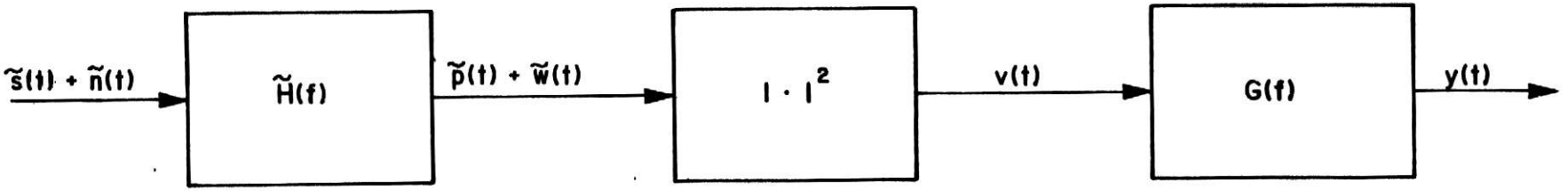
TOA and TOD Error Analysis

For the receiver system of Figure 1, the analysis for TOA and TOD are essentially the same. For this reason the development to follow will be for the TOA event only. The analysis is similar to that given in Skolnik [1].

Let ε denote TOA error. The relationship between the signal $y_s(t)$ and the signal plus noise $y(t)$ can be seen in Figure 3. Let $y_n(t)$ denote the

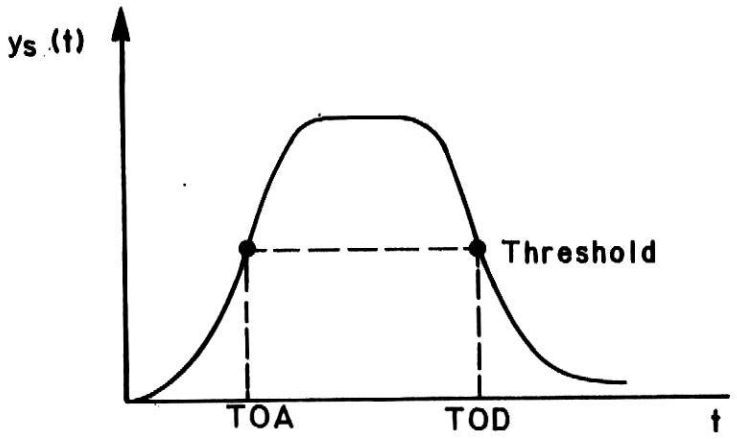


a) Physical System

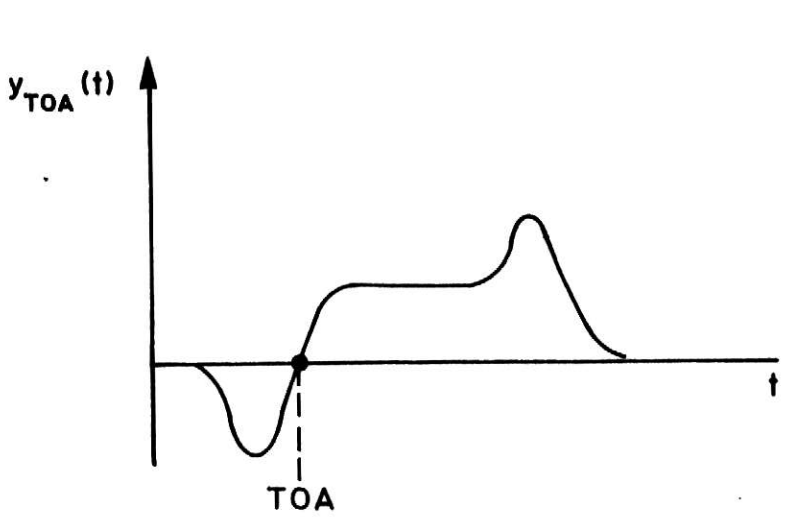


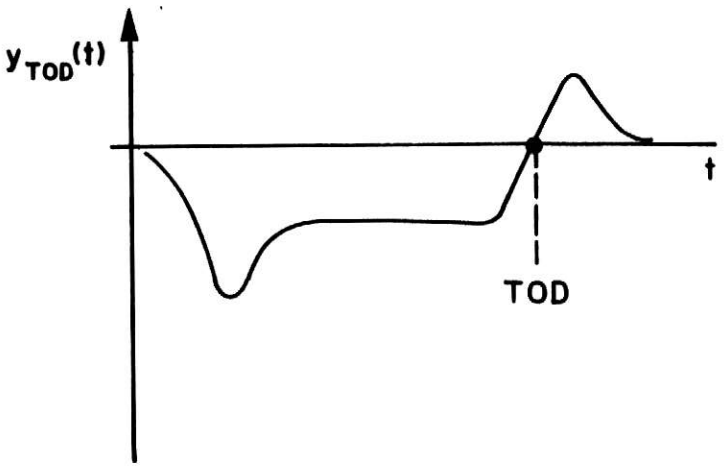
b) Equivalent Low-Pass Model

Figure 1. Block Diagram of Receiver Model



a) Single Post-filter Case





b) Two Post-Filter Case

Figure 2. Time-of-Arrival and Time-of-Departure Events

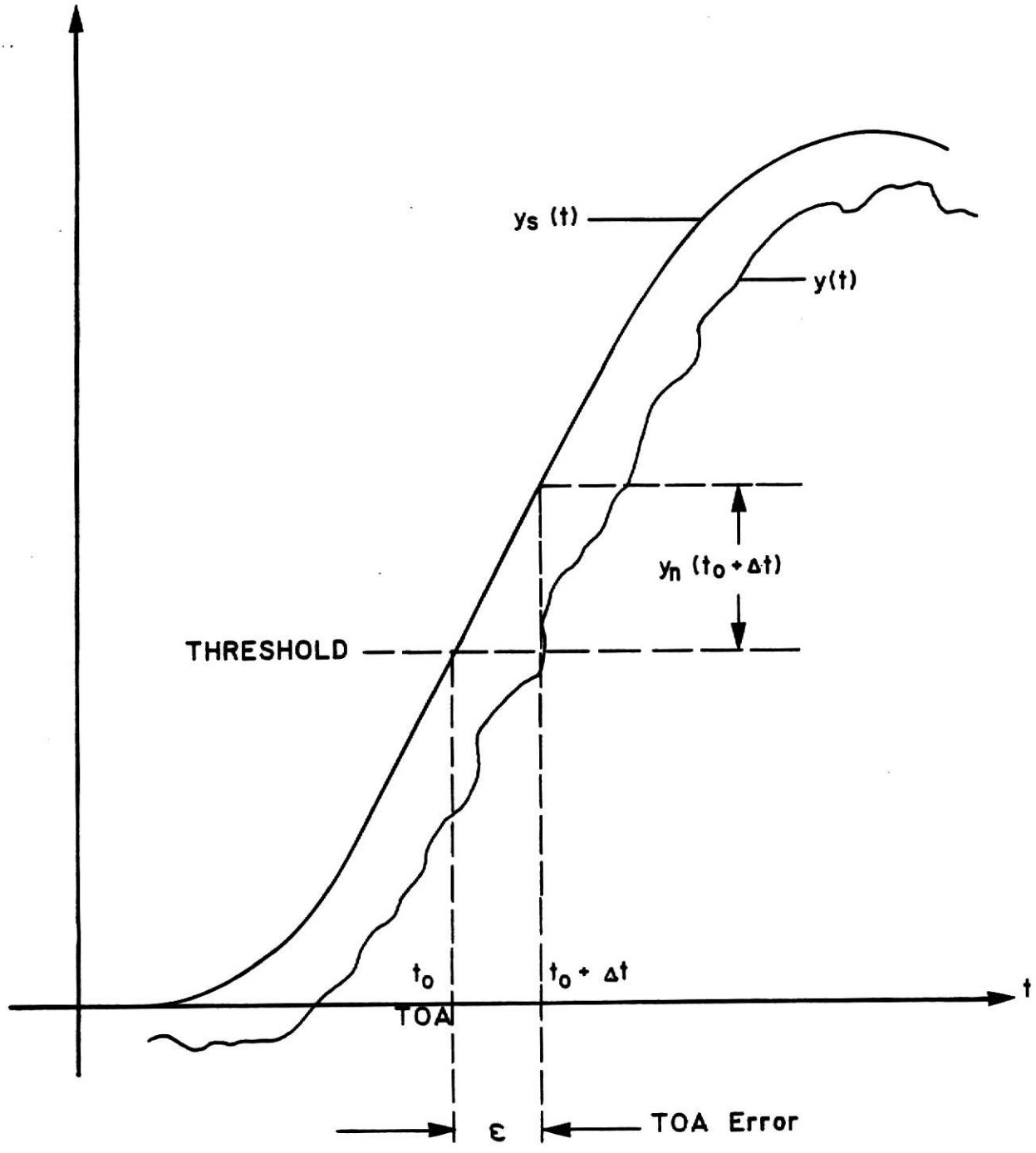


Figure 3. TOA Error Formulation

noise component of the signal. Then the slope of the signal can be approximated by

$$\dot{y}_s(t_o) = \frac{y_n(t_o + \Delta t)}{\varepsilon} \quad . \tag{2}$$

This approximation holds when the slope remains relatively constant at the threshold crossing and when the signal-to-noise ratio (SNR) is moderately high. Rearranging (2) to obtain the TOA error, ε, gives

$$\varepsilon = \frac{y_n(t_o + \Delta t)}{\dot{y}_s(t_o)} \quad . \tag{3}$$

Since the slope of a deterministic signal is not random, it is evident that the RMS TOA error σ_{ta} is related to the RMS valve of the output noise component by

$$\sigma_{ta} = \frac{\sigma_n(t_o)}{\dot{y}_s(t_o)} \quad , \tag{4}$$

where $\sigma_n(t_o)$ is the square root of the variance of the output noise process at the time of threshold crossing.

The programs described in the next section are used to calculate numerical solutions for $\sigma_n(t)$ and $\dot{y}_s(t_o)$. The key relationships (developed in Appendix A) are:

$$y_s(t) = \sum_n \sum_m s_n s_m^* \tilde{H}(\omega_n)\tilde{H}^*(\omega_m)G(\omega_n-\omega_m)e^{j(\omega_n-\omega_m)t} \tag{5}$$

$$\dot{y}_s(t) = \sum_n \sum_m j(\omega_n-\omega_m)\, s_n s_m^*\, \tilde{H}(\omega_n)\tilde{H}^*(\omega_m) \cdot G(\omega_n-\omega_m)e^{j(\omega_n-\omega_m)t} \tag{6}$$

$$\sigma_n^2(t) = 2\,\mathrm{Re}\left\{\sum_n \sum_m s_n s_m^* \hat{H}(\omega_n)\hat{H}^*(\omega_m) e^{j(\omega_n-\omega_m)t} \cdot \frac{N_o}{2} \int_{-\infty}^{\infty} |\hat{H}(f)|^2\, G(f+f_n)\, G^*(f+f_m)\,df\right\}$$

$$+ \frac{N_o^2}{4} \int_{-\infty}^{} \int_{-\infty}^{} |\hat{H}(f')|^2 |\hat{H}(f-f')|^2 |G(f)|^2\, df'df \qquad (7)$$

where

$\hat{H}(f)$ = Equivalent low-pass transfer function of the pre-filter.

$G(f)$ = Transfer function of post-filter.

N_o = Receiver noise density.

s_n = n-th Fourier co-efficient of the low-pass equivalent of the signal.

A Second Measurement System

The measurement system shown in Figure 4 is composed of the same basic components as that of the square-law receiver in Figure 1. However, the system in Figure 4 compares a half-amplitude version of the input pulse with a delayed version of the input pulse. In this way the TOA and TOD events always correspond to 50 percent of the pulse amplitude. The advantage of this system is that it is not affected by changes in the amplitude of the input pulse. Study of the system shows that the TOA event corresponds to the first zero-crossing of the TOA output signal. It should be noted that the delay time T must be greater than the rise time of the pulse and less than the pulse-width.

An equivalent model of the system in Figure 4 is shown in Figure 5. This model assumes that the power splitter characteristics are independent of frequency and uses only one square-law detector. The system of Figure 5 may be analyzed in the same manner as that of Figure 1. It is only necessary to modify the post-filter transfer function G(f). Otherwise the results obtained for the RMS TOA error case remain unchanged.

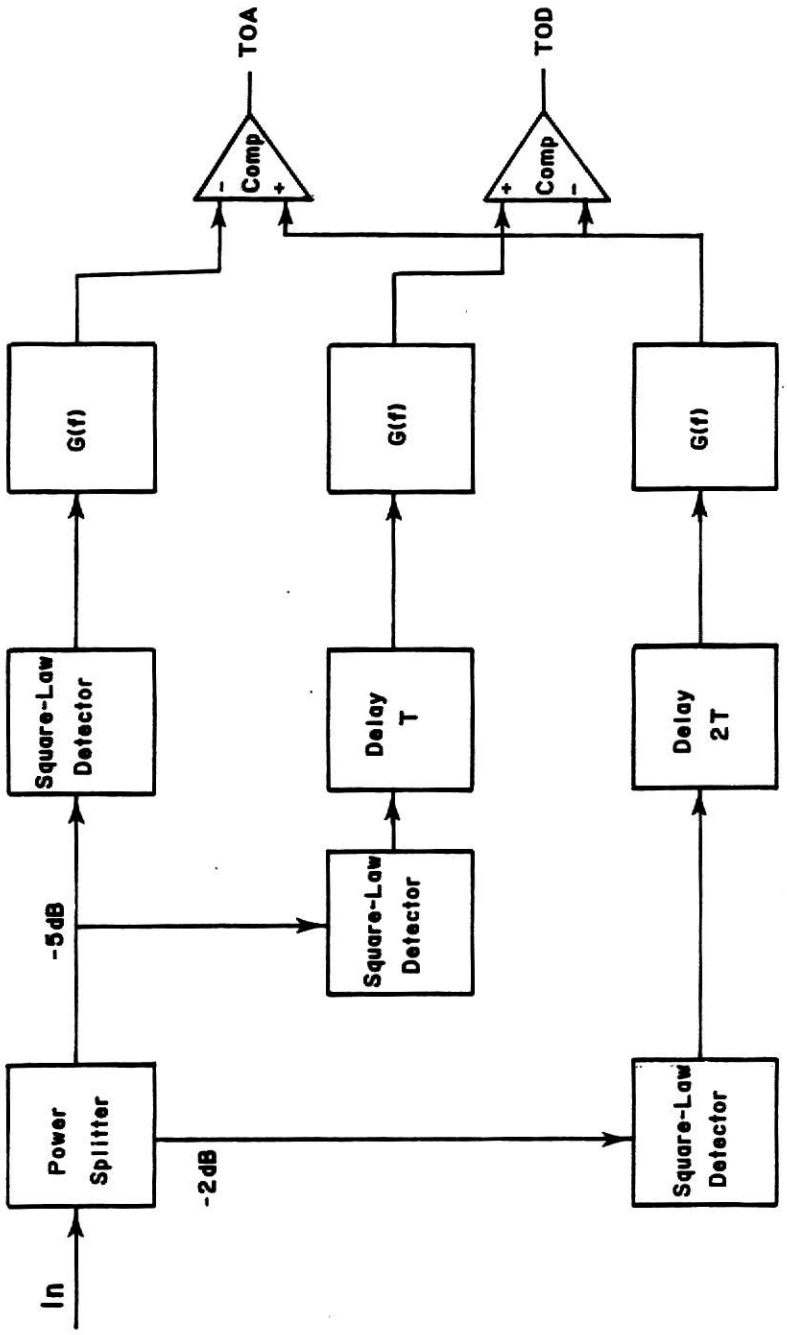


Figure 4. Pulse Width Measurement System

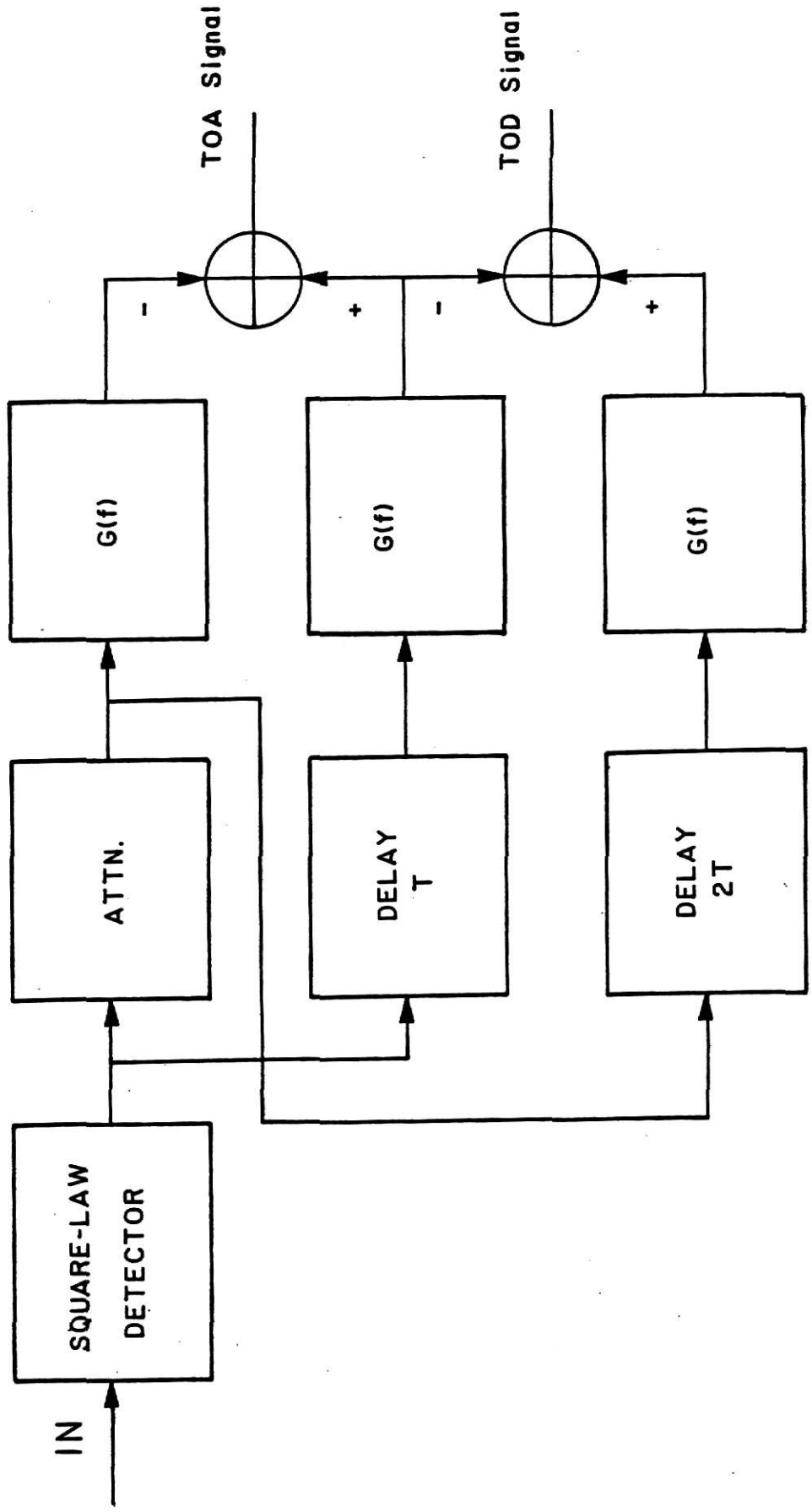


Figure 5. Equivalent System Model

Example Applications

The numerical procedures previously described are used to develop the computer program TOAANAL described in Section III. TOAANAL is useful in predicting the performance of two pulse-width measurement systems. The first is a single post-filter system, shown in Figure 1. The second is the two post-filter system shown in Figure 4.

The program TOAANAL may be used to prepare the type of graph shown in Figure 6 where σ_p (RMS pulse-wdith error) is plotted versus SNR as measured in the pre-filter bandwidth. The post-filter bandwdith is used as a parameter in this family of curves and it is clear that the best choice for this particular system is a bandwdith of 6 MHz. Using TOAANAL it is possible to explore various filter configuratons and determine the best combination for a given input signal.

The program TOAANAL can also be used to find the sigal power required for a given system to meet a specified performance level. If, for example, the RMS pulse-width error is specified as σ_p = 10 ns, then TOAANAL may be used to find the input signal power in dBm at threshold for a given configuration and pulse shape. The curves shown in Figure 7 were obtained using TOAANAL. Again, it is evident from the illustration that TOAANAL is useful for determining the best system configuration for a given pulse.

TOAANAL has been used to evaluate filter choices and bandwidths for systems used to measure the width of 20, 50, and 100 ns pulses. Pulse rise times of approximately one-third of the pulse-width were used. The results are summarized in Figures 8-10 and are given without comment except for the following observation. Because the percent variation in the width of these short pulses is large, it is necessary to use a different delay time, T, in the measurement system for each case. This may present some problem in the hardware implementation of a system for measurement of short pulses.

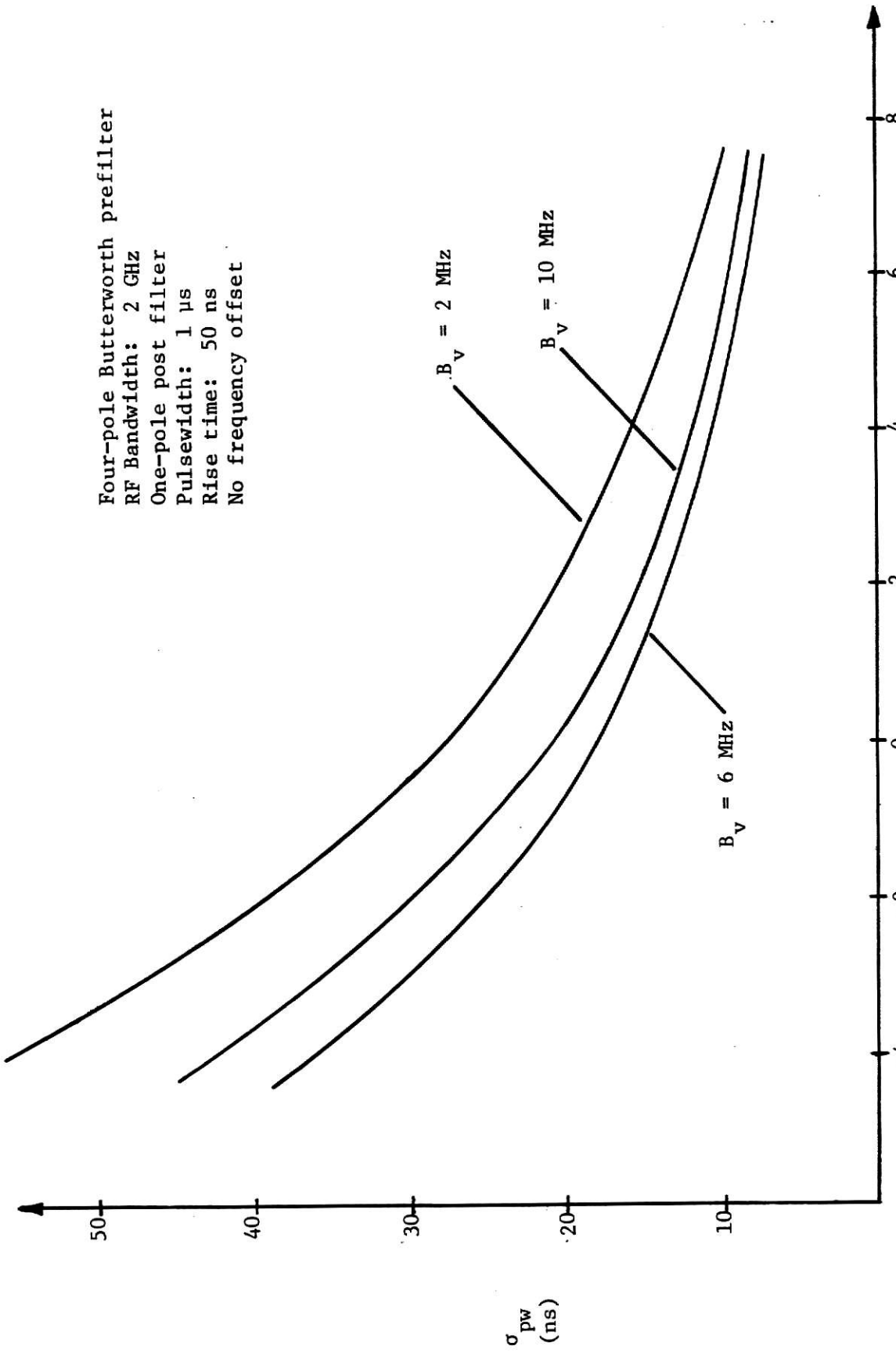


Figure 6. RMS Puslewidth Error Versus SNR

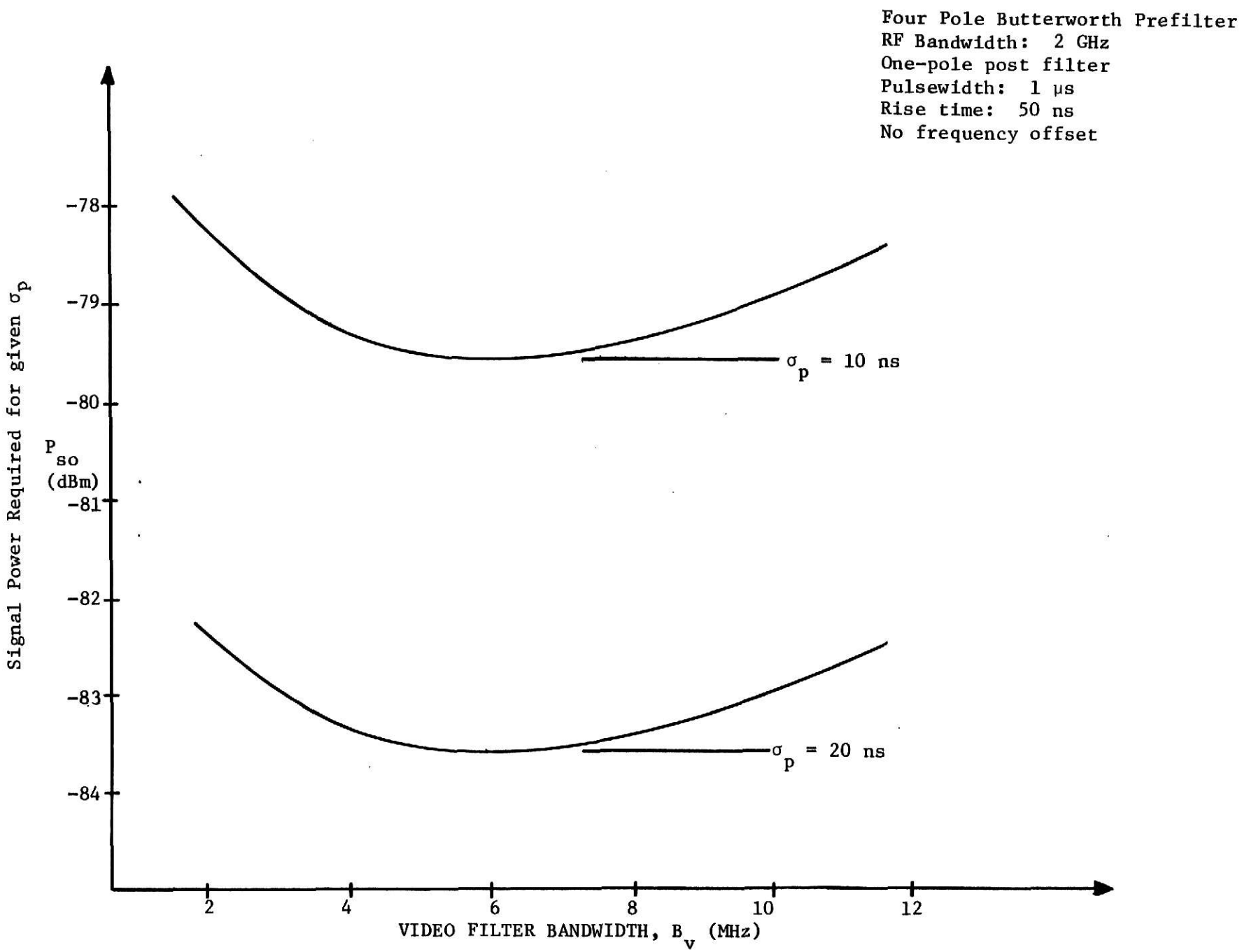


Figure 7. Signal Power Versus Video Bandwidth

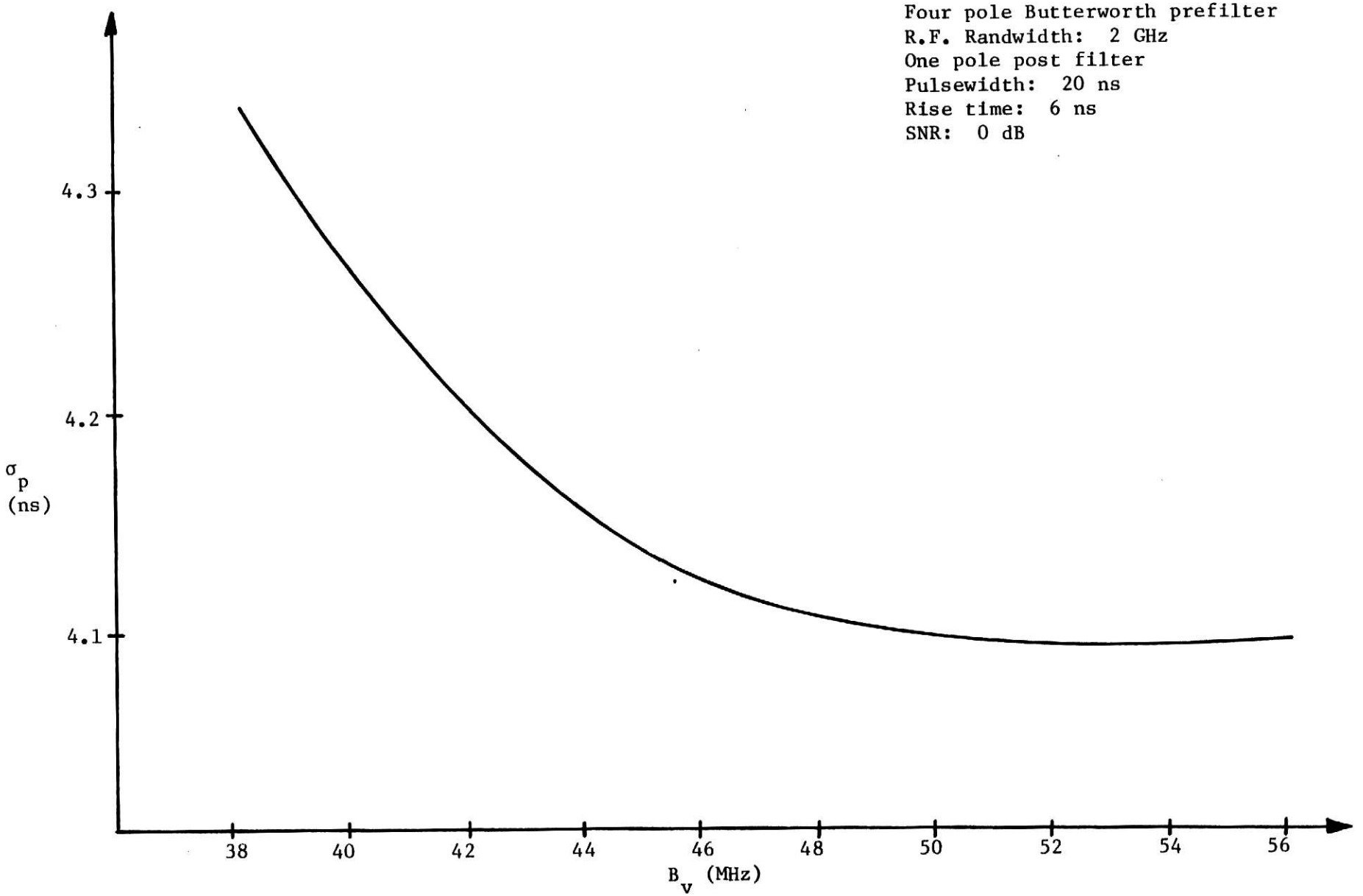


Figure 8. RMS Pulse-width Error Vs. Video Bandwidth

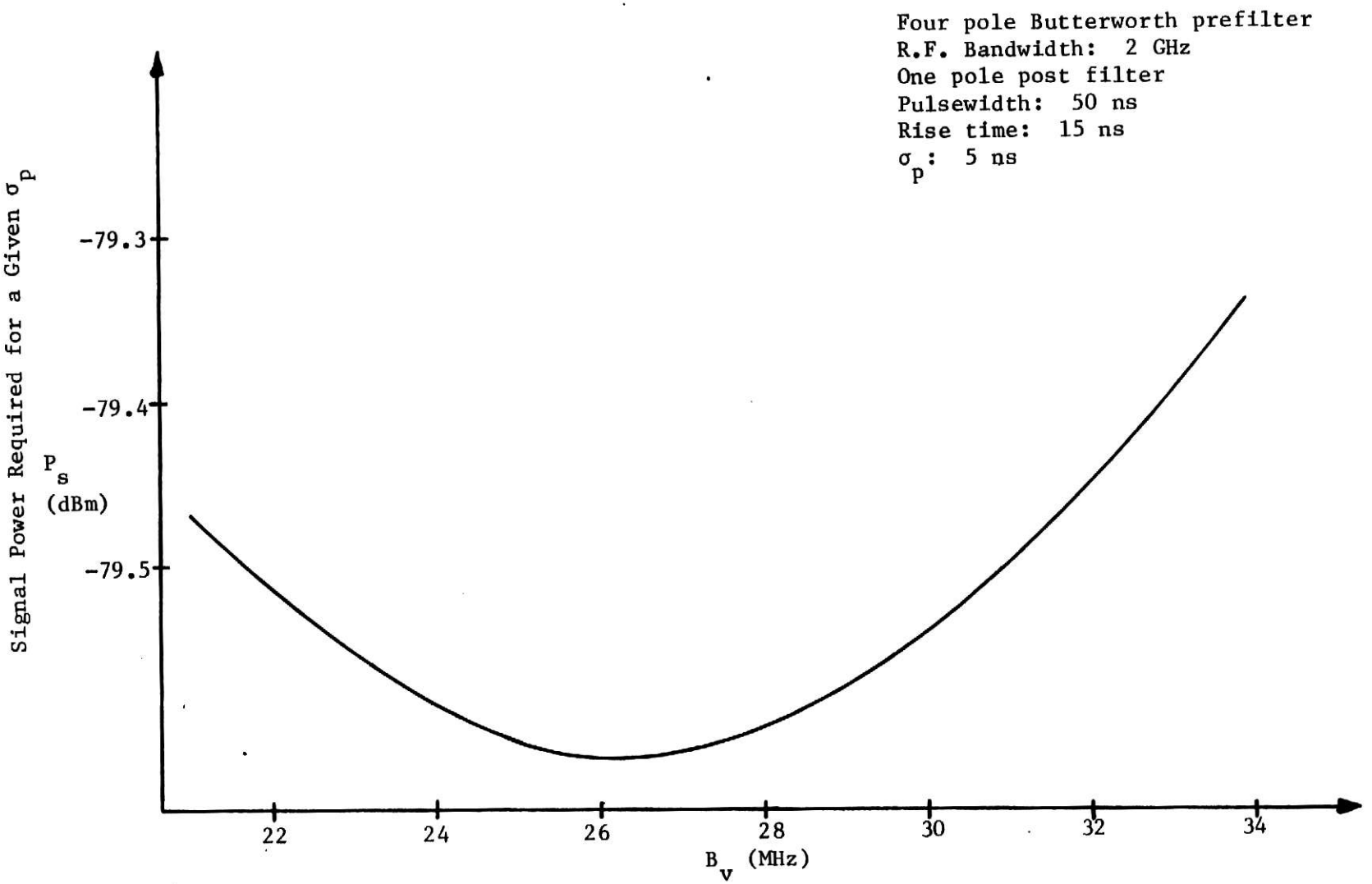


Figure 9. Signal Power for Specified Error Vs. Video Bandwidth

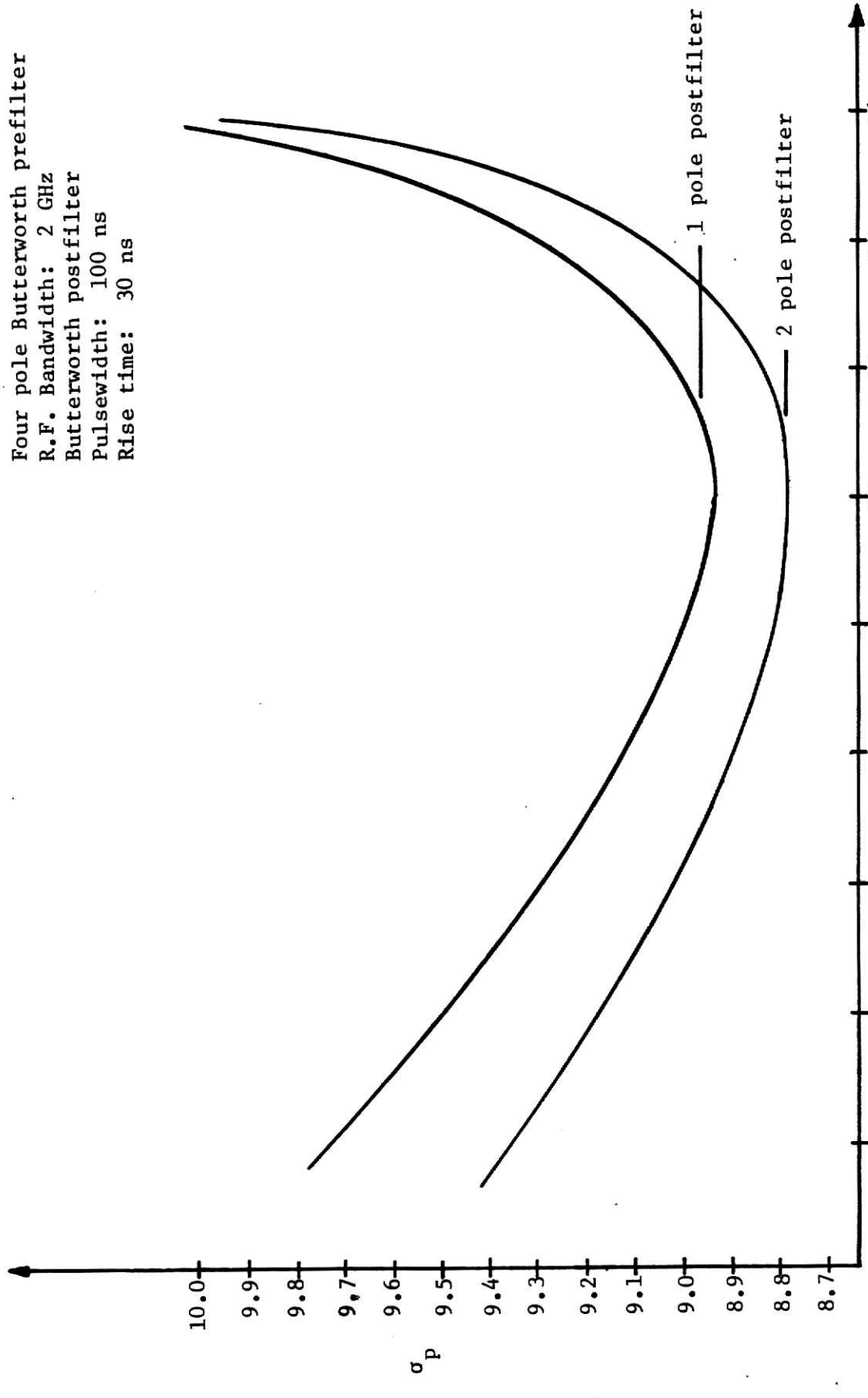


Figure 10. The Influence of Filter Order on Pulsewidth Error

III. COMPUTER PROGRAM DESCRIPTION

The equations needed to calculate the TOA signal, signal slope and noise variance for the receiver model in Figure 1 where given in Section II. Also developed, were relationships for TOA error, pulse-width, and RMS pulse-width error. TOAANAL is a menu driven program which uses these relationships to numerically evaluate receiver performance for a variety of cases. Listings for the Fortran program TOAANAL and associated subroutines described in this section are included in Appendix B. The main program, TOAANAL, is a menu-driven control program. TOAANAL controls the subroutines where the actual calculation of TOA signal, signal slope, noise variance, pulse-width, and RMS pulse-width error take place. The subroutines called by TOAANAL are: 1) Toadata, 2) Pulse, 3) Pulsel, 4) Error 4, 5) Error 41. A general flow diagram for TOAANAL is shown in Figure 11.

Subroutine Toadata

Toadata calculates the TOA or TOD signal, signal slope, and noise variance for the two post-filter model (see Figure 5) as well as for the single post-filter model (see Figure 1). Inputs to Toadata include:

Pre-filter parameters (i.e., bandwidth and number of poles)

Post-filter parameters

Delay time

Attenuation

Input pulse parameters (i.e., pulse width, rise time, and offset from center)

Fundamental frequency

Number of sample points

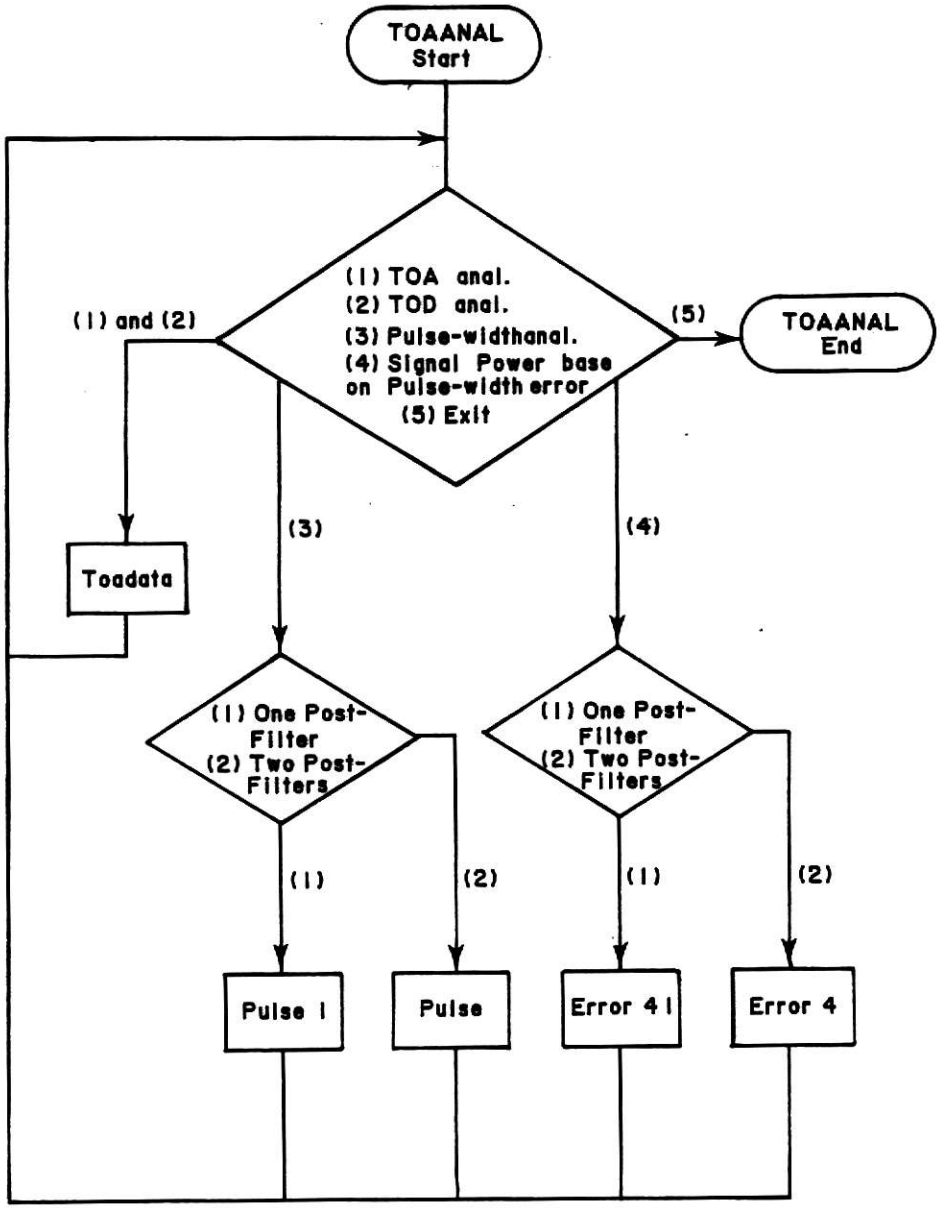


Figure 11. General Flow Diagram for TOAANAL

Signal filename

Signal slope filename

Noise variance filename.

The output of Toadata consists of a printout of all user supplied inputs and the storage on disk of the signal, signal slope, and noise variance data files. For more detailed information on data inputs and outputs see the example runs at end the of this Section. A general program flow chart for Toadata is shown in Figure 12.

Subroutines Pulse and Pulsel

The subroutines Pulse and Pulsel use the signal, signal slope and noise variance data, calculated using Toadata, to compute estimates of pulse-width and RMS pulse-width error. Subroutine Pulse is used for the two post-filter case, and Pulsel is used for the single post-filter case. The inputs to Pulse and Pulsel include:

Receiver signal-to-noise ratio (SNR)

SNR reference bandwidth

Signal filename

Signal slope filename

Noise variance filename.

The output of Pulse and Pulsel consist of a printout of all user inputs as well as:

Time-of-arrival (TOA)	TOD error
TOA error	Pulse-width
Time-of-departure (TOD)	RMS pulse-width error.

For more information on inputs and outputs see the example runs at the end of this section.

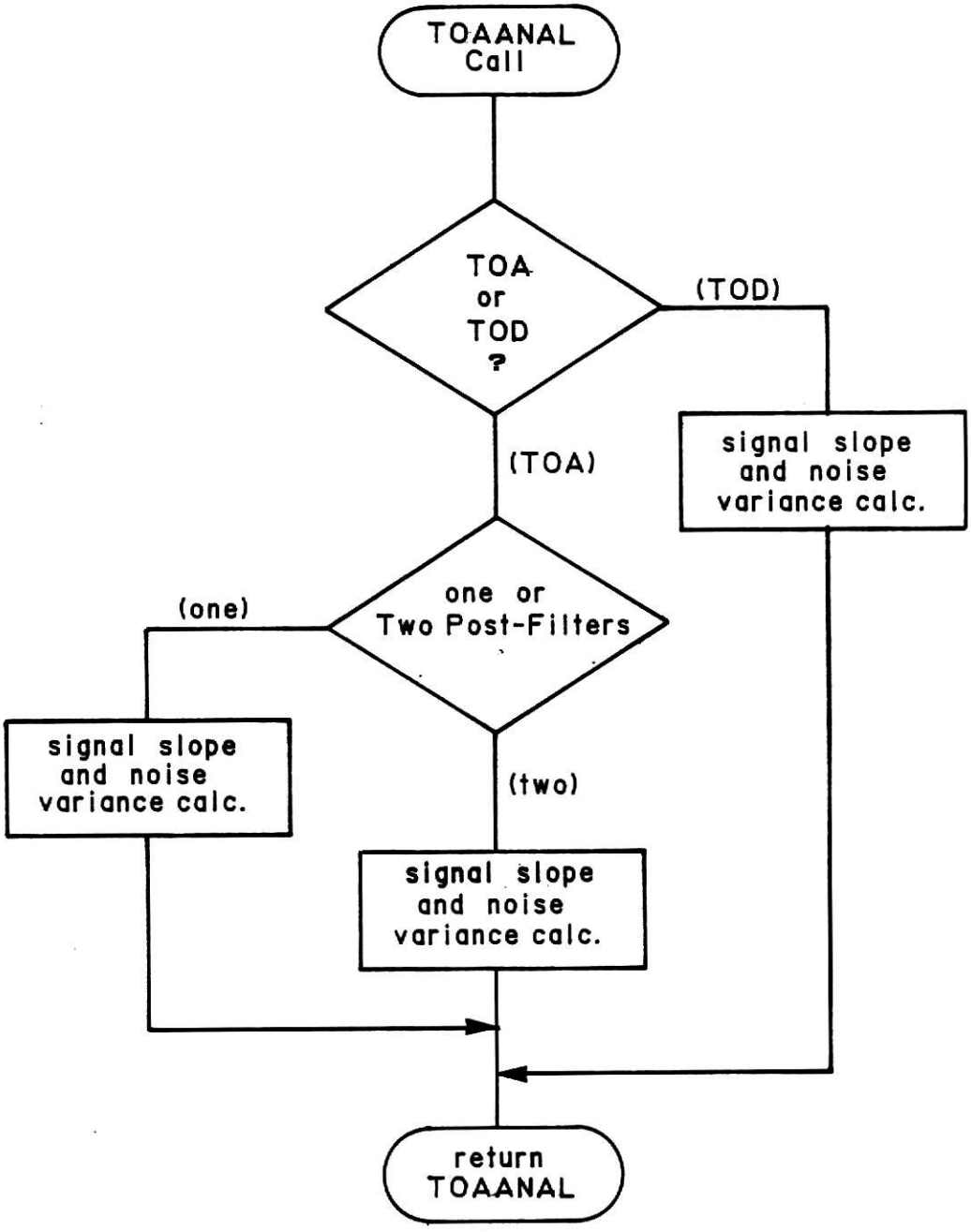


Figure 12. Toadata Flow Chart

Subroutines Error4 and Error41

The subroutines Error4 and Error41 use the signal, signal slope, and noise variance data to estimate the signal power at the receiver input needed to produce a specified pulse-width error. These subroutines also calculate the SNR corresponding to a specified pulse-width error. Subroutine Error4 is used for the two post-filter case, while Error41 is used for the single post-filter case. The inputs to Error4 and Error41 include:

Receiver noise figure
Receiver gain
RF bandwidth
RMS pulse-width error.
Signal filename
Signal slope filename
Noise variance filename

The outputs consist of a printout of all user inputs, as well as:

Signal-to-noise ratio

Signal power required to produce the specified pulse error.

For more information on inputs and outputs see the example runs which follow.

Example Runs

The following are example runs of TOAANAL. These examples show all important user prompts as well as sample user responses enclosed in angle brackets < >.

To run TOAANAL insert the disk containing the TOAANAL.EXE file into disk drive A, then type A:TOAANAL and hit return. If the default disk drive is already set to A: then just type TOAANAL. The CRT will then display a short warning about data entry errors. After the warning message the main menu shown in Figure 13 is displayed.

```
WHAT WOULD YOU LIKE TO DO?

1   TOA ANALYSIS?
2   TOD ANALYSIS?
3   PULSEWIDTH ANALYSIS?
4   SIGNAL POWER BASED ON PULSEWIDTH ERROR?
5   EXIT THE PROGRAM?

ENTER 1-5
```

Figure 13. TOAANAL main menu.

Example 1: TOA Signal, Slope, and Noise Variance Generation.

This example shows how to generate the TOA signal, TOA signal slope, and noise variance for the two post-filter case. From the main menu enter "1". The first prompt is for the pre-filter parameters and is shown in Figure 14. In this example, a butterworth pre-filter with four poles and RF bandwidth of 1 GHz is used. The next prompt asks if one or two-post filters are desired. In this case "y" for "yes" is entered. The next user prompt asks for post-filter parameters (see Figure 15). After entering the post-filter parameters, the input pulse characteristics prompt shown in Figure 16 appears. The last data prompt asks for the fundamental frequency (f_o) and the number of sample points required (see Figure 17). There are two important constraints on the fundamental frequency. The first constraint on f_o is that it must be low enough to allow the high frequency terms of the Discrete Fourier transform to approach zero at the Nyquist frequency (f_N) where the Nyquist frequency is,

$$f_N = f_o \times (\text{number of samples}/2) \quad .$$

```
SQUARE LAW DETECTOR ——→ TOA RAW DATA CALCULATIONS

BUTTERWORTH PRE-FILTER

        RF BANDWIDTH (Hz)  ?   <1e+9>

        NUMBER OF POLES (1-4)  ?  <4>
```

Figure 14. Pre-filter parameter prompt.

```
BUTTERWORTH POST-FILTER ——→ DELAY PATH

        DELAY TIME (s)  <1e-7>

        CUTOFF FREQUENCY (Hz)  ?  <2e+6>

        NUMBER OF POLES (1-4)  ?  <3>

BUTTERWORTH POST-FILTER ——→ ATTENUATION + DELAY PATH

        ATTENUATION (dB)  ?   <-3>

        CUTOFF FREQUENCY (Hz)  ? <2e+6>

        NUMBER OF POLES (1-4)  ? <3>
```

Figure 15. Post-filter parameter prompt.

```
INPUT PULSE

        WIDTH (s)    ?  <1e-6>

        RISE TIME (s)    ?    <50e-9>

        OFFSET FROM CENTER (Hz)    ?  <0>
```

Figure 16. Input pulse characteristics prompt.

```
MISCELLANEOUS

        FUNDAMENTAL FREQUENCY (Hz)    ?    <400000>

        NUMBER OF SAMPLES (128,256,512)    ?    <128>
```

Figure 17. Miscellaneous inputs.

```
FILE TO CONTAIN TOA SIGNAL?          <sig>

FILE TO CONTAIN TOA SLOPE?           <A:slope>

FILE TO CONTAIN TOA SIGMAS?          <A:sigmas.dat>
```

Figure 18. Filename prompt.

The second constraint on f_o is that it must be low enough that the system response to a pulse decays to zero within one period. The number of sample points needed depends on the rise time and width of the input pulse. If not enough sample points are used the input pulse will not be accurately represented. The best guess is to start with 256 and go from there.

This concludes the data entry section of the program. The data previously entered is now displayed so that any needed corrections can be made. The last prompt ask for the names of the files which will store the TOA signal, TOA signal slope and noise variance (see Figure 18). Two things should be noted in this example. First, if the default disk drive is A: then all the files will be written on disk A. Second the .dat is not necessary, but it may be helpful when directory size is large. As with data entry, the file names can be corrected if one or more were entered incorrectly. After about four minutes (for 128 points) the computer computations will be completed and the output in Figure 19 will be printed.

This example showed how to generate the TOA signal, TOA signal slope and noise variance for the two post-filter case. In order to generate the TOD Signal, TOD signal slope, and noise variance, selection (2) should be entered from the main menu. After that there is virtually no difference in data input or filename input.

Example 2: Output Signal, Slope and Noise Variance for One Post-Filter.

Example 2 shows how to generate the output signal, signal slope and noise variance for the one post-filter case. There is virtually no difference between this example and example 1.

```
SQUARE LAW DETECTOR ——→ TOA RAW DATA CALCULATIONS
     BUTTERWORTH PRE-FILTER
          RF BANDWIDTH              : 0.100000E+10 Hz
          NUMBER OF POLES           : 4
     BUTTERWORTH POST-FILTER ——→ DELAY PATH
          DELAY TIME                : 0.100000E-06 s
          CUTOFF FREQUENCY          : 0.200000E+07 Hz
          NUMBER OF POLES           : 3
     BUTTERWORTH POST-FILTER ATTENUATION + DELAY PATH
          ATTENUATION               : -3.00000     dB
          CUTOFF FREQUENCY          : 0.200000E+07 Hz
          NUMBER OF POLES           : 3
     INPUT PULSE
          WIDTH                     : 0.100000E-05 s
          RISE TIME                 : 0.500000E-07 s
          OFFSET FROM CARRIER       :  .000000     Hz
     MISCELLANEOUS
          FUNDAMENTAL FREQUENCY     : 400000.      Hz
          NUMBER OF SAMPLES         : 128

               TOA SIGNAL STORED IN FILE:    sig
               TOA SLOPE STORED IN FILE:     A:slope
               TOA SIGMAS STORED IN FILE:    A:sigmas.dat
```

Figure 19. Output of Toadata.

From the main menu enter "1". Then enter the rest of the parameters as in example 1 except for the two post-filter prompt (see Figure 20). For one post-filter enter "n" (for "no").

The output for this case (see Figure 21) looks much like the output for example 1.

Example 3: Obtaining Pulse-Width and RMS Pulse-Width Error

This example shows how to obtain estimates for pulse-width and RMS pulse-wdith error. This example assumes that the appropriate data files have already been generated, i.e., the TOA and TOD signal, signal slope and noise variance files.

```
SQUARE LAW DETECTOR ——→ TOA RAW DATA CALCULATIONS

BUTTERWORTH PRE-FILTER

          RF BANDWIDTH (Hz)  ?   <1e9>

          NUMBER OF POLES (1-4)  ?    <4>

TWO POST-FILTERS                 ?

          ANSWER YES OR NO  : <n>

BUTERWORTH POST-FILTER

          CUTOFF FREQUENCY (Hz)  ?   <2e+6>

          NUMBER OF POLES (1-4)  ?   <1>

INPUT PULSE

          WIDTH (s)  ?   <1e-6>

          RISE TIME (s)  ?   <50e-9>

          OFFSET FROM CENTER (Hz)  ?   <0>

MISCELLANEOUS

          FUNDAMENTAL FREQUENCY (Hz)  ?   <400000>

          NUMBER OF SAMPLES (128,256,512)  ? <128>
```

Figure 20. Input prompts for one post-filter.

```
SQUARE LAW DETECTOR ——→ TOA RAW DATA CALCULATIONS

     BUTTERWORTH PRE-FILTER

          RF BANDWDITH          :  0.100000E+10 Hz
          NUMBER OF POLES       :  4

     BUTTERWORTH POST-FILTER

          CUTOFF FREQUENCY      :  0.200000E+07 Hz
          NUMBER OF POLES       :  1

     INPUT PULSE

          WIDTH                 :  0.100000E-05 s
          RISE TIME             :  0.500000E-07 s
          OFFSET FROM CARRIER   :   .000000     Hz

     MISCELLANEOUS

          FUNDAMENTAL FREQUENCY :  400000.      Hz
          NUMBER OF SAMPLES     :    128

          TOA SIGNAL STORED IN FILE:   sig1

          TOA SLOPE STORED IN FILE:    slope1

          TOA SIGMAS STORED IN FILE:   sigmas1
```

Figure 21. Output of Toadata for one post-filter.

From the main menu enter "3". The first prompt asks if the pulse-width analysis is for the one post-filter model or the two post-filter model. In this example the two post-filter model is chosen (see Figure 22).

```
WHICH WOULD YOU LIKE TO PERFORM?

     1.  PULSEWIDTH ANALYSIS (ONE POST-FILTER)

     2.  PULSEWIDTH ANALYSIS (TWO POST-FILTER)

ENTER (1) OR (2).    <2>
```

Figure 22. Pulse-width analysis prompt.

Next, the file names of the TOA and TOD data files are entered as shown in Figure 23.

```
PULSE WIDTH ESTIMATE ——→ TWO POST-FILTERS

     TOA SIGNAL FILE?        <sig>
     TOA SLOPE FILE?         <slope>
     TOA SIGMAS FILE?        <sigmas>

     TOD SIGNAL FILE?        <dsig>
     TOD SLOPE FILE?         <dslope>
     TOD SIGMAS FILE?        <dsigmas>
```

Figure 23. Filename prompt.

After the program has read the data from disk the signal-to-noise ratio and SNR reference bandwidth are entered. There is also a prompt which asks if the threshold occurs after the negative peak signal (see Figure 24). In most cases the signal does occur after the negative peak signal, but for some combinations of filter selections it may not. Answer yes in general. If the results are supspect; it will be obvious, then try no.

```
SIGNAL TO NOISE RATIO (dB)  ?  <10>
SNR REFERENCE BANDWDITH (Hz)  ?  <1e+7>

SEARCH FOR THRESHOLD AFTER OCCURRENCE OF
NEGATIVE PEAK SIGNAL?

ANSWER YES OR NO  : <y>
```

Figure 24. SNR and SNR reference prompts.

This section of the program can be run for as many different SNR and SNR reference bandwidths as necessary. This section of the program may be exited by entering -101 for the SNR. The output is shown in Figure 25. Because there is little difference between the two post-filter prompts, and one post-filter prompts, no example for the one post-filter case will be given.

```
PULSE WIDTH ESTIMATE ——→ TWO POST-FILTERS


USER SUPPLIED INFORMATION

     TOA SIGNAL FILE      : sis

     TOA SLOPE FILE       : slope

     TOA SIGMAS FILE      : sigmas

     TOD SIGNAL FILE      : dsis

     TOD SLOPE FILE       : dslope

     TOD SIGMAS FILE      : dsigmas


     THRESHOLD                  :   .000000      %

     SIGNAL TO NOISE RATIO      : 10.0000        db

     SNR REFERENCE BANDWIDTH    :  0.100000E+08 Hz


                    RESULTS

          AT SPECIFIED PERCENTAGE OF PEAK SIGNAL
TIME-OF-ARRIVAL    (0.295807E-06s) AND TOA ERROR 0.142179E-06 s
TIME-OF-DEPARTURE (0.129954E-05s) AND TOD ERROR 0.138666E-06 s
     PULSE WIDTH (0.100373E-05s) = 0.198603E-06 s
```

Figure 25. Pulse-width analysis output.

Example 4: Signal Power Based on Pulse-Width Error

This examples shows the user prompts and responses needed to obtain estimates of the signal power (at the receiver) needed to satisfy a given pulse-width error specification. This example assumes that the TOA and TOD data files have already been generated.

From the main menu enter "4". As in example 3, the filenames containing the TOA and TOD data are entered first. Next the receiver noise figure, gain, and RF bandwidth are entered as shown in Figure 26. There is a slight delay as the data files are read. Then the user specified pulse-width error is input as shown in Figure 27.

```
NOISE FIGURE (dB)     ? <20>
RECEIVER GAIN (dB)    ? <10000>
RF BANDWDITH (Hz)     ? <1e+9>

SEARCH FOR THRESHOLD AFTER OCCURENCE
OF NEGATIVE PEAK SIGNAL?
ANSWER YES OR NO   : <y>
```

Figure 26. Receiver prompts.

```
INPUT PULSE WIDTH OF -1 TO EXIT
PULSE WIDTH ERROR (s) ? <1e-9>
```

Figure 27. Pulse-width error prompt.

The output is shown in Figure 28. Note that the signal-to-noise ratio corresponding to the receiver input power is also given. Since the prompts for the two post-filter case are almost identical to those of the one post-filter case, no example for the latter is given.

```
SIGNAL POWER BASED ON PULSEWIDTH ERROR
(TWO POST-FILTERS)

USER SUPPLIED INFORMATION COMMON TO ALL CASES

TOA SIGNAL FILE     :  signal

TOA SLOPE FILE      :  slope

TOA SIMGAS FILE     :  simgas

TOD SIGNAL FILE     :  dsignal

TOD SLOPE FILE      :  dslope

TOD SIGMAS FILE     :  dsimgas

THRESHOLD           :   .000000      %
NOISE FIGURE        :   10.0000      dB
RECEIVER GAIN       :   100.000      dB
RF BANDWIDTH        :  0.100000E+10  Hz
FIRST THRESHOLD AFTER NEGATIVE PEAK SIGNAL

                RESULTS
PULSE WIDTH ERROR   : 0.100000E-07  s
SIGNAL POWER =       -68.7040       dBm
SNR          =        5.29602       dB

PULSE WIDTH ERROR   : 0.100000E-08  s
SIGNAL POWER =       -50.3958       dBm
SNR          =        32.6042       dB
```

Figure 28. Output of Error4.

Program Requirements and Limitations

The program TOAANAL was developed using a Zenith Z-150 personal computer. This particular computer was equipped with 640 K of RAM, a 10

megabyte harddisk, an 8087 co-processor, and ran on version 2.11 of the MS DOS operating system. TOAANAL has been run without modification on an IBM PC XT. The IBM computer had 512 K of RAM and ran on the PC DOS operating system.

The program TOAANAL needs approximately 300 K of RAM to run. Therefore, any IBM compatible PC type computer with 320 K of RAM should run this program. TOAANAL takes about thirty-one (31) minutes to generate signal, signal slope, and noise variance when 512 sample points are used to represent the signal. It takes approximately nine minutes when 256 points are used for signal representation.

IV. CONCLUSION

In this report a computer program which can be used to predict pulse-width and RMS pulse-width measurement error was developed. The program is useful for square-law pulse receivers which measure the width of received pulses. Systems of this type include radar and intercept receivers. The advantage of the numerical procedures used by the program is that input pulse shape and the transfer functions of the receiver are taken into account. The program has been tested on both a Zenith Z-150 and an IBM PC XT, with these tests producing the same correct solutions.

ACKNOWLEDGMENTS

I wish to express my appreciation for the support afforded me by my parents, friends, and particularly that from the members of my graduate committee, Dr. D. R. Hummels, Dr. S. A. Dyer and Dr. L. Dean Bark, for without their assistance this report would not have been possible.

APPENDIX A

SYSTEM EQUATIONS

The following numerical procedure is taken from Hummels and Ratcliffe [2]. A simplified block diagram of the receiver under consideration together with its equivalent low pass model is shown in Figure 1 of Section II. It consists of a bandpass pre-filter having transfer function H(f) followed by a square-law envelope detector and low pass (video) filter with transfer function G(f). The input to the receiver is a signal of interest, s(t), plus a stationary bandpass Gaussian noise process, n(t). All of the results presented herein are for the case where n(t) is a white noise process with power spectrum

$$S_n(f) = N_o/2, \tag{1}$$

over a band of frequencies somewhat wider than the pre-filter bandwidth; however, this is not a requirement of the analysis.

The equivalent low-pass model is developed in the usual way [3] with a signal and its complex envelope related via

$$s(t) = \mathrm{Re}\{\tilde{s}(t)e^{j\omega_c t}\} \tag{2}$$

where $\tilde{s}(t)$ is the complex envelope of s(t) and ω_c is the center frequency or carrier frequency of the signal and is taken as the center frequency of the bandpass pre-filter for convenience. The complex envelope, $\tilde{n}(t)$, of the input noise process is a stationary complex Gaussian process. The power spectrum of $\tilde{n}(t)$ is related to that of n(t) by the relation

$$S_n(f) = \frac{1}{4} S_{\tilde{n}}(f - f_c) + \frac{1}{4} S_{\tilde{n}}(-f - f_c) \quad , \tag{3}$$

thus the power spectrum of $\tilde{n}(t)$ to be used here is

$$S_{\tilde{n}}(f) = 2\ N_o \tag{4}$$

over a band of frequencies somewhat wider than the band passed by the low pass equivalent of the pre-filter. The transfer function of the pre-filter and its low-pass equivalent, $\tilde{H}(f)$, are related via

$$H(f) = \tilde{H}(f - f_c) + \tilde{H}(-f - f_c) \tag{5}$$

The complex envelope of the signal at the pre-filter output is a sum $\tilde{p}(t) + \tilde{w}(t)$ of filtered signal and noise envelopes respectively. The action of the square-law detector is to produce a voltage v(t) proportional to the magnitude squared of this complex envelope, i.e.,

$$v(t) = |\tilde{p}(t) + \tilde{w}(t)|^2$$

or,

$$v(t) = |\tilde{p}(t)|^2 + \tilde{p}(t)\tilde{w}^*(t) + \tilde{p}^*(t)\tilde{w}(t) + |\tilde{w}(t)|^2 \tag{6}$$

The output, y(t), is obtained by passing v(t) through a low-pass filter with transfer function G(f).

It is evident from (6) that $|\tilde{p}(t)|^2$ is the desired signal component at the detector output. In the next section the output produced by this term is determined. The remaining terms constitute a nonstationary noise process.

AN EQUATION FOR THE OUTPUT SIGNAL

Calculation of the output signal as a function of time is facilitated by expanding the input envelope $\tilde{s}(t)$ in a Fourier series, via

$$\tilde{s}(t) = \sum_{n=-\infty}^{\infty} s_n e^{j\omega_n t} \tag{7}$$

where $\omega_n = n\,\omega_o = \frac{n2\pi}{T}$

and $$s_n = \frac{1}{T} \int \tilde{s}(t) e^{-j\omega_n t}\, dt.$$

At this point, no restrictions are placed on $\tilde{s}(t)$ other than it be periodic and have a Fourier expansion. One case of interest is that where $\tilde{s}(t)$ is a periodic rectangular pulse train, in which case

$$s_n = A\tau f_o \text{ sinc } (n\pi f_o \tau) \tag{8}$$

where A is the pulse amplitude, τ is the pulse width and f_o is pulse repitition frequency, i.e., the fundamental frequency of the pulse train.

The Fourier series for $\tilde{p}(t)$ may be written by inspection as

$$\tilde{p}(t) = \sum_{n=-\infty}^{\infty} p_n e^{j\omega_n t} \tag{9}$$

where $$p_n = s_n \tilde{H}(\omega_n). \tag{10}$$

The signal component at the square-law detector output is now found by forming $|\tilde{p}(t)|^2$ as

$$|\tilde{p}(t)|^2 = \tilde{p}(t)\, \tilde{p}^*(t)$$

or

$$|\tilde{p}(t)|^2 = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} p_n p_m^* e^{j(\omega_n - \omega_m)t} . \tag{11}$$

The signal portion of the output y(t) may be found by convolving $|\tilde{p}(t)|^2$ with the impulse response g(t) of the output filter. This amounts to modifying each term of the series given in (11) by the transfer function $G(\omega)$ evaluated at the frequency of the exponential. The output signal is thus

$$y_s(t) = \sum_n \sum_m p_n p_m^* G(\omega_n - \omega_m) e^{j(\omega_n - \omega_m)t} .$$

Substitution for p_n and p_m^* using (1) yields the following form which is useful for numerical evaluation of $y_s(t)$,

$$y_s(t) = \sum_n \sum_m s_n s_m^* \tilde{H}(\omega_n) \tilde{H}^*(\omega_m) G(\omega_n - \omega_m) e^{j(\omega_n - \omega_m)t} \tag{12}$$

DETERMINATION OF THE CORRELATION FUNCTION OF THE OUTPUT NOISE PROCESS

Let $\eta(t)$ denote the noise components at the square-law detector output. Then,

$$\eta(t) = \tilde{p}(t)\,\tilde{w}^*(t) + \tilde{p}^*(t)\tilde{w}(t) + |\tilde{w}(t)|^2. \tag{13}$$

Clearly, $\eta(t)$ is a nonstationary process because of the product terms involving signal and noise. Our objective is to determine the correlation function and mean function of the output noise generated by $\eta(t)$. We begin by determining the mean function of $\eta(t)$

The Mean Function, $\overline{\eta(t)}$

The mean of $\eta(t)$ is found directly as

$$\begin{aligned}\overline{\eta(t)} &= E\{\eta(t)\} \\ &= \tilde{p}(t)\,E\{\tilde{w}^*(t)\} + \tilde{p}^*(t)E\{\tilde{w}(t)\} + E\{|\tilde{w}(t)|^2\}\ .\end{aligned}$$

Since $\tilde{w}(t)$ is a zero-mean process, it follows that

$$\begin{aligned}\overline{\eta(t)} &= E\{|\tilde{w}(t)|^2\} \\ &= E\{\tilde{w}(t)\,\tilde{w}^*(t)\} \\ &= R_{\tilde{w}}(0).\end{aligned}$$

Thus the mean of $\eta(t)$ is constant at

$$\overline{\eta(t)} = R_{\tilde{w}}(0) \tag{14}$$

where

$$R_{\tilde{w}}(0) = \int_{-\infty}^{\infty} S_{\tilde{n}}(f)\,|\tilde{H}(f)|^2\,df. \tag{15}$$

Mean of the Output Noise, $\overline{y_n(t)}$.

Let $y_n(t)$ denote the noise portion of $y(t)$, i.e., the noise generated by passing $\eta(t)$ through the low-pass filter. The mean of $y_n(t)$ is readily obtained as

$$\overline{y_n(t)} = \overline{\eta(t)}\, G(0)$$

or equivalently

$$\overline{y_n(t)} = G(0) \int_{-\infty}^{\infty} S_{\tilde{n}}(f) |\tilde{H}(f)|^2 \, df \quad . \tag{16}$$

The Correlation Function of $\eta(t)$.

The first step in finding the correlation function of the output noise is the determination of the correlation function of $\eta(t)$. As noted earlier, $\eta(t)$ is nonstationary, thus it is necessary to evaluate the form

$$R_\eta(t,s) = E\{\eta(t)\eta^*(s)\} \quad . \tag{17}$$

Substitution with the aid of (13) yields

$$\begin{aligned} R_\eta(t,s) = E\{&[\tilde{p}(t)\tilde{w}^*(t) + \tilde{p}^*(t)\tilde{w}(t) + \tilde{w}(t)\,\tilde{w}^*(t)] \\ &\cdot [\tilde{p}^*(s)\tilde{w}(s) + \tilde{p}(s)\tilde{w}^*(s) + \tilde{w}^*(s)\tilde{w}(s)]\} \quad . \end{aligned} \tag{18}$$

Expanding the indicated product leads to

$$\begin{aligned} R_\eta(t,s) = \; & \\ & E\{\tilde{p}(t)\tilde{p}^*(s)\tilde{w}^*(t)\tilde{w}(s) + \tilde{p}(t)\tilde{p}(s)\tilde{w}^*(t)\tilde{w}^*(s) + \tilde{p}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s) \\ & + \tilde{p}^*(t)\tilde{p}^*(s)\tilde{w}(t)\tilde{w}(s) + \tilde{p}^*(t)\tilde{p}(s)\tilde{w}(t)\tilde{w}^*(s) + \tilde{p}^*(t)\tilde{w}(t)\tilde{w}^*(s)\tilde{w}(s) \\ & + \tilde{p}(s)\tilde{w}(t)\tilde{w}^*(t)\tilde{w}(s) + \tilde{p}(s)\tilde{w}(t)\tilde{w}^*(t)\tilde{w}(s) + \tilde{w}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s)\} \end{aligned} \tag{19}$$

The fact that $\tilde{w}(t)$ is a zero-mean stationary complex Gaussian random process allows for considerable simplification [4] when carrying out the indicated expectations. The non-zero terms which remain after averaging are

$$\begin{aligned} R_\eta(t,s) = \; & \tilde{p}(t)\tilde{p}^*(s)\, E\{\tilde{w}^*(t)\tilde{w}(s)\} \\ & + \tilde{p}^*(t)\tilde{p}(s)\, E\{\tilde{w}(t)\tilde{w}^*(s)\} \\ & + E\{\tilde{w}(t)\tilde{w}^*(t)\}\, E\{\tilde{w}^*(s)\tilde{w}(s)\} \\ & + E\{\tilde{w}(t)\tilde{w}^*(s)\}\, E\{\tilde{w}^*(t)\tilde{w}(s)\} \end{aligned} \tag{20}$$

which may be rearranged as

$$R_\eta(t,s) = [\tilde{p}(t)\tilde{p}^*(s) + \tilde{p}^*(t)\tilde{p}(s)]\, R_{\tilde{w}}(t-s) + R^2_{\tilde{w}}(t-s) + R^2_{\tilde{w}}(0) \quad . \tag{21}$$

where $R_{\tilde{w}}(t-s)$ is the correlation function of $\tilde{w}(t)$ and may be obtained via the transform relation

$$R_{\tilde{w}}(t-s) = \int_{-\infty}^{\infty} S_{\tilde{w}}(f) e^{j2\pi f(t-s)}\, df \tag{22}$$

The power spectrum $S_{\tilde{w}}(f)$ is in turn found from the power spectrum of the input noise process using

$$S_{\tilde{w}}(f) = S_n(f)\left|\tilde{H}(f)\right|^2 \quad . \tag{23}$$

The Correlation Function of $y_n(t)$.

Since $y_n(t)$ is the result of a linear filtering operation on $\eta(t)$, we may find $R_{yn}(t,s)$ using a well known [5] relation, namely

$$Ry_n(t,s) = \int_{-\infty}^{\infty}\int_{\infty}^{\infty} R_\eta(\alpha,\beta)\; g(t-\alpha)g(s-\beta)d\alpha d\beta \quad . \tag{24}$$

Substitution for $R_\eta(\alpha,\beta)$ leads to

$$R_{yn}(t,s) = R_{y1}(t,s) + R_{y2}(t,s) + R_{y3}(t,s) + R_{y4}(t,s) \tag{25}$$

where

$$R_{y_1}(t,s) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{p}(\alpha)\tilde{p}(\beta)R_{\tilde{w}}(\alpha-\beta)g(t-\alpha)g(s-\beta)d\alpha d\beta \tag{26}$$

$$R_{y2}(t,s) = R^*_{y1}(t,s)$$

$$R_{y_3}(t,s) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R^2_{\tilde{w}}(\alpha-\beta)g(t-\alpha)g(s-\beta)d\alpha d\beta \tag{27}$$

$$R_{y_4}(t,s) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R^2_{\tilde{w}}(0)g(t-\alpha)g(s-\beta)d\alpha d\beta \tag{28}$$

The last term (28) is readily recognized as the square of the mean, i.e.,

$$R_{y_4}(t,s) = \overline{y_n(t)}^2 = R_{\tilde{w}}^2(0)\,|G(0)|^2 \tag{29}$$

The remaining terms which constitute the covariance function of $y_n(t)$ require individual consideration.

First consider $R_{y_1}(t,s)$. A useful change of variables is

$$\nu = t - \alpha$$

$$\mu = s - \beta$$

The double integral for $R_{y_1}(t,s)$ then becomes

$$R_{y_1}(t,s) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{p}(t-\nu)\tilde{p}^*(s-\mu)R_{\tilde{w}}(t-s+\mu-\nu)g(\nu)g(\mu)d\nu d\mu. \tag{30}$$

Now, recall that

$$R_{\tilde{w}}(t-s+\mu-\nu) = \int_{-\infty}^{\infty} S_{\tilde{w}}(f)e^{j2\pi f(t-s+\mu-\nu)}df. \tag{31}$$

Substitution yields, after some rearranging,

$$R_{y_1}(t,s) = \int_{-\infty}^{\infty} S_{\tilde{w}}(f) \int_{-\infty}^{\infty} \tilde{p}(t-\nu)g(\nu)e^{-j2\pi f\nu}d\nu \tag{32}$$

$$\cdot \int_{-\infty}^{\infty} \tilde{p}^*(s-\mu)g(\mu)e^{j2\pi f\mu}d\mu\; e^{j2\pi f(t-s)}df.$$

A form more amenable to numerical solution is obtained by again using the Fourier series expansion for $\tilde{p}(t)$. Recall that

$$\tilde{p}(\nu) = \sum_n p_n e^{j\omega_n\nu}$$

Thus it follows that

$$\tilde{p}(t-\nu) = \sum_n p_n e^{j\omega_n(t-\nu)}$$

or equivalently,

$$\tilde{p}(t-\nu) = \sum_n p_n e^{-j\omega_n \nu} e^{j\omega_n t} \quad . \tag{33}$$

This series may be substituted in (32) to eliminate the integral with respect to ν. The steps are as follows:

$$\int_{-\infty}^{\infty} \tilde{p}(t-\nu) g(\nu) e^{-j2\pi f\nu} d\nu$$

$$= \int_{-\infty}^{\infty} \left| \sum_n p_n e^{-j\omega_n \nu} e^{j\omega_n t} \right| g(\nu)\, e^{-j2\pi f\nu} d\nu$$

$$= \sum_n p_n e^{j\omega_n t} \int_{-\infty}^{\infty} g(\nu)\, e^{-j2\pi(f+f_n)\nu} d\nu$$

$$= \sum_n p_n\, G(f+f_n) e^{j\omega_n t} \quad . \tag{34}$$

In a similar way, the integral with respect to μ reduces to

$$\int_{-\infty}^{\infty} \tilde{p}^*(s-\mu) g(\mu) e^{j2\pi f\mu} d\mu$$

$$= \sum_m p_m^*\, G^*(f+f_m)\, e^{-j\omega_m s} \tag{35}$$

Substitution of these series forms into (32) and changing the order of integration and summation yields $R_{y_1}(t,s)$ as

$$R_{y_1}(t,s) = \sum_n \sum_m p_n p_m^* e^{j\omega_n t} e^{-j\omega_n s} \int_{-\infty}^{\infty} S_w(f)\, G(f+f_n) \cdot G^*(f+f_m) e^{j2\pi f(t-s)} df \quad . \tag{36}$$

Since $R_{y_2}(t,s) = R_{y_1}^*(t,s)$, only $R_{y_3}(t,s)$ remains to be considered. Let the Fourier transform of $R_w^2(\alpha-\beta)$ be denoted as $F(f)$, then

$$R_w^2(\alpha-\beta) = \int_{-\infty}^{\infty} F(f)\, e^{j2\pi f(\alpha-\beta)} df \tag{37}$$

where

$$F(f) = \int_{-\infty}^{\infty} S_{w}(f')S_{w}(f-f')df' \quad . \tag{38}$$

Substitution of (37) into (27) results in

$$R_{y_3}(t,s) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(f)g(t-\alpha)g(s-\beta)e^{j2\pi f(\alpha-\beta)}dfd\alpha d\beta. \tag{39}$$

Making the change of variables

$$\nu = t - \alpha$$

$$\mu = s - \beta$$

and rearranging slightly yields

$$R_{y_3}(t,s) = \int_{-\infty}^{\infty} F(f) \int_{-\infty}^{\infty} g(\nu)\ e^{-j2\pi f\nu}d\nu \int_{-\infty}^{\infty} g(\mu)\ e^{j2\pi f\mu}\ d\mu$$

$$\cdot\ e^{j2\pi f(t-s)}df \tag{40}$$

The integrals with respect to ν and μ may now be replaced by the appropriate transforms to give

$$R_{y_3}(t,s) = \int_{-\infty}^{\infty} F(f)\ |G(f)|^2\ e^{j2\pi f(t-s)}df \tag{41}$$

Finally, the correlation function of the output noise, $y_n(t)$, may be summarized as

$$R_{y_n}(t,s) = 2Re\{R_{y_1}(t,s)\} + R_{y_3}(t,s) + R_{y_4}(t,s)$$

$$= 2Re\left\{\sum_n\sum_m p_n p_m^*\ e^{j\omega_n t}\ e^{-j\omega_n s}\int_{-\infty}^{\infty} S_{w}(f)G(f+f_n)G^*(f+f_m)\right.$$

$$\left.\cdot\ e^{j2\pi f(t-s)}df\right\}$$

$$+ \int_{-\infty}^{\infty} F(f)\ |G(f)|^2\ e^{j2\pi f(t-s)}df + R_{w}^2(0)\ |G(0)|^2 \tag{42}$$

The variance $\sigma^2(t)$ of a sample at time t of the output noise process is of major interest. It is readily found as

$$\sigma^2(t) = R_{y_n}(t,t) - R_{\tilde{w}}^2(0)\ |G(0)|^2$$

or,

$$\sigma^2(t) = 2\mathrm{Re}\left\{\sum_n \sum_m p_n p_m^* e^{j(\omega_n-\omega_m)t} \int_{-\infty}^{\infty} S_{\tilde{w}}(f)G(f+f_n)G^*(f+f_m)df\right\}$$

$$+ \int_{-\infty}^{\infty} F(f)\ |G(f)|^2\ df\ . \qquad (43)$$

APPENDIX B

```
      PROGRAM TOAANAL
C*********************************************************************
C
C
C     TOAANAL -> TIME-OF-ARRIVAL ANALYSIS PROGRAM
C
C     FORTRAN77 SOURSE FILENAME:              TOAANAL.EXE
C
C     DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C     REVISION          DATE                  PROGRAMMER
C     --------          ----                  ----------
C
C       1.0             11-22-84              LONNIE HADEN
C
C*********************************************************************
C


      CHARACTER*17 ASIGNAL, ASLOPE, ASIGMAS, DSIGNAL, DSLOPE,
     &             DSIGMAS
      LOGICAL TOA1
      COMMON /ASS/ TOA1
      EXTERNAL  PULSE, ERROR4, PULSE1, ERROR41
      CHARACTER NOTHING
      CALL PRNT
      PRINT *, ' '
      PRINT *,    '******************  WARNING  **********************'
      PRINT *,' '
      PRINT*,'DO TO THE NATURE OF THIS PROGRAMS RUN-TIME ENVIRONMENT'
      PRINT *,'NO CORRECTIONS CAN BE MADE TO DATA AFTER IT HAS BEEN '
      PRINT *,'TYPED IN. '
      PRINT *,'DON''T PANIC! YOU CAN MAKE CORRECTIONS LATER IN THE'
      PRINT *,'PROGRAM.'
      PRINT *,' '
      PRINT *,' '
      PRINT *,' HIT RETURN TO CONTINUE '
      READ91, NOTHING
   91 FORMAT(A1)
    5 CALL PRNT
      PRINT *,'   WHAT WOULD YOU LIKE TO DO ?'
      PRINT *,' '
      PRINT *,' 1       TOA ANALYSIS ?'
      PRINT *,' 2       TOD ANALYSIS ?'
      PRINT *,' 3       PULSEWIDTH ANALYSIS ?'
      PRINT *,' 4       SIGNAL POWER BASED ON PULSEWIDTH ERROR ?'
      PRINT *,' 5       EXIT THE PROGRAM ?'
      PRINT *,' '
      PRINT *,' '
      PRINT *,' ENTER 1-5  '
      READ *,NTYPE
      CALL PRNT
      IF(NTYPE.EQ.1) THEN
        TOA1 = .TRUE.
      ELSE
        TOA1 = .FALSE.
      ENDIF
      GO TO (10,10,30,40,50) NTYPE
   10 CALL TOADATA(TOA1)
      GO TO 5
   30 PRINT *,' '
      PRINT *,'       WHICH WOULD YOU LIKE TO PERFORM ?'
      PRINT *,' '
      PRINT *,'    1  PULSEWIDTH ANALYSIS (ONE POST-FILTER)'
      PRINT *,'    2  PULSEWIDTH ANALYSIS (TWO POST-FILTERS)'
      PRINT *,' '
```

```
        PRINT *,' '
        PRINT *,' ENTER (1) OR (2).  '
        READ *, NPULSE
        IF(NPULSE.EQ.1) CALL PULSE1
        IF(NPULSE.EQ.2) CALL PULSE
        GO TO 5
 40     PRINT *,'       SIGNAL POWER BASED ON PULSEWIDTH ERROR FOR ?'
        PRINT *,' '
        PRINT *,'                1  (ONE POST-FILTER)'
        PRINT *,'                2  (TWO POST-FILTERS)'
        PRINT *,' '
        PRINT *,' '
        READ *, NERROR
        IF(NERROR.EQ.1) CALL ERROR41
        IF(NERROR.EQ.2) CALL ERROR4
        GO TO 5
 50     END
C
C
C       MAIN PROGRAM END
C
C
      SUBROUTINE TOADATA(TOA1)
C*****************************************************************************
C
C     SUBROUTINE TOADATA
C
C     SQUARE LAW DETECTOR -> RAW DATA CALCULATIONS
C
C     DEPARTMENT OF ELECTRICAL ENGINEERING      KANSAS STATE UNIVERSITY
C
C     REVISION              DATE                PROGRAMMER
C     --------              ----                ----------
C        2.0                4-30-82             F. W. RATCLIFFE
C        2.1               11-22-83             MARK HERMAN
C
C     PC REVISION           DATE                PROGRAMMER
C     -----------           ----                ----------
C        5.0               11-22-85             LONNIE HADEN
C
C*****************************************************************************
C
      COMPLEX  BUTTER1, EULER, G, G1, G2, HWIG, INTEGRATOR,
     &         PWIG(0:511), REDUCE, SAW, SWIG(0:511),
     &         TAPPED, TERM(0:511), VALUE, GWIG(0:511),F12
      CHARACTER*17 GQNAME, NAME1, NAME2, NAME3, NAME4, NAME5, NAME6
      CHARACTER    NOTHING
      INTEGER  OFFSET, POINTER
      LOGICAL  EVEN, TOA, TOA1, TOA12, CK
      REAL     ABSCISSA(-46:46), GQDATA(186), GTERM(-46:46),
     &         GTIME(0:511), RESULT(0:512), SUM, SWDATA(0:256),
     &         TAP1(0:32), TAP2(0:32), SWTERM(-46:46),
     &         WEIGHT(-46:46), DOUBLE
      COMMON /TODJOB/ PWIG, SWIG, TERM, RESULT, SWDATA, GQDATA
     &       /HWIG1/  NPIN, FIN, /G11/ NPOUT1, FOUT1, /G21/ NPOUT2,
     &                FOUT2, /   / REDUCE, DELAY
C
C*****************************************************************************
C
C                          INITIALIZATION
C
      TOA12 = TOA1
      A = 1.0
      DELAY = 0.0
      EVEN = .FALSE.
      GQNAME = 'C:GQDATA.DAT'
```

```
      LENGTH = 184
      FOUT2 = 1.0
      ICHAN = 1
      NPOUT2 = 1
      NWEIGHT2 = 0
      PI = 3.141593
      REDUCE = CMPLX(0.0,0.0)
      TAP2(0) = 1.0
      TPOST2 = 1.0
      TSPACE2 = 1.0
   83 PRINT *,' '
      IF (TOA1) THEN
        PRINT *,  '  SQUARE LAW DETECTOR -> TOA RAW DATA CALCULATIONS '
      ELSE
        PRINT *,  '  SQUARE LAW DETECTOR -> TOD RAW DATA CALCULATINGS '
      ENDIF
      PRINT *, ' '
C
C*********************************************************************
C
C                  INPUT INFORMATION FROM CONSOLE
C
      PRINT *, ' '
      PRINT *, 'BUTTERWORTH PRE-FILTER'
      PRINT *, ' '
      PRINT *, '          RF BANDWIDTH (Hz) ?  '
      READ *, RF
      PRINT *, '          NUMBER OF POLES (1-4) ?  '
      READ *, NPIN
      PRINT *, ' '
      PRINT *, ' '
      CALL REPLY ('  TWO POST-FILTERS          ?  ', TOA)
      PRINT *, ' '
      IF (TOA)  PRINT *, 'BUTTERWORTH POST-FILTER -> DELAY PATH'
      IF (.NOT. TOA)  PRINT *, 'BUTTERWORTH POST-FILTER'
      IF (TOA)  PRINT *, ' '
      IF (TOA)  PRINT *, '          DELAY TIME (s)  '
      IF (TOA)  READ *, DELAY
      PRINT *, ' '
      PRINT *, '          CUTOFF FREQUENCY (Hz) ?  '
      READ *, FOUT1
      PRINT *, '          NUMBER OF POLES (1-4) ?  '
      READ *, NPOUT1
      IF (.NOT. TOA)  GO TO 240
      PRINT *, ' '
      PRINT *, ' BUTTERWORTH POST-FILTER -> ATTENUATION + DELAY PATH'
      PRINT *, ' '
      PRINT *, '          ATTENUATION (dB) ?  '
      READ *, REDUCEDB
      PRINT *, ' '
      PRINT *, '          CUTOFF FREQUENCY (Hz) ?  '
      READ *, FOUT2
      PRINT *, '          NUMBER OF POLES (1-4) ?  '
      READ *, NPOUT2
  240 CONTINUE
C
      PRINT *, ' '
      PRINT *, ' '
      PRINT *, 'INPUT PULSE'
      PRINT *, ' '
      PRINT *, '          WIDTH (s) ?  '
      READ *, TAU
      PRINT *, '          RISE TIME (s) ?  '
      READ *, TRISE
      PRINT *, '          OFFSET FROM CENTER (Hz) ?  '
      READ *, DELTAF
```

```
      PRINT *, ' '
      PRINT *, ' '
      PRINT *, 'MISCELLANEOUS'
      PRINT *, ' '
      PRINT *, '          FUNDAMENTAL FREQUENCY (Hz) ?  '
      READ *, F0
      PRINT *, '          NUMBER OF SAMPLES (128,256,512) ?  '
      READ *, NPOINT
      CALL PRNT
      PRINT *,'  CHECK FOR VALID DATA ! '
      PRINT *,' '
      PRINT *,'       BUTTERWORTH PRE-FILTER'
      PRINT *,' '
      PRINT *,'       RF BANDWIDTH (Hz)   ', RF
      PRINT *,'       NUMBER OF POLES     ', NPIN
      PRINT *,' '
      PRINT *,'       TWO POST-FILTERS    ', TOA
      PRINT *,' '
      IF(TOA) PRINT *,'       BUTTERWORTH POST-FILTER + DELAY PATH'
      IF(.NOT.TOA) PRINT *,'       BUTTERWORTH POST-FILTER'
      IF(.NOT.TOA) PRINT *,' '
      IF(TOA) PRINT *,' '
      IF(TOA) PRINT *,'       DELAY TIME     ', DELAY
      PRINT *,'       CUTOFF FREQ (Hz)    ', FOUT1
      PRINT *,'       NUMBER OF POLES     ', NPOUT1
      PRINT *,' '
      PRINT *,'  HIT RETURN TO CONTINUE'
      READ97, NOTHING
   97 FORMAT(A1)
      IF(.NOT.TOA) GO TO 241
        PRINT *,' '
        PRINT *,'       BUTTERWORTH POST-FILTER + ATTENUATION AND DELAY'
        PRINT *,' '
        PRINT *,'       ATTENUATION (dB)    ', REDUCEDB
        PRINT *,'       CUTOFF FREQ (Hz)    ', FOUT2
        PRINT *,'       NUMBER OF POLES     ', NPOUT2
  241 PRINT *,' '
      PRINT *,'       INPUT PULSE'
      PRINT *,' '
      PRINT *,'       WIDTH (s)           ', TAU
      PRINT *,'       RISE TIME (s)       ', TRISE
      PRINT *,'       OFFSET FROM CENTER (Hz) ', DELTAF
      PRINT *,' '
      PRINT *,'       MISCELLANEOUS'
      PRINT *,' '
      PRINT *,'       FUNDAMENTAL FREQUENCY (Hz)  ', F0
      PRINT *,'       NUMBER OF POINTS  ', NPOINT
      PRINT *,' '
      CALL REPLY(' DO YOU WISH TO CORRECT DATA ? ', CK)
      IF(CK) GO TO 83
      PRINT *,' '
      PRINT *,' '
      IF (TOA1) THEN
        CALL WNAME ('   FILE TO CONTAIN TOA SIGNAL ?   ', NAME1)
        CALL WNAME ('   FILE TO CONTAIN TOA SLOPE  ?   ', NAME2)
        CALL WNAME ('   FILE TO CONTAIN TOA SIGMAS ?   ', NAME3)
      ELSE
        CALL WNAME ('   FILE TO CONTAIN TOD SIGNAL ?   ', NAME4)
        CALL WNAME ('   FILE TO CONTAIN TOD SLOPE  ?   ', NAME5)
        CALL WNAME ('   FILE TO CONTAIN TOD SIGMAS ?   ', NAME6)
      ENDIF
   82 CALL PRNT
      PRINT *,'  CHECK FOR VALID FILE NAMES ! '
      PRINT *,' '
      IF (TOA1) THEN
        PRINT *,'    (1)    TOA SIGNAL FILE IS : ', NAME1
```

```
        PRINT *,'     (2)     TOA SLOPE  FILE IS : ', NAME2
        PRINT *,'     (3)     TOA SIGMAS FILE IS : ', NAME3
      ELSE
        PRINT *,'     (4)     TOD SIGNAL FILE IS : ', NAME4
        PRINT *,'     (5)     TOD SLOPE  FILE IS : ', NAME5
        PRINT *,'     (6)     TOD SIGMAS FILE IS : ', NAME6
      ENDIF
      CALL REPLY('  DO YOU WISH TO CORRECT ?    ', CK)
      IF(.NOT.CK) GO TO 81
      PRINT *,' '
      PRINT *,'  ENTER THE NUMBER OF THE INCORRECT FILE NAME .  '
      READ *, NCK
      GO TO (2,3,4,7,8,9) NCK
  2   PRINT *,'  NEW TOA SIGNAL FILE ? '
      READ19, NAME1
      GO TO 82
  3   PRINT *,'  NEW TOA SLOPE FILE ? '
      READ19, NAME2
      GO TO 82
  4   PRINT *,'  NEW TOA SIGMAS FILE ? '
      READ19, NAME3
      GO TO 82
  7   PRINT *,'  NEW TOD SIGNAL FILE ? '
      READ19, NAME4
      GO TO 82
  8   PRINT *,'  NEW TOD SLOPE FILE ? '
      READ19, NAME5
      GO TO 82
  9   PRINT *,'  NEW TOD SIGMAS FILE ? '
      READ19, NAME6
      GO TO 82
 19   FORMAT(A17)
C
C*******************************************************************************
C
C     CALCULATE PARAMETERS DERIVED FROM CONSOLE INPUT
C
 81   FIN = RF * 0.5
      FPRE = 8.0 * FIN
      FPOST = 16.0 * FOUT1
      KLIMIT = LENGTH / 4
      IF (KLIMIT * 4 .EQ. LENGTH)  EVEN = .TRUE.
      LAST = NPOINT - 1
      LIMIT = NPOINT / 2 - 1
      PERIOD = 1.0 / F0
      IF (TOA)  REDUCE =  CMPLX((10.0 ** (REDUCEDB / 20.0)) ** 2, 0.0)
      STEPS = REAL(NPOINT - 1)
C
C*******************************************************************************
C
C     CALCULATE LOW-PASS EQUIVALENT OF INPUT SIGNAL
C
           DO 400 LOOP = 0, LAST
           T = (PERIOD * LOOP) / STEPS
           SWIG(LOOP) = EULER(W(DELTAF) * T)
           TEMP = 0.0
           IF (T .LE. TRISE)  TEMP = (A * T) / TRISE
           IF (T .GT. TRISE .AND. T .LT. TAU) TEMP = A
           IF ( T .GE. TAU .AND. T .LE. TRISE + TAU)
     &                  TEMP = A - (A * (T - TAU)) / TRISE
           SWIG(LOOP) = SWIG(LOOP) * CMPLX(TEMP, 0.0)
 400       CONTINUE
C
C*******************************************************************************
C
C     CALCULATE LOW-PASS EQUIVALENT OF SIGNAL AT PRE-FILTER OUTPUT
```

```
C     PLEASE NOTE -> SWIG IS NOT SYMMETRIC ABOUT LIMIT + 1
C
      CALL CFFT (SWIG(0), NPOINT, 0)
C
      PWIG(0) = SWIG(0)
            DO 420 LOOP = 1, LIMIT
            LOCATE = NPOINT - LOOP
            VALUE = HWIG (F0 * LOOP)
            PWIG(LOOP) = SWIG(LOOP) * VALUE
            PWIG(LOCATE) = SWIG(LOCATE) *CONJG(VALUE)
  420       CONTINUE
C
      VALUE = HWIG(F0 * (LIMIT + 1))
      PWIG(LIMIT + 1) = SWIG(LIMIT + 1) * (VALUE)
C
      CALL CFFT (PWIG(0), NPOINT, 1)
C
C*****************************************************************************
C
C     CALCULATE SIGNAL AT POST-FILTER OUTPUT
C
            DO 440 LOOP = 0, LAST
            TERM(LOOP) = PWIG(LOOP) * CONJG(PWIG(LOOP))
  440       CONTINUE
C
      CALL CFFT (TERM(0), NPOINT, 0)
C
      TERM(0) = TERM(0) * G(0.0)
            DO 460 LOOP = 1, LIMIT
            LOCATE = NPOINT - LOOP
            VALUE = G(F0 * LOOP)
            TERM(LOOP) = TERM(LOOP) * VALUE
            TERM(LOCATE) = TERM(LOCATE) * CONJG(VALUE)
  460       CONTINUE
C
       VALUE = G(F0 * (LIMIT + 1))
       TERM(LIMIT + 1) = TERM(LIMIT + 1) * VALUE
C
       CALL CFFT (TERM(0), NPOINT, 1)
C
            DO 480 LOOP = 0, LAST
            RESULT(LOOP) = REAL(TERM(LOOP))
  480       CONTINUE
C
C     STORE F0 FOR USE IN SUBROUTINE PULSE
C
      RESULT(NPOINT) = F0
C
C
      IF (TOA1) THEN
        CALL WDATA (NPOINT+1, RESULT(0), NAME1)
      ELSE
        CALL WDATA (NPOINT, RESULT(0), NAME4)
      ENDIF
C
C*****************************************************************************
C
C     CALCULATE SLOPE OF SIGNAL AT POST-FILTER OUTPUT
C
            DO 500 LOOP = 0, LAST
            TERM(LOOP) = PWIG(LOOP) * CONJG(PWIG(LOOP))
  500       CONTINUE
C
      CALL CFFT (TERM(0), NPOINT, 0)
C
      TERM(0) = TERM(0) * G(0.0)
```

```
            DO 520 LOOP = 1, LIMIT
            F = F0 * LOOP
            LOCATE = NPOINT - LOOP
            F12 = G(F)
            VALUE = F12 * CMPLX(0.0, W(F))
            TERM(LOOP) = TERM(LOOP) * VALUE
            TERM(LOCATE) = TERM(LOCATE) * CONJG(VALUE)
  520       CONTINUE
C
      F = F0 * (LIMIT + 1)
      F12 = G(F)
      VALUE = F12 * CMPLX(0.0, W(F))
      TERM(LIMIT + 1) = TERM(LIMIT + 1) * VALUE
C
      CALL CFFT (TERM(0), NPOINT, 1)
C
           DO 540 LOOP = 0, LAST
           RESULT(LOOP) = REAL(TERM(LOOP))
 540       CONTINUE
C
      IF (TOA1) THEN
         CALL WDATA (NPOINT, RESULT(0), NAME2)
      ELSE
         CALL WDATA (NPOINT, RESULT(0), NAME5)
      ENDIF
C
C*********************************************************************
C
C     CALCULATE INPULSE RESPONSE OF POST-FILTER
C
      TERM(0) = G(0.0)
           DO 560 LOOP = 1, LIMIT
           TERM(LOOP) = G(F0 * LOOP)
           TERM(NPOINT - LOOP) = CONJG(TERM(LOOP))
  560      CONTINUE
C
      TERM(LIMIT + 1) = G(F0 * (LIMIT + 1))
C
      CALL CFFT (TERM(0), NPOINT, 1)
C
           DO 580 LOOP = 0, LAST
           GTIME(LOOP) = REAL(TERM(LOOP))
  580      CONTINUE
C
C*********************************************************************
C
C     PRECALCULATE COMPONENTS OF SINGLE INTEGRAL
C
           DO 600 LOOP = 0, LIMIT + 1
           SWDATA(LOOP) = SW(F0 * REAL(LOOP))
  600      CONTINUE
C
C*********************************************************************
C
C     EVALUTE SINGLE INTERGRAL
C
           DO 660 LOOP1 = 0, LAST
              DO 620 LOOP2 = 0, LAST
              LOCATE = LOOP1 - LOOP2
              IF (LOCATE . LT. 0)  LOCATE = LOCATE + NPOINT
              TERM(LOOP2) = PWIG(LOCATE) *
     &                      CMPLX(GTIME(LOOP2), 0.0)
  620         CONTINUE
C
           CALL CFFT (TERM(0), NPOINT, 0)
C
```

```
                SUM = SWDATA(0) * SQUARED(TERM(0))
                      DO 640 LOOP2 = 1, LIMIT
                      SUM = SUM + (SQUARED (TERM(NPOINT - LOOP2)) +
     &                      SQUARED(TERM(LOOP2))) * SWDATA(LOOP2)
  640                 CONTINUE
C
                SUM = SUM + SWDATA(LIMIT + 1) * SQUARED(TERM(LIMIT + 1))
                RESULT(LOOP1) = 2.0 * F0 * SUM
  660           CONTINUE
C
C*******************************************************************************
C
C       READ AND UNPACK GAUSS QUADRATURE DATA
C
        CALL RDATA (LENGTH, GQDATA, GQNAME)
        OFFSET = KLIMIT * 2
        IF (.NOT. EVEN)  OFFSET = OFFSET + 1
        POINTER = OFFSET
                DO 680 LOOP = -KLIMIT, KLIMIT
                IF (EVEN .AND. LOOP .EQ. 0)  GO TO 680
                ABSCISSA(LOOP) = GQDATA(POINTER + OFFSET)
                WEIGHT(LOOP) = GQDATA(POINTER)
                POINTER = POINTER - 1
  680           CONTINUE
C
C*******************************************************************************
C
C       PRECALCULATE COMPONENTS OF DOUBLE INTEGRAL
C
                DO 700 LOOP = -KLIMIT, KLIMIT
                IF (EVEN .AND. LOOP .EQ. 0)  GO TO 700
                F12 = G(FPOST * ABSCISSA(LOOP))
                F13 = SW(FPRE * ABSCISSA(LOOP))
                GTERM(LOOP) = WEIGHT(LOOP) * SQUARED(F12)
                SWTERM(LOOP) = WEIGHT(LOOP) * F13
  700           CONTINUE
C
C*******************************************************************************
C
C       EVALUATE DOUBLE INTEGRAL
C
        DOUBLE1 = 0.0
                DO 740 LOOP1 = -KLIMIT, KLIMIT
                IF (EVEN .AND. LOOP1 .EQ. 0)  GO TO 740
                F1 = FPOST * ABSCISSA(LOOP1)
                        DO 720 LOOP2 = -KLIMIT, KLIMIT
                        IF (EVEN .AND. LOOP2 .EQ. 0)  GO TO 720
                        F2 = FPRE * ABSCISSA(LOOP2)
                        DOUBLE1 = DOUBLE1 + SWTERM(LOOP2) * SW(F1 - F2)
     &                            * GTERM(LOOP1)
  720                   CONTINUE
  740           CONTINUE
C
        DOUBLE1 = FPRE * FPOST * DOUBLE1
        RESULT(NPOINT) = DOUBLE1
        IF (TOA1) THEN
          CALL WDATA (NPOINT + 1, RESULT(0), NAME3)
        ELSE
          CALL WDATA (NPOINT + 1, RESULT(0), NAME6)
        ENDIF
C
C*******************************************************************************
C
C       DOCUMENT RUN
C
        OPEN(UNIT = 4, FILE = 'LST:', FORM = 'FORMATTED')
```

```
      IF (TOA1) THEN
        WRITE(4,*)' '
        WRITE(4,*)'SQUARE LAW DETECTOR -> TOA RAW DATA CALCULATIONS'
      ELSE
        WRITE(4,*)' '
        WRITE(4,*)'SQUARE LAW DETECTOR -> TOD RAW DATA CALCULATIONS'
      ENDIF
      WRITE(4,*) ' '
      WRITE(4,*) '      BUTTERWORTH PRE-FILTER      '
      WRITE(4,*) ' '
C     WRITE(4,10) '      INTEGRATION PERIOD      : ', TPRE , ' s '
   10 FORMAT(A30, G12.6, A3)
      WRITE(4,10) '      RF BANDWIDTH            : ', RF, ' Hz'
      WRITE(4,20) '      NUMBER OF POLES         : ', NPIN
   20 FORMAT(A30, I1)
      WRITE(4,*) ' '
      IF(TOA)WRITE(4,*)'      BUTTERWORTH POST-FILTER -> DELAY PATH '
      IF (.NOT. TOA)  WRITE(4,*) '      BUTTERWORTH POST-FILTER '
      WRITE(4,*) ' '
      IF(TOA)WRITE(4,10) '      DELAY TIME                : ', DELAY, ' s '
      WRITE(4,10) '      CUTOFF FREQUENCY        : ', FOUT1, ' Hz'
      WRITE(4,20) '      NUMBER OF POLES         : ', NPOUT1
      IF (.NOT. TOA)  GO TO 940
      WRITE(4,*) ' '
      WRITE(4,*) 'BUTTERWORTH POST-FILTER -> ATTENUATION + DELAY PATH'
      WRITE(4,*) ' '
      WRITE(4,10) '      ATTENUATION             : ', REDUCEDB, ' dB'
      WRITE(4,10) '      CUTOFF FREQUENCY        : ', FOUT2, ' Hz'
      WRITE(4,20) '      NUMBER OF POLES         : ', NPOUT2
  940 CONTINUE
      WRITE(4,*) ' '
      WRITE(4,*) '  INPUT PULSE  '
      WRITE(4,*) ' '
      WRITE(4,10) '      WIDTH                   : ', TAU, ' s '
      WRITE(4,10) '      RISE TIME               : ', TRISE, ' s '
      WRITE(4,10) '      OFFSET FROM CARRIER     : ', DELTAF, ' Hz'
      WRITE(4,*) ' '
      WRITE(4,*) '  MISCELLANEOUS  '
      WRITE(4,*) ' '
      WRITE(4,10) '      FUNDAMENTAL FREQUENCY   : ', F0, ' Hz'
      WRITE(4,40) '      NUMBER OF SAMPLES       : ', NPOINT
   40 FORMAT(A30, I5)
      WRITE(4,*) ' '
      IF (TOA1) THEN
        WRITE(4,50) '       TOA SIGNAL STORED IN FILE:  ', NAME1
   50   FORMAT(A33, A23)
        WRITE(4,50) '       TOA SLOPE STORED IN FILE :  ', NAME2
        WRITE(4,50) '       TOA SIGMAS STORED IN FILE:  ', NAME3
      ELSE
        WRITE(4,50) '       TOD SIGNAL STORED IN FILE:  ', NAME4
        WRITE(4,50) '       TOA SLOPE STORED IN FILE :  ', NAME5
        WRITE(4,50) '       TOA SIGMAS STORED IN FILE:  ', NAME6
      ENDIF
      CLOSE(4)
      END
C
C
C
      COMPLEX FUNCTION BUTTER1 (N, FNORM1)
      COMPLEX SS
      SS = CMPLX(0.0, FNORM1)
      S2 = -FNORM1 * FNORM1
      GO TO (11, 21, 31, 41) N
   11 BUTTER1 = CMPLX(1.0 / (SS + 1.0))
      GO TO 51
   21 BUTTER1 = CMPLX(1.0 / (S2 + 1.414 * SS + 1.0))
```

```
      GO TO 51
   31 BUTTER1 = CMPLX(1.0 /((SS + 1.0) * (S2 +SS +1)))
      GO TO 51
   41 BUTTER1 = CMPLX(1.0 / ((S2 + .765 * SS + 1.0)
     &          * (S2 + 1.848 * SS + 1.0)))
   51 RETURN
      END
C
C
C
C
C*******************************************************************************
C
C     DEFINE EULER'S IDENITY
C
      COMPLEX FUNCTION EULER(ANGLE)
      EULER = CMPLX(COS(ANGLE), SIN(ANGLE))
      END
C
C*******************************************************************************
C
C     DEFINE MAGNITUDE SQUARED OF A COMPLEX NUMBER
C
      REAL FUNCTION  SQUARED(VALUE)
      COMPLEX VALUE
      SQUARED = (VALUE * CONJG(VALUE))
      END
C
C*******************************************************************************
C
C     DEFINE ANGULAR FREQUENCY
C
      REAL FUNCTION  W(F)
      PI = 3.141593
      W = 2.0 * PI * F
      END
C
C*******************************************************************************
C
C     DEFINE FREQUENCY RESPONSE OF POST-FILTER -> DELAY PATH
C
      COMPLEX FUNCTION G1(F)
      COMPLEX BUTTER1
      COMMON /G11/ NPOUT1, FOUT1
      G1 = BUTTER1 (NPOUT1, F / FOUT1)
      END
C
C
C*******************************************************************************
C
C     DEFINE FREQUENCY RESPONSE OF POST-FILTER -> ATTENUATION + DELAY PATH
C
      COMPLEX FUNCTION G2(F)
      COMPLEX BUTTER1
      COMMON /G21/ NPOUT2, FOUT2
      G2 = BUTTER1 (NPOUT2, F / FOUT2)
      END
C
C
C*******************************************************************************
C
C     DEFINE FREQUENCY RESPONSE OF COMBINED POST-FILTER
C
      COMPLEX FUNCTION G(F)
      REAL W
      COMPLEX REDUCE, G1, G2, EULER, WW1, WW2
```

```
      COMPLEX G1RES, G2RES, EURES, EURES1
      LOGICAL TOA1
      COMMON /   / REDUCE, DELAY, /ASS/ TOA1
      IF (TOA1) THEN
        IF(F.EQ.0.0) THEN
          G = CMPLX(1.0 - REAL(REDUCE),  0.0)
        ELSE
          G1RES = G1(F)
          EURES = EULER(-W(F)*DELAY)
          WW1 = G1RES * EURES
          G2RES = G2(F)
          WW2 = G2RES * REDUCE
          G = WW1 - WW2
        ENDIF
      ELSE
       IF(F.EQ.0.0) THEN
        G = CMPLX(REAL(REDUCE) - 1.0, 0.0)
       ELSE
        G1RES = G1(F)
        G2RES = G2(F)
        EURES = EULER(-W(F) * 2.0 * DELAY)
        EURES1 = EULER(-W(F) * DELAY)
        G = G2RES * EURES * REDUCE - G1RES * EURES1
       ENDIF
      ENDIF
      END
C
C*************************************************************************
C
C     DEFINE FREQUENCY RESPONSE OF PRE-FILTER
C
      COMPLEX FUNCTION HWIG(F)
      COMPLEX BUTTER1
      COMMON /HWIG1/ NPIN, FIN
      HWIG = BUTTER1 (NPIN, F / FIN)
      END
C
C
C*************************************************************************
C
C     DEFINE NOISE DENSITY AT PRE-FILTER OUTPUT
C
      REAL FUNCTION SW(F)
      COMPLEX HWIG
      SW = 2.0 * SQUARED(HWIG(F))
      END
C
C
C     MAIN PROGRAM END !!!!!!!!!
C
C
      SUBROUTINE CFFT (X, N, INV)
      COMPLEX W1, T1, X(1:512)
C
C     CALCULATE THE NUMBER OF ITERATIONS (LOG. N TO THE BASE 2)
C
      ITER = 0
      IREM = N
   12 IREM =IREM/2
      IF (IREM .EQ. 0) GO TO 22
      ITER = ITER + 1
      GO TO 12
   22 CONTINUE
      SIGN = -1.0
      IF (INV .EQ. 1) SIGN = 1.0
      NXP2 = N
```

```
      DO 52 IT = 1, ITER
C
C     COMPUTATION FOR EACH ITERATION
C     NXP: NUMBER OF POINTS IN A PARTITION
C     NXP2: NXP/2
C
      NXP = NXP2
      NXP2 = NXP / 2
      WPWR = 3.141592 / NXP2
C
C     CALCULATE THE MULTIPLIER
C
      DO 42 M = 1, NXP2
      ARG = REAL(M - 1) * WPWR
      W1 = CMPLX(COS(ARG), SIGN * SIN(ARG))
      DO 42 MXP = NXP, N, NXP
C
C     COMPUTATION FOR EACH PARTITION
C
      J1 = MXP - NXP + M
      J2 = J1 + NXP2
      T1 = (X(J1) - X(J2))
      X(J1) = (X(J1) + X(J2))
   42 X(J2) = (T1 * W1)
   52 CONTINUE
C
C     UNSCRAMBLE THE BIT-REVERSED DFT COEFFS
C
      N2 = N / 2
      N1 = N - 1
      J = 1
      DO 65 I = 1, N1
      IF (I .GE. J) GO TO 55
      T1 = (X(J))
      X(J) = (X(I))
      X(I) = (T1)
   55 K = N2
   60 IF (K .GE. J) GO TO 65
      J = J - K
      K = K / 2
      GO TO 60
   65 J = J + K
      IF (INV . EQ. 1) GO TO 75
      DO 70 I = 1, N
   70 X(I) = (X(I) / N)
   75 CONTINUE
      RETURN
      END
C
C
C
C
C
C
C
      SUBROUTINE WNAME (ISTRING, NAME)
      CHARACTER ISTRING*33, NAME*17
      PRINT *,' '
      PRINT5, ISTRING
    5 FORMAT (A33)
      READ10, NAME
      PRINT *,' '
   10 FORMAT (A17)
      RETURN
      END
C
```

```
C
C
      SUBROUTINE RDATA (LENGTH, VAR, NAME)
      CHARACTER NAME*17, A*1
      REAL VAR(LENGTH+3)
      OPEN (UNIT = 1, FILE = NAME)
      DO 101 LOOP = 1, LENGTH
         READ(UNIT = 1, FMT = 23) VAR(LOOP)
  101    CONTINUE
   23 FORMAT(G16.10)
      CLOSE (UNIT = 1)
      RETURN
      END
C
C
C
      SUBROUTINE WDATA (LENGTH, VAR, NAME)
      CHARACTER  NAME*17, A*1
      REAL VAR(LENGTH)
      OPEN (UNIT = 1, FILE = NAME)
      DO 102 LOOP = 1, LENGTH
          WRITE(UNIT = 1, FMT = 24) VAR(LOOP)
  102 CONTINUE
    24 FORMAT(G16.10)
      CLOSE (UNIT = 1)
      RETURN
      END
C
C
C
       SUBROUTINE REPLY (HEADER, RESULT)
       CHARACTER HEADER*30, TEST*1
       LOGICAL RESULT
       RESULT = .TRUE.
       PRINT14, HEADER
   14  FORMAT(A30)
       PRINT *,'      ANSWER YES OR NO :  '
       READ26, TEST
   26  FORMAT(A1)
       IF (TEST .EQ. 'N' .OR. TEST .EQ. 'n') RESULT = .FALSE.
       RETURN
       END
C
C
C
       SUBROUTINE PRNT
       DO 6 I = 0, 22
         PRINT *,' '
    6  CONTINUE
       RETURN
       END
```

```
      SUBROUTINE PULSE
C*************************************************************************
C
C     SUBROUTINE PULSE
C
C     PULSE-WIDTH ESTIMATE -> TWO POST-FILTERS
C
C     DEPARTMENT OF ELECTRICAL ENGINEERING       KANSAS STATE UNIVERSITY
C
C     REVISION                DATE               PROGRAMMER
C     --------                ----               ----------
C       2.0                  4-30-82             F. W. RATCLIFFE
C       2.1                  6-18-82             DIANE VONTHAER
C       2.2                 11-22-83             MARK HERMAN
C
C     PC REVISION             DATE               PROGRAMMER
C     -----------             ----               ----------
C        5.0                11-22-85             LONNIE HADEN
C
C*************************************************************************
C
      INTEGER COUNT, PRIOR1, PRIOR2
      CHARACTER*17  NAME1, NAME2, NAME3, NAME4, NAME5, NAME6
      REAL     NO, SIGMAS1(0:512), SIGNAL1(0:512), SLOPE1(0:511),
     &         POSPEAK1, NEGPEAK1, SIGNAL2(0:511), SLOPE2(0:511),
     &         SIGMAS2(0:512), PEAK2, INTER1, INTER2, NEGPEAK2
      LOGICAL LOOK, MINUS, ERR, RESP, CK
C
C*************************************************************************
C
C     INITIALIZATION
C
      CALL PRNT
      A = 1.0
      PERCENT = 0.0
      RESP = .FALSE.
      ERR = .FALSE.
    4 IF (ERR) CALL REPLY(' DO YOU WISH TO EXIT ? ',RESP)
      IF (RESP) GO TO 3
      ERR = .FALSE.
        PRINT*,' '
        PRINT *, 'PULSE WIDTH ESTIMATE -> TWO POST-FILTERS'
        PRINT *,' '
C
C
      CALL RNAME1('     TOA SIGNAL FILE ?        ', NAME1)
      CALL RNAME1('     TOA SLOPE  FILE ?        ', NAME2)
      CALL RNAME1('     TOA SIGMAS FILE ?        ', NAME3)
      PRINT *,' '
C
      CALL RNAME1('     TOD SIGNAL FILE ?        ', NAME4)
      CALL RNAME1('     TOD SLOPE  FILE ?        ', NAME5)
      CALL RNAME1('     TOD SIGMAS FILE ?        ', NAME6)
   82 CALL PRNT
      PRINT *,'  CHECK FOR VALID FILE NAMES !'
      PRINT *,' '
      PRINT *,'     (1)   TOA SIGNAL FILE IS : ', NAME1
      PRINT *,'     (2)   TOA SLOPE FILE IS  : ', NAME2
      PRINT *,'     (3)   TOA SIGMAS FILE IS : ', NAME3
      PRINT *,'     (4)   TOD SIGNAL FILE IS : ', NAME4
      PRINT *,'     (5)   TOD SLOPE FILE IS  : ', NAME5
      PRINT *,'     (6)   TOD SIGMAS FILE IS : ', NAME6
      CALL REPLY('  DO YOU WISH TO CORRECT ?   ', CK)
      IF(.NOT.CK) GO TO 81
      PRINT *,' '
```

```
      PRINT *,' ENTER THE NUMBER OF THE INCORRECT FILE NAME.  '
      READ *, NCK
      PRINT *,' '
      GO TO (13,14,15,16,17,18) NCK
   13 PRINT *,' NEW TOA SIGNAL FILE ?  '
      READ19, NAME1
      GO TO 82
   14 PRINT *,' NEW TOA SLOPE FILE ?  '
      READ19, NAME2
      GO TO 82
   15 PRINT *,' NEW TOA SIGMAS FILE ?  '
      READ19, NAME3
      GO TO 82
   16 PRINT *,' NEW TOD SIGNAL FILE ?  '
      READ19, NAME4
      GO TO 82
   17 PRINT *,' NEW TOD SLOPE FILE ?  '
      READ19, NAME5
      GO TO 82
   18 PRINT *,' NEW TOD SIGMAS FILE ?  '
      READ19, NAME6
      GO TO 82
   19 FORMAT(A17)
C
C*****************************************************************************
C
C     INPUT DATA FROM EXTERNAL FILES
C
   81 CALL RDATA1(LENGTH1, SIGNAL1(0), NAME1)
      CALL RDATA1(LENGTH2, SLOPE1(0), NAME2)
      CALL RDATA1(LENGTH3, SIGMAS1(0), NAME3)
C
      CALL RDATA1(LENGTH4, SIGNAL2(0), NAME4)
      CALL RDATA1(LENGTH5, SLOPE2(0), NAME5)
      CALL RDATA1(LENGTH6, SIGMAS2(0), NAME6)
C
C
C
      NLENGTH=(LENGTH1-1)+LENGTH2+LENGTH3-LENGTH4-LENGTH5-LENGTH6
      IF(NLENGTH .NE. 0)THEN
        CALL PRNT
        PRINT *,'        *******   ERROR   *******'
        PRINT *,' '
        PRINT *,'     FILES ARE NOT THE SAME LENGTH !!'
        PRINT *,' '
        PRINT7,' FILE ',NAME1, 'HAS LENGTH', LENGTH1-1
        PRINT7,' FILE ',NAME2,' HAS LENGTH ',LENGTH2
        PRINT7,' FILE ',NAME3,' HAS LENGTH ',LENGTH3-1
        PRINT7,' FILE ',NAME4,' HAS LENGTH ',LENGTH4
        PRINT7,' FILE ',NAME5,' HAS LENGTH ',LENGTH5
        PRINT7,' FILE ',NAME6,' HAS LENGTH ',LENGTH6-1
    7   FORMAT(A6,A17,A12,I3)
        PRINT *,' '
        PRINT *,' '
        PRINT *,' '
        GO TO 4
      ELSE
        NPOINT1 = LENGTH2
        NPOINT2 = LENGTH2
      ENDIF
C
C*****************************************************************************
C
C     INPUT DATA FROM CONSOLE
C
      ICOUNT = 0
```

```
   2    IF(ICOUNT.GT.0) PRINT *,'ENTER -101 SNR TO EXIT '
        PRINT *,' '
        PRINT *, '      SIGNAL TO NOISE RATIO (dB)  ?  '
        READ *, SNR
        IF(SNR.EQ.-101) GO TO 3
        PRINT *, '      SNR REFERENCE BANDWIDTH (Hz)  ?  '
        READ *, BREF
        PRINT *,' '
        PRINT *,'       SEARCH FOR THRESHOLD AFTER OCCURRENCE OF'
        CALL REPLY('        NEGATIVE PEAK SIGNAL ?', MINUS)
        PRINT *,' '
C
C     PERIOD = 1.0 / F0
C
        F0 = SIGNAL1(NPOINT1)
        PERIOD = 1.0 / F0
C
C*************************************************************************
C
C     CALCULATE SIGNAL POWER
C
        PS = A ** 2 * 0.5
C
C*************************************************************************
C
C     CALCULATE INPUT NOISE DENSITY
C
        NO = PS / (10.0 ** (SNR / 10.0) * BREF)
C
C*************************************************************************
C
C     CALCULATE NOISE VARIANCE
C
        DOUBLE1 = SIGMAS1(NPOINT1) * NO ** 2
        DOUBLE2 = SIGMAS2(NPOINT2) * NO ** 2
        LAST = NPOINT1 - 1
          DO 300 LOOP = 0, LAST
            SIGMAS1(LOOP) = SIGMAS1(LOOP) * NO + DOUBLE1
            SIGMAS2(LOOP) = SIGMAS2(LOOP) * NO + DOUBLE2
 300      CONTINUE
C
C*************************************************************************
C
C     DETERMINE PEAK SIGNAL VALUE
C
        POSPEAK1 = SIGNAL1(0)
        NEGPEAK1 = SIGNAL1(0)
        POSPEAK2 = SIGNAL2(0)
        NEGPEAK2 = SIGNAL2(0)
           DO 400 LOOP = 0, LAST
            POSPEAK1 = AMAX1(POSPEAK1, SIGNAL1(LOOP))
            NEGPEAK1 = AMIN1(NEGPEAK1, SIGNAL1(LOOP))
            POSPEAK2 = AMAX1(POSPEAK2, SIGNAL2(LOOP))
            NEGPEAK2 = AMIN1(NEGPEAK2, SIGNAL2(LOOP))
  400      CONTINUE
C
C*************************************************************************
C
C     CALCULATE ESTIMATE FOR FIRST SIGNAL VALUE ABOVE THRESHOLD
C
        LOOK = .TRUE.
        IF (MINUS) LOOK = .FALSE.
        LOCATE1 = -1
        THRESHOLD1 = POSPEAK1 * PERCENT * 0.01
            DO 500 LOOP = 0, LAST
             IF (SIGNAL1(LOOP).EQ.NEGPEAK1) LOOK = .TRUE.
```

```
            IF (.NOT.LOOK) GO TO 500
            IF (SIGNAL1(LOOP).LT.THRESHOLD1) GO TO 500
            LOCATE1 = LOOP
            GO TO 600
  500     CONTINUE
  600   CONTINUE
        LOOK = .TRUE.
        IF (MINUS) LOOK = .FALSE.
        LOCATE2 = -1
        THRESHOLD2 = POSPEAK2 * PERCENT * 0.01
            DO 700 LOOP = 0, LAST
            IF (SIGNAL2(LOOP).EQ.NEGPEAK2) LOOK = .TRUE.
            IF (.NOT.LOOK) GO TO 700
            IF (SIGNAL2(LOOP).LT.THRESHOLD2) GO TO 700
            LOCATE2 = LOOP
            GO TO 800
  700     CONTINUE
  800   CONTINUE
C
        IF (LOCATE1.LT.0) PRINT *,'ERROR -> TOA THRESHOLD NOT LOCATED'
        IF (LOCATE2.LT.0) PRINT *,'ERROR -> TOD THRESHOLD NOT LOCATED'
        IF (LOCATE1.LT.0.OR.LOCATE2.LT.0) ERR = .TRUE.
        IF (ERR) GO TO 4
C
C*************************************************************************
C
C     LOCATE ACTUAL THRESHOLD CROSSINGS BY INTERPOLATION
C
        LOCAT1 = LOCATE1 - 1
        DIFSIG1 = SIGNAL1(LOCATE1) - SIGNAL1(LOCAT1)
        INTER1 = (SIGNAL1(LOCATE1) - THRESHOLD1) / DIFSIG1
C
        LOCAT2 = LOCATE2 - 1
        DIFSIG2 = SIGNAL2(LOCATE2) - SIGNAL2(LOCAT2)
        INTER2 = (SIGNAL2(LOCATE2) - THRESHOLD2) / DIFSIG2
C
C*************************************************************************
C
C     DETERMINE SLOPES AND SIGMAS OF ACTUAL THRESHOLD BY INTERPOLATION
C
        DSLOPE1 = SLOPE1(LOCATE1) - SLOPE1(LOCAT1)
        ASLOPE = SLOPE1(LOCATE1) - INTER1 * DSLOPE1
        DSIGMAS1 = SIGMAS1(LOCATE1) - SIGMAS1(LOCAT1)
        VAR1 = SIGMAS1(LOCATE1) - INTER1 * DSIGMAS1
C
        DSLOPE2 = SLOPE2(LOCATE2) - SLOPE2(LOCAT2)
        BSLOPE = SLOPE2(LOCATE2) - INTER2 * DSLOPE2
        DSIGMAS2 = SIGMAS2(LOCATE2) - SIGMAS2(LOCAT2)
        VAR2 = SIGMAS2(LOCATE2) - INTER2 * DSIGMAS2
C
C*************************************************************************
C
C     CALCULATE TOA AND TOD ERRORS
C
        ERROR1 = SQRT(VAR1) / ASLOPE
        ERROR2 = SQRT(VAR2) / BSLOPE
C
C*************************************************************************
C
C     DOCUMENT RUN
C
        OPEN(UNIT = 4, FILE = 'LST:', FORM = 'FORMATTED')
        WRITE(4,*) ' '
        IF(ICOUNT.GT.0) GO TO 12
        WRITE(4,*) 'PULSE WIDTH ESTIMATE -> TWO POST-FILTERS'
        WRITE(4,*) ' '
```

```
      WRITE(4,*) 'USER SUPPLIED INFORMATION'
      WRITE(4,*) ' '
      WRITE(4,11) '     TOA SIGNAL FILE      :  ', NAME1
      WRITE(4,11) '     TOA SLOPE FILE       :  ', NAME2
      WRITE(4,11) '     TOA SIGMAS FILE      :  ', NAME3
      WRITE(4,11) '     TOD SIGNAL FILE      :  ', NAME4
      WRITE(4,11) '     TOD SLOPE FILE       :  ', NAME5
      WRITE(4,11) '     TOD SIGMAS FILE      :  ', NAME6
      WRITE(4,*) ' '
      WRITE(4,22) '     THRESHOLD                  :  ', PERCENT, ' % '
 11   FORMAT(A27, A17)
 12   WRITE(4,*) ' '
      WRITE(4,22) '     SIGNAL TO NOISE RATIO      :  ', SNR, ' dB'
      WRITE(4,22) '     SNR REFERENCE BANDWIDTH    :  ', BREF, ' Hz'
 22   FORMAT(A34, G12.6, A3)
      WRITE(4,*) ' '
      WRITE(4,*)'                          RESULTS                          '
      WRITE(4,*)' '
      WRITE(4,*) '   AT SPECIFIED PERCENTAGE OF PEAK SIGNAL'
      ICOUNT = 1
      STEPS = REAL(NPOINT1 - 1)
      RTIME1 = (PERIOD * LOCAT1) / STEPS
      TOA = (-SIGNAL1(LOCAT1) + (ASLOPE * RTIME1)) / ASLOPE
      WRITE(4,33) TOA, ERROR1
 33   FORMAT(' TIME-OF-ARRIVAL ',G12.6,'s',' TOA ERROR ',G12.6,'s')
      RTIME2 = (PERIOD * LOCAT2) / STEPS
      TOD = (-SIGNAL2(LOCAT2) + (BSLOPE * RTIME2)) / BSLOPE
      WRITE(4,34) TOD, ERROR2
 34   FORMAT(' TIME-OF-DEPARTURE',G12.6,'s ','TOD ERROR',G12.6,'s')
      WIDTH = TOD - TOA
      VALUE3 = SQRT((ERROR1 ** 2) + (ERROR2 ** 2))
      WRITE(4,44)' PULSE WIDTH ',WIDTH,'s ',' RMS ERROR ',VALUE3,'s'
      WRITE(4,*)' '
      CLOSE(4)
 44   FORMAT(A13, G12.6, A2, A10, G12.6, A1)
      GO TO 2
  3   RETURN
      END
C
C
C
C
      SUBROUTINE RNAME1 (ISTRING, NAME)
      CHARACTER ISTRING*28, NAME*17
      PRINT5, ISTRING
  5   FORMAT (A28)
      READ10, NAME
 10   FORMAT (A17)
      RETURN
      END
C
C
C
C
      SUBROUTINE RDATA1 (LENGTH, VAR, NAME)
      CHARACTER NAME*17, A*1
      REAL VAR(513)
      LENGTH = 0
      OPEN (UNIT = 1, FILE = NAME)
      DO 101 LOOP = 1, 513
         READ(UNIT = 1, FMT = 21, END = 1) VAR(LOOP)
         LENGTH = LENGTH + 1
101   CONTINUE
 21   FORMAT(E16.8)
  1   CLOSE(UNIT = 1)
      RETURN
      ---
```

```
      SUBROUTINE PULSE1
C*************************************************************************
C
C       SUBROUTINE PULSE1
C
C       PULSEWIDTH ESTIMATE ->  FOR ONE POST-FILTER
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING     KANSAS STATE UNIVERSITY
C
C       PC REVISION            DATE              PROGRAMMER
C       -----------            ----              ----------
C          5.0                11-22-85          LONNIE HADEN
C
C*************************************************************************
C
      CHARACTER*17 NAME1, NAME2, NAME3
      REAL     NO, SIGMAS(0:512), SIGNAL(0:512), SLOPE(0:511),
     &         POSPEAK1, INTER1, INTER2
      LOGICAL  ERR, RESP, CK
C
C*************************************************************************
C
C
C
      CALL PRNT
      ERR = .FALSE.
      RESP = .FALSE.
    1 IF(ERR) CALL REPLY(' DO YOU WISH TO EXIT ? ', RESP)
      IF(RESP) GO TO 3
      ERR = .FALSE.
      PRINT *,' '
      PRINT *,' PULSE WIDTH ESTIMATE -> ONE POST-FILTER'
      PRINT *,' '
C
C
      CALL RNAME1('      TOA SIGNAL FILE ?        ', NAME1)
      CALL RNAME1('      TOA SLOPE  FILE ?        ', NAME2)
      CALL RNAME1('      TOA SIGMAS FILE ?        ', NAME3)
   82 CALL PRNT
      PRINT *,'  CHECK FOR VALID FILE NAMES !'
      PRINT *,' '
      PRINT *,'     (1)   TOA SIGNAL FILE IS : ', NAME1
      PRINT *,'     (2)   TOA SLOPE FILE IS  : ', NAME2
      PRINT *,'     (3)   TOA SIGMAS FILE IS : ', NAME3
      CALL REPLY('  DO YOU WISH TO CORRECT ?   ', CK)
      IF(.NOT.CK) GO TO 81
      PRINT *,' '
      PRINT *,' ENTER THE NUMBER OF THE INCORRECT FILE NAME.  '
      READ *, NCK
      PRINT *,' '
      GO TO (13,14,15) NCK
   13 PRINT *,' NEW TOA SIGNAL FILE ?  '
      READ19, NAME1
      GO TO 82
   14 PRINT *,' NEW TOA SLOPE FILE ?  '
      READ19, NAME2
      GO TO 82
   15 PRINT *,' NEW TOA SIGMAS FILE ?  '
      READ19, NAME3
      GO TO 82
   19 FORMAT(A17)
C
C
   81 CALL RDATA1(LENGTH1, SIGNAL(0), NAME1)
      CALL RDATA1(LENGTH2, SLOPE(0),  NAME2)
      CALL RDATA1(LENGTH3, SIGMAS(0), NAME3)
```

```
C
C*************************************************************************
C
C     INITIALIZATION
C
      CALL PRNT
      ICOUNT1 = 0
      A = 1.0
      NPOINT = LENGTH2
C
   2  IF(ICOUNT1.EQ.1) PRINT *,' ENTER -101 SNR TO EXIT.'
      PRINT *,' '
      PRINT *,' '
      PRINT *,'       SIGNAL TO NOISE RATIO (dB)    ? '
      READ *, SNR
      IF(SNR.EQ.-101) GO TO 3
      PRINT *,'       SNR REFERENCE BANDWIDTH (Hz)  ? '
      READ *, BREF
      PRINT *,'       THRESHOLD (% OF PEAK SIGNAL)  ? '
      READ *, PERCENT
C
      F0 = SIGNAL(NPOINT)
      PERIOD = 1.0 / F0
C
C*************************************************************************
C
C     CALCULATE SIGNAL POWER
C
      PS = A ** 2 / 2
C
C*************************************************************************
C
C     CALCULATE INPUT NOISE DENSITY
C
      NO = PS / (10.0 ** (SNR / 10.0) * BREF)
C
C
      DOUBLE1 = SIGMAS(NPOINT) * NO ** 2
      LAST = NPOINT - 1
         DO 300 LOOP = 0, LAST
          SIGMAS(LOOP) = SIGMAS(LOOP) * NO + DOUBLE1
  300     CONTINUE
C
C*************************************************************************
C
C     DETERMINE PEAK SIGNAL VALUE
C
      POSPEAK = SIGNAL(0)
         DO 400 LOOP = 0, LAST
          POSPEAK = AMAX1(POSPEAK, SIGNAL(LOOP))
  400     CONTINUE
C
C*************************************************************************
C
C     CALCULATE ESTIMATE FOR FIRST SIGNAL VALUE ABOVE THRESHOLD
C
      THRESHOLD = PERCENT * POSPEAK * 0.01
      DO 20 I = 0, NPOINT-1
        IF(SIGNAL(I).EQ.THRESHOLD.OR.SIGNAL(I).GT.THRESHOLD) THEN
          LOCATE1 = I
          GO TO 30
        ELSE
          LOCATE1 = -1
        ENDIF
  20  CONTINUE
C
```

```
C     FIND SECOND THRESHOLD
C
  30   DO 40 I = LOCATE1, NPOINT-1
         IF(SIGNAL(I).EQ.THRESHOLD.OR.SIGNAL(I).LT.THRESHOLD) THEN
           LOCATE2 = I
           GO TO 50
         ELSE
           LOCATE2 = -1
         ENDIF
  40   CONTINUE
C
C
C
  50   IF (LOCATE1.LT.0) PRINT *,'ERROR -> TOA THRESHOLD NOT LOCATED'
       IF (LOCATE2.LT.0) PRINT *,'ERROR -> TOD THRESHOLD NOT LOCATED'
       IF (LOCATE1.LT.0.OR.LOCATE2.LT.0) ERR = .TRUE.
       IF (ERR) GO TO 1
C
C*************************************************************************
C
C     LOCATE ACTUAL THRESHOLD CROSSINGS BY INTERPOLATION
C
       LOCAT1 = LOCATE1 - 1
       DIFSIG1 = SIGNAL(LOCATE1) - SIGNAL(LOCAT1)
       INTER1 = (SIGNAL(LOCATE1) - THRESHOLD) / DIFSIG1
C
       LOCAT2 = LOCATE2 - 1
       DIFSIG2 = SIGNAL(LOCAT2) - SIGNAL(LOCATE2)
       INTER2 = (SIGNAL(LOCAT2) - THRESHOLD) / DIFSIG2
C
C*************************************************************************
C
C     DETERMINE SLOPES AND SIGMAS OF ACTUAL THRESHOLD BY INTERPOLATION
C
       DSLOPE1 = SLOPE(LOCATE1) - SLOPE(LOCAT1)
       ASLOPE = SLOPE(LOCATE1) - INTER1 * DSLOPE1
       DSIGMAS1 = SIGMAS(LOCATE1) - SIGMAS(LOCAT1)
       VAR1 = SIGMAS(LOCATE1) - INTER1 * DSIGMAS1
C
       DSLOPE2 = SLOPE(LOCAT2) - SLOPE(LOCATE2)
       BSLOPE = SLOPE(LOCAT2) - INTER2 * DSLOPE2
       DSIGMAS2 = SIGMAS(LOCAT2) - SIGMAS(LOCATE2)
       VAR2 = SIGMAS(LOCAT2) - INTER2 * DSIGMAS2
C
C*************************************************************************
C
C     CALCULATE TOA AND TOD ERRORS
C
       ERROR1 = SQRT(VAR1) / ASLOPE
       ERROR2 = -SQRT(VAR2) / BSLOPE
C
C*************************************************************************
C
C     DOCUMENT RUN
C
       OPEN(UNIT = 4, FILE = 'LST:', FORM = 'FORMATTED')
       WRITE(4,*) ' '
       IF(ICOUNT1.EQ.1) GO TO 12
       WRITE(4,*) 'PULSE WIDTH ESTIMATE -> ONE POST-FILTER'
       WRITE(4,*) ' '
       WRITE(4,*) 'USER SUPPLIED INFORMATION'
       WRITE(4,*) ' '
       WRITE(4,11) '     TOA SIGNAL FILE     :  ', NAME1
       WRITE(4,11) '     TOA SLOPE FILE      :  ', NAME2
       WRITE(4,11) '     TOA SIGMAS FILE     :  ', NAME3
  11   FORMAT(A27, A17)
```

```
12    WRITE(4,*) ' '
      WRITE(4,22) '     SIGNAL TO NOISE RATIO      :  ', SNR, ' dB'
      WRITE(4,22) '     SNR REFERENCE BANDWIDTH    :  ', BREF, ' Hz'
      WRITE(4,22) '     THRESHOLD                  :  ', PERCENT, ' % '
22    FORMAT(A34, G12.6, A3)
      WRITE(4,*) ' '
      WRITE(4,*)'                        RESULTS                        '
      WRITE(4,*)' '
      WRITE(4,*) '   AT SPECIFIED PERCENTAGE OF PEAK SIGNAL'
      ICOUNT1 = 1
      STEPS = REAL(NPOINT - 1)
      RTIME1 = (PERIOD * LOCAT1) / STEPS
      TOA=(THRESHOLD-SIGNAL(LOCATE1)+(ASLOPE*RTIME1))/ASLOPE
      WRITE(4,33)'TIME-OF-ARRIVAL ',TOA,'s ','TOA ERROR ',ERROR1,'s'
 33   FORMAT(A17,G12.6,A2,A10,G12.6,A1)
      RTIME2 = (PERIOD * LOCAT2) / STEPS
      TOD=(SIGNAL(LOCAT2)-THRESHOLD+(BSLOPE*RTIME2))/BSLOPE
      WRITE(4,34)' TIME-OF-DEPARTURE ',TOD,'s ','TOD ERROR ',
     &    ERROR2,'s'
34    FORMAT(A19,G12.6,A2,A10,G12.6,A1)
      WIDTH = TOD - TOA
      VALUE3 = SQRT((ERROR1 ** 2) + (ERROR2 ** 2))
      WRITE(4,44)' PULSE WIDTH ',WIDTH,'s ','RMS ERROR ',VALUE3,'s'
      WRITE(4,*)' '
44    FORMAT(A13, G12.6, A2, A10, G12.6, A1)
      CLOSE(4)
      GO TO 2
 3    RETURN
      END
```

```
      SUBROUTINE ERROR4
C*************************************************************************
C
C       SUBROUTINE ERROR4
C
C       SIGNAL POWER BASED ON PULSE-WIDTH ERROR -> TWO POST-FILTERS
C
C       DEPARTMENT OF ELECTRIAL ENGINEERING      KANSAS STATE UNIVERSITY
C
C       REVISION                DATE             PROGRAMMER
C       --------                ----             ----------
C         0.0                  9-10-81           F. W. RATCLIFFE
C         1.2                  4-30-82           F. W. RATCLIFFE
C         1.3                  6-18-82           DIANE VONTHAER
C         1.4                  1-30-84           MARK HERMAN
C
C       PC REVISION             DATE             PROGRAMMER
C       -----------             ----             ----------
C         5.0                 11-22-85           LONNIE HADEN
C
C*************************************************************************
C
      CHARACTER*17 NAME1, NAME2, NAME3, NAME4, NAME5, NAME6
      INTEGER      PRIOR1, PRIOR2
      LOGICAL      LOOK, MINUS, SUBSEQUENT, ERR, RESP, CK
      REAL         NO, TERM1(0:512), TERM2(0:512), INTER1, INTER2
      EXTERNAL REPLY, RNAME1, RDATA1
C
C*************************************************************************
C
C     INITIALIZATION
C
      CALL PRNT
 1    ICHAN1 = 1
      ICHAN2 = 2
      ERR = .FALSE.
      RESP = .FALSE.
      IF(ERR) CALL REPLY(' DO YOU WISH TO EXIT ?  ', RESP)
      IF(RESP) GO TO 2
      ERR = .FALSE.
C
      SUBSEQUENT = .FALSE.
      PRINT *,' '
      PRINT *, 'SIGNAL POWER BASED ON PULSEWIDTH ERROR'
      PRINT *,' '
 200  CONTINUE
C
      CALL RNAME1('     TOA SIGNAL FILE ?        ', NAME1)
      CALL RNAME1('     TOA SLOPE  FILE ?        ', NAME2)
      CALL RNAME1('     TOA SIGMAS FILE ?        ', NAME3)
      PRINT *,' '
      CALL RNAME1('     TOD SIGNAL FILE ?        ', NAME4)
      CALL RNAME1('     TOD SLOPE  FILE ?        ', NAME5)
      CALL RNAME1('     TOD SIGMAS FILE ?        ', NAME6)
  82  CALL PRNT
      PRINT *,'  CHECK FOR VALID FILE NAMES !'
      PRINT *,' '
      PRINT *,'     (1)   TOA SIGNAL FILE IS : ', NAME1
      PRINT *,'     (2)   TOA SLOPE FILE IS  : ', NAME2
      PRINT *,'     (3)   TOA SIGMAS FILE IS : ', NAME3
      PRINT *,'     (4)   TOD SIGNAL FILE IS : ', NAME4
      PRINT *,'     (5)   TOD SLOPE FILE IS  : ', NAME5
      PRINT *,'     (6)   TOD SIGMAS FILE IS : ', NAME6
      CALL REPLY('  DO YOU WISH TO CORRECT ?   ', CK)
      IF(.NOT.CK) GO TO 81
      PRINT *,' '
```

```
      PRINT *,' ENTER THE NUMBER OF THE INCORRECT FILE NAME.  '
      READ *, NCK
      PRINT *,' '
      GO TO (13,14,15,16,17,18) NCK
   13 PRINT *,' NEW TOA SIGNAL FILE ?  '
      READ19, NAME1
      GO TO 82
   14 PRINT *,' NEW TOA SLOPE FILE ?  '
      READ19, NAME2
      GO TO 82
   15 PRINT *,' NEW TOA SIGMAS FILE ?  '
      READ19, NAME3
      GO TO 82
   16 PRINT *,' NEW TOD SIGNAL FILE ?  '
      READ19, NAME4
      GO TO 82
   17 PRINT *,' NEW TOD SLOPE FILE ?  '
      READ19, NAME5
      GO TO 82
   18 PRINT *,' NEW TOD SIGMAS FILE ?  '
      READ19, NAME6
      GO TO 82
   19 FORMAT(A17)
C
C
C
   81 CALL PRNT
      PERCENT = 0.0
      PRINT *, '      NOISE FIGURE (dB)  ?  '
      READ *, FDB
      PRINT *, '      RECEIVER GAIN (dB) ?  '
      READ *, GDB
      PRINT *, '      RF BANDWIDTH (Hz)  ?  '
      READ *, RF
      PRINT *,' '
      PRINT *,'       SEARCH FOR THRESHOLD AFTER OCCURRENCE OF'
      CALL REPLY('       NEGATIVE PEAK SIGNAL ? ', MINUS)
C
C*****************************************************************************
C
C     CALCULATE PARAMETERS DERIVED FROM CONSOLE INPUT
C
      F = 10.0 ** (FDB / 10.0)
      G = 10.0 ** (GDB / 10.0)
      NO = 4.0E-21 * F * G
C
C*****************************************************************************
C
C     DETERMINE MAXIMUM AND MINIMUM SIGNAL VLAUES
C
      CALL RDATA1 (LENGTH1, TERM1(0), NAME1)
      CALL RDATA1 (LENGTH4, TERM2(0), NAME4)
      IF(LENGTH1-1.EQ.LENGTH4) THEN
        NPOINT1 = LENGTH4
        NPOINT2 = LENGTH4
        LAST = NPOINT1 - 1
      ELSE
        CALL PRNT
        PRINT *,'     ****** ERROR ******'
        PRINT *,' FILES ARE NOT THE SAME LENGTH'
        WRITE(6,7)'FILE ',NAME1,' HAS LENGTH ',LENGTH1
        WRITE(6,7)'FILE ',NAME4,' HAS LENGTH ',LENGTH4
    7   FORMAT(A5,A17,A12,I3)
        ERR = .TRUE.
        GO TO 1
      ENDIF
```

```
      SIGMAX1 = TERM1(0)
      SIGMIN1 = TERM1(0)
      SIGMAX2 = TERM2(0)
      SIGMIN2 = TERM2(0)
            DO 300 LOOP = 0, LAST
              SIGMAX1 = AMAX1 (SIGMAX1, TERM1(LOOP))
              SIGMIN1 = AMIN1 (SIGMIN1, TERM1(LOOP))
              SIGMAX2 = AMAX1 (SIGMAX2, TERM2(LOOP))
              SIGMIN2 = AMIN1 (SIGMIN2, TERM2(LOOP))
  300       CONTINUE
C
C*****************************************************************************
C
C     LOCATE THRESHOLD
C
      LOCATE1 = -1
      LOOK = .TRUE.
      IF (MINUS) LOOK =.FALSE.
      THRESHOLD1 = SIGMAX1 * PERCENT * 0.01
            DO 400 LOOP = 0, LAST
              IF (TERM1(LOOP) .EQ. SIGMIN1) LOOK = .TRUE.
              IF (.NOT. LOOK) GO TO 400
              IF (TERM1(LOOP) .LT. THRESHOLD1) GO TO 400
              LOCATE1 = LOOP
              GO TO 500
  400       CONTINUE
  500 CONTINUE
C
C
      LOCATE2 = -1
      LOOK = .TRUE.
      IF (MINUS) LOOK = .FALSE.
      THRESHOLD2 = SIGMAX2 * PERCENT * 0.01
            DO 600 LOOP = 0, LAST
              IF (TERM2(LOOP) .EQ. SIGMIN2) LOOK = .TRUE.
              IF (.NOT. LOOK) GO TO 600
              IF (TERM2(LOOP) .LT. THRESHOLD2) GO TO 600
              LOCATE2 = LOOP
              GO TO 700
  600       CONTINUE
  700 CONTINUE
C
      IF (LOCATE1 .LT. 0) PRINT*,'ERROR -> TOA THRESHOLD NOT LOCATED'
      IF (LOCATE2 .LT. 0) PRINT*,'ERROR -> TOD THRESHOLD NOT LOCATED'
      IF (LOCATE1.LT.0.OR.LOCATE2.LT.0) ERR = .TRUE.
      IF (ERR) GO TO 1
C
C*****************************************************************************
C
C     LOCATE ACTUAL THRESHOLD CROSSINGS BY INTERPOLATION
C
      LOCAT1 = LOCATE1 - 1
      DIFSIG1 = TERM1(LOCATE1) - TERM1(LOCAT1)
      INTER1 = (TERM1(LOCATE1) - THRESHOLD1) / DIFSIG1
C
      LOCAT2 = LOCATE2 - 1
      DIFSIG2 = TERM2(LOCATE2) - TERM2(LOCAT2)
      INTER2 = (TERM2(LOCATE2) - THRESHOLD2) / DIFSIG2
C
C*****************************************************************************
C
C     DETERMINE SLOPES AND SIGMAS OF ACTUAL THRESHOLD CROSSINGS
C     BY INTERPOLATION
C
      CALL RDATA1(LENGTH2, TERM1(0), NAME2)
      DSLOPE1 = TERM1(LOCATE1) - TERM1(LOCAT1)
```

```
      SLOPE1 = TERM1(LOCATE1) - INTER1 * DSLOPE1
      CALL RDATA1(LENGTH3, TERM1(0), NAME3)
      DSIGMA1 = TERM1(LOCATE1) - TERM1(LOCAT1)
      SIGMA1 = TERM1(LOCATE1) - INTER1 * DSIGMA1
      DOUBLE1 = TERM1(NPOINT1)
C
C
C
      CALL RDATA1(LENGTH5, TERM2(0), NAME5)
      DSLOPE2 = TERM2(LOCATE2) - TERM2(LOCAT2)
      SLOPE2 = TERM2(LOCATE2) - INTER2 * DSLOPE2
      CALL RDATA1(LENGTH6, TERM2(0), NAME6)
      DSIGMA2 = TERM2(LOCATE2) - TERM2(LOCAT2)
      SIGMA2 = TERM2(LOCATE2) - INTER2 * DSIGMA2
      DOUBLE2 = TERM2(NPOINT2)
  800 CONTINUE
C
C
C
      IF(LENGTH2.EQ.LENGTH5.AND.LENGTH3.EQ.LENGTH6) THEN
        ERR = .FALSE.
      ELSE
        PRINT*,'      ***** ERROR *****'
        PRINT*,' FILE LENGTHS ARE NOT EQUAL '
        PRINT*,' '
        WRITE(6,7)'FILE ',NAME2,' HAS LENGTH ',LENGTH2
        WRITE(6,7)'FILE ',NAME3,' HAS LENGTH ',LENGTH3-1
        WRITE(6,7)'FILE ',NAME5,' HAS LENGTH ',LENGTH5
        WRITE(6,7)'FILE ',NAME6,' HAS LENGTH ',LENGTH6-1
        ERR = .TRUE.
        GO TO 1
      ENDIF
C
C*************************************************************************
C
C     INPUT INFORMATION FROM CONSOLE FOR THIS CASE
C
      PRINT *,' '
      PRINT *,'  INPUT PULSE WIDTH OF -1 TO EXIT '
      PRINT *,' '
      PRINT *, '      PULSE WIDTH ERROR (s) ?  '
      READ *, ERROR
      IF(ERROR .EQ. -1)GO TO 2
C
C*************************************************************************
C
C     CALCULATE SIGNAL POWER
C
      A = ERROR ** 2 * SLOPE1 ** 2 * SLOPE2 ** 2
      B = -NO * (SIGMA1 * SLOPE2 ** 2 + SIGMA2 * SLOPE1 ** 2)
      C = -NO ** 2 * (SLOPE2 ** 2 * DOUBLE1 + SLOPE1 ** 2 * DOUBLE2)
      ASQUARED = (-B + SQRT (B ** 2 - 4 * A * C)) / (2.0 * A)
      PS = ASQUARED / (2.0 * G)
      PSDBM = 10.0 * ALOG10 (PS * 1000.0)
C
C*************************************************************************
C
C    CLACULATE SIGNAL TO NOISE RATIO FOR THIS SIGNAL POWER
C
      SNR = PSDBM + 174.0 - FDB - 10.0 * ALOG10(RF)
C
C*************************************************************************
C
C    DOCUMENT RUN
C
      OPEN(UNIT = 4, FILE = 'LST:', FORM = 'FORMATTED')
```

```
      IF (SUBSEQUENT) GO TO 900
      SUBSEQUENT = .TRUE.
      WRITE(4,*) ' '
      WRITE(4,*) '   SIGNAL POWER BASED ON PULSEWIDTH ERROR  '
      WRITE(4,*) ' '
      WRITE(4,*) '   USER SUPPLIED INFORMATION COMMON TO ALL CASES'
      WRITE(4,*) ' '
      WRITE(4,11) '      TOA SIGNAL FILE  :  ', NAME1
      WRITE(4,11) '      TOA SLOPE  FILE  :  ', NAME2
      WRITE(4,11) '      TOA SIGMAS FILE  :  ', NAME3
      WRITE(4,11) '      TOD SIGNAL FILE  :  ', NAME4
      WRITE(4,11) '      TOD SLOPE  FILE  :  ', NAME5
      WRITE(4,*) ' '
      WRITE(4,11) '      TOD SIGMAS FILE  :  ', NAME6
11    FORMAT(A25, A17)
      WRITE(4,22) '      THRESHOLD        :  ', PERCENT, '  %'
      WRITE(4,22) '      NOISE FIGURE     :  ', FDB, ' dB'
      WRITE(4,22) '      RECEIVER GAIN    :  ', GDB, ' dB'
      WRITE(4,22) '      RF BANDWIDTH     :  ', RF, ' Hz'
22    FORMAT(A25, G12.6, A3)
      IF(MINUS) WRITE(4,*)'FIRST THRESHOLD AFTER NEGATIVE PEAK SIGNAL'
      IF (.NOT. MINUS) WRITE(4,*)'     FIRST THRESHOLD'
      WRITE(4,*)' '
      WRITE(4,*) '                      RESULTS                    '
900   CONTINUE
      WRITE(4,*)' '
      WRITE(4,22)'     PULSE WIDTH ERROR :  ', ERROR, '  s'
      WRITE(4,33) '      SIGNAL POWER = ', PSDBM, ' dBm'
      WRITE(4,33) '      SNR          = ', SNR, ' dB '
      WRITE(4,*)' '
      CLOSE(4)
33    FORMAT(A20, G15.6, A4)
      IF (SUBSEQUENT) GO TO 800
    2 RETURN
      END
```

```
      SUBROUTINE ERROR41
C*************************************************************************
C
C       SUBROUTINE ERROR41
C
C       SIGNAL POWER BASED ON PULSE-WIDTH ERROR -> ONE POST-FILTER
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING     KANSAS STATE UNIVERSITY
C
C       PC REVISION              DATE            PROGRMMER
C       -----------              ----            ---------
C           5.0                11-22-84          LONNIE HADEN
C
C*************************************************************************
C
      CHARACTER*17 NAME1, NAME2, NAME3, NAME4, NAME5, NAME6
      INTEGER      PRIOR1, PRIOR2
      LOGICAL      LOOK, MINUS, SUBSEQUENT, ERR, RESP, CK
      REAL         NO, TERM1(0:512), TERM2(0:512), INTER1, INTER2
      EXTERNAL REPLY, RNAME1, RDATA1
C
C*************************************************************************
C
C     INITIALIZATION
C
      CALL PRNT
 1    ICHAN1 = 1
      ICHAN2 = 2
      ERR = .FALSE.
      RESP = .FALSE.
      IF(ERR) CALL REPLY(' DO YOU WISH TO EXIT ?  ', RESP)
      IF(RESP) GO TO 2
      ERR = .FALSE.
C
      SUBSEQUENT = .FALSE.
      PRINT *,' '
      PRINT *, 'SIGNAL POWER BASED ON PULSEWIDTH ERROR'
      PRINT *,'          (ONE POST-FILTER)  '
      PRINT *,' '
      PRINT *,' '
 200  CONTINUE
C
      CALL RNAME1('     TOA SIGNAL FILE ?        ', NAME1)
      CALL RNAME1('     TOA SLOPE  FILE ?        ', NAME2)
      CALL RNAME1('     TOA SIGMAS FILE ?        ', NAME3)
  82  CALL PRNT
      PRINT *,'  CHECK FOR VALID FILE NAMES !'
      PRINT *,' '
      PRINT *,'     (1)   TOA SIGNAL FILE IS : ', NAME1
      PRINT *,'     (2)   TOA SLOPE FILE IS  : ', NAME2
      PRINT *,'     (3)   TOA SIGMAS FILE IS : ', NAME3
      CALL REPLY('  DO YOU WISH TO CORRECT ?   ', CK)
      IF(.NOT.CK) GO TO 81
      PRINT *,' '
      PRINT *,' ENTER THE NUMBER OF THE INCORRECT FILE NAME.  '
      READ *, NCK
      PRINT *,' '
      GO TO (13,14,15) NCK
  13  PRINT *,' NEW TOA SIGNAL FILE ?  '
      READ19, NAME1
      GO TO 82
  14  PRINT *,' NEW TOA SLOPE FILE ?  '
      READ19, NAME2
      GO TO 82
  15  PRINT *,' NEW TOA SIGMAS FILE ?  '
      READ19, NAME3
```

```
      GO TO 82
   19 FORMAT(A17)
C
C
C
   81 CALL PRNT
      PRINT *, '     THRESHOLD (% OF PEAK SIGNAL) ?  '
      READ *, PERCENT
      PRINT *, '     NOISE FIGURE (dB)  ?  '
      READ *, FDB
      PRINT *, '     RECEIVER GAIN (dB) ?  '
      READ *, GDB
      PRINT *, '     RF BANDWIDTH (Hz)  ?  '
      READ *, RF
      PRINT *,' '
C
C*************************************************************************
C
C     CALCULATE PARAMETERS DERIVED FROM CONSOLE INPUT
C
      F = 10.0 ** (FDB / 10.0)
      G = 10.0 ** (GDB / 10.0)
      NO = 4.0E-21 * F * G
C
C*************************************************************************
C
C     DETERMINE MAXIMUM AND MINIMUM SIGNAL VLAUES
C
      CALL RDATA1 (LENGTH1, TERM1(0), NAME1)
      NPOINT = LENGTH1 - 1
      SIGMAX = TERM1(0)
             DO 300 LOOP = 0, NPOINT-1
               SIGMAX = AMAX1 (SIGMAX, TERM1(LOOP))
  300        CONTINUE
C
C*************************************************************************
C
C     LOCATE THRESHOLD
C
C
      THRESHOLD = PERCENT * SIGMAX * 0.01
      DO 20 I = 0, NPOINT-1
        IF(TERM1(I).EQ.THRESHOLD.OR.TERM1(I).GT.THRESHOLD) THEN
          LOCATE1 = I
          GO TO 30
        ELSE
          LOCATE1 = -1
        ENDIF
   20   CONTINUE
C
C
C
   30   DO 40 I = LOCATE1, NPOINT-1
          IF(TERM1(I).EQ.THRESHOLD.OR.TERM1(I).LT.THRESHOLD) THEN
            LOCATE2 = I
            GO TO 50
          ELSE
            LOCATE2 = -1
          ENDIF
   40   CONTINUE
C
C
C
   50   IF (LOCATE1 .LT. 0) PRINT*,'ERROR -> TOA THRESHOLD NOT LOCATED'
        IF (LOCATE2 .LT. 0) PRINT*,'ERROR -> TOD THRESHOLD NOT LOCATED'
        IF (LOCATE1.LT.0.OR.LOCATE2.LT.0) ERR = .TRUE.
```

```
      IF (ERR) GO TO 1
C
C*************************************************************************
C
C     LOCATE ACTUAL THRESHOLD CROSSINGS BY INTERPOLATION
C
      LOCAT1 = LOCATE1 - 1
      DIFSIG1 = TERM1(LOCATE1) - TERM1(LOCAT1)
      INTER1 = (TERM1(LOCATE1) - THRESHOLD) / DIFSIG1
C
      LOCAT2 = LOCATE2 - 1
      DIFSIG2 = TERM1(LOCAT2) - TERM1(LOCATE2)
      INTER2 = (TERM1(LOCAT2) - THRESHOLD) / DIFSIG2
C
C*************************************************************************
C
C     DETERMINE SLOPES AND SIGMAS OF ACTUAL THRESHOLD CROSSINGS
C     BY INTERPOLATION
C
      CALL RDATA1(LENGTH2, TERM1(0), NAME2)
      CALL RDATA1(LENGTH3, TERM2(0), NAME3)
C
C
C
      DSLOPE1 = TERM1(LOCATE1) - TERM1(LOCAT1)
      SLOPE1 = TERM1(LOCATE1) - INTER1 * DSLOPE1
      DSIGMA1 = TERM2(LOCATE1) - TERM2(LOCAT1)
      SIGMA1 = TERM2(LOCATE1) - INTER1 * DSIGMA1
      DOUBLE = TERM2(NPOINT)
C
C
C



      DSLOPE2 = TERM1(LOCAT2) - TERM1(LOCATE2)
      SLOPE2 = TERM1(LOCAT2) - INTER2 * DSLOPE2
      DSIGMA2 = TERM2(LOCAT2) - TERM2(LOCATE2)
      SIGMA2 = TERM2(LOCAT2) - INTER2 * DSIGMA2
  800 CONTINUE
C
C
C*************************************************************************
C
C     INPUT INFORMATION FROM CONSOLE FOR THIS CASE
C
      PRINT *,' '
      PRINT *,'  INPUT PULSE WIDTH OF -1 TO EXIT '
      PRINT *,' '
      PRINT *, '       PULSE WIDTH ERROR (s) ?  '
      READ *, ERROR
      IF(ERROR .EQ. -1)GO TO 2
C
C*************************************************************************
C
C     CALCULATE SIGNAL POWER
C
      A = ERROR ** 2 * SLOPE1 ** 2 * SLOPE2 ** 2

      B = -NO * (SIGMA1 * SLOPE2 ** 2 + SIGMA2 * SLOPE1 ** 2)
C
C*******   CHECK  **************
C
      C = -NO ** 2 * (SLOPE2 ** 2 * DOUBLE + SLOPE1 ** 2 * DOUBLE)
C
C******
C
      --------  . -  ---- -- .. -  . . - . -.. . -- - . ..
```

```
      ASQUARED = (-B + SQRT (B ** 2 - 4 * A * C)) / (2.0 * A)
      PS = ASQUARED / (2.0 * G)
      PSDBM = 10.0 * ALOG10 (PS * 1000.0)
C
C*******************************************************************************
C
C     CLACULATE SIGNAL TO NOISE RATIO FOR THIS SIGNAL POWER
C
      SNR = PSDBM + 174.0 - FDB - 10.0 * ALOG10(RF)
C
C*******************************************************************************
C
C     DOCUMENT RUN
C
      OPEN(UNIT = 4, FILE = 'LST:', FORM = 'FORMATTED')
      IF (SUBSEQUENT) GO TO 900
      SUBSEQUENT = .TRUE.
      WRITE(4,*) ' '
      WRITE(4,*) '    SIGNAL POWER BASED ON PULSEWIDTH ERROR  '
      WRITE(4,*) ' '
      WRITE(4,*) '    USER SUPPLIED INFORMATION COMMON TO ALL CASES'
      WRITE(4,*) ' '
      WRITE(4,11) '      TOA SIGNAL FILE  :  ', NAME1
      WRITE(4,11) '      TOA SLOPE  FILE  :  ', NAME2
      WRITE(4,11) '      TOA SIGMAS FILE  :  ', NAME3
11    FORMAT(A25, A17)
      WRITE(4,*) ' '
      WRITE(4,22) '      THRESHOLD        :  ', PERCENT, '  %'
      WRITE(4,22) '      NOISE FIGURE     :  ', FDB, ' dB'
      WRITE(4,22) '      RECEIVER GAIN    :  ', GDB, ' dB'
      WRITE(4,22) '      RF BANDWIDTH     :  ', RF, ' Hz'
      WRITE(4,*) ' '
      WRITE(4,*) ' '
22    FORMAT(A25, G12.6, A3)
      WRITE(4,*) '     FIRST THRESHOLD'
      WRITE(4,*)' '
      WRITE(4,*) '                          RESULTS                '
900   CONTINUE
      WRITE(4,*)' '
      WRITE(4,22)'      PULSE WIDTH ERROR :  ', ERROR, '  s'
      WRITE(4,33) '      SIGNAL POWER = ', PSDBM, ' dBm'
      WRITE(4,33) '      SNR          = ', SNR, ' dB '
      WRITE(4,*)' '
33    FORMAT(A20, G15.6, A4)
      CLOSE(4)
      IF (SUBSEQUENT) GO TO 800
    2 RETURN
      END
```

REFERENCES

[1] M. I. Skolnik, Introduction to Radar Systems, McGraw-Hill, Inc., New York, 1980.

[2] D. R. Hummels and F. W. Ratcliffe, Filter Selection for Crystal Video Receivers, Engineering Experiment Station, Project 2762, Manhattan, Kansas, 1981.

[3] M. Schwartz, W. R. Bennett, and S. Stein, Communication Systems and Techniques, McGraw-Hill, Inc., New York, 1966.

[4] J. B. Thomas, Statistical Communication Theory, John Wiley & Sons, Inc., New York, 1969.

[5] J. M. Wozencraft and I. M. Jacops, Principles of Communication Engineering, John Wiley & Sons, Inc., New York, 1965.

[6] D. R. Hummels, M. Brack, and M. Herman, Intercept Receiver Study Interim Report, Engineering Experiment Station, Project 2835, Manhattan, Kansas, 1984.

A NUMERICAL PROCEDURE FOR COMPUTING ERRORS IN THE MEASUREMENT OF PULSE TIME-OF-ARRIVAL AND PULSE-WIDTH

by

LONNIE A. HADEN

B.S., Kansas State University, 1984

AN ABSTRACT OF A MASTER'S REPORT

Submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1985

ABSTRACT

The principal goal of this report was the development of a computer program which can be used to predict pulse-width and RMS pulse-width measurement error for a square-law pulse receiver. The report is designed to be used as a manual for using the program. The computer program can be used for two different types of square-law receivers. The first is a single post-filter receiver, and the second is a two post-filter receiver. There are two main advantages of the numerical procedures used by the program over previous procedures. The first is that it takes pulse shape into account, and, secondly, it takes the transfer functions of the receiver filters into account.