# A NUMERICAL PROCEDURE FOR COMPUTING PROBABILITY OF DETECTION FOR A WIDEBAND PULSE RECEIVER

by

SCOTT D. BRILES

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

1986

Approved by:

Donald R. Hummels

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

## I. INTRODUCTION

This report describes a procedure for computing the probability of detection for a wideband pulse receiver which uses square-law detection. A Gaussian assumption based on a large ratio of pre-filter to post-filter bandwidth is the only significant restriction in the procedures developed. The actual transfer functions of filters used in the receiver and the parameters of the received RF pulse are used in the calculation of detection probability.

The numerical procedures described herein have been used to develop a computer program, PDETGAUS, which can be used to make performance predictions for wideband receivers. Sections II-V of this report provide the receiver modeling and analysis which justify the numerical procedures implemented in the program PDETGAUS. Section VI is intended to serve as a user's guide for the program.

# II. RECEIVER MODELING

Determining the probability of detection requires finding numerical expressions for the output signal and the output noise variance. The numerical expressions are derived from an equivalent low-pass model of the receiver. The equivalent low-pass model is developed as follows.

A block diagram of the square-law receiver and the corresponding equivalent low-pass model is shown in Figure 1. The receiver consists of a bandpass pre-filter having transfer function H(f) and a square-law envelope detector followed by a low-pass post-filter having transfer function G(f). Both filters are assumed to be unity gain linear filters. The input to the receiver is comprised of the signal of interest, s(t), plus the noise process, n(t). The signal of interest, s(t), is a radio frequency pulse with carrier frequency, $\omega_c$. The noise process, n(t), is a stationary, zero mean, white Gaussian random process. The power spectrum of n(t) is given by

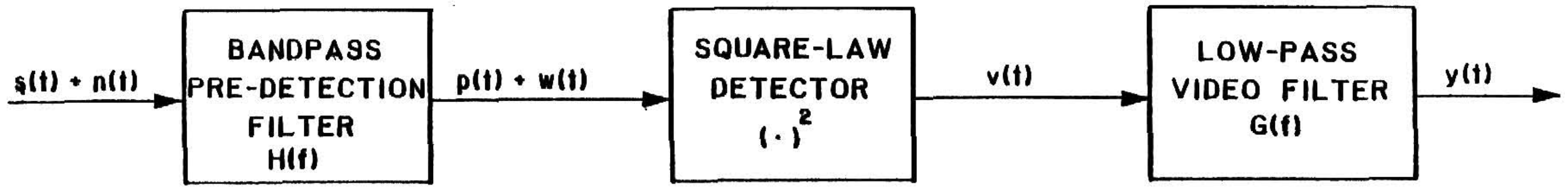
$$S_n(f) = \frac{No}{2}\ , \tag{2.1}$$

over at least a bandwidth wider than the pre-filter bandwidth.

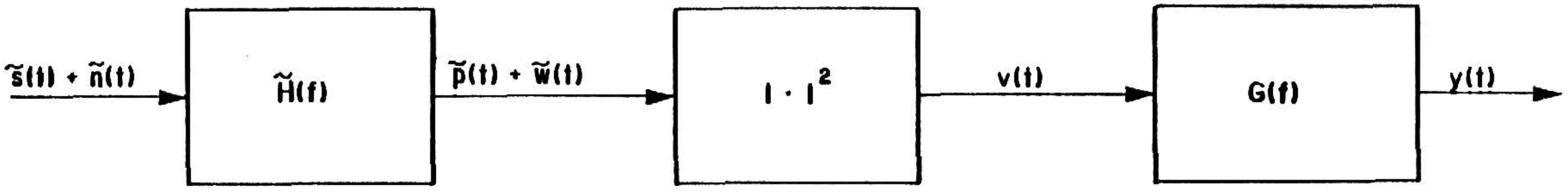
The low-pass model is developed in the usual way [1] where the signal s(t) is represented by its complex envelope $\tilde{s}(t)$. The relationship between s(t) and $\tilde{s}(t)$ is

$$s(t) = \mathrm{Re}\{\tilde{s}(t)e^{j\omega_c t}\} \tag{2.2}$$

where $\omega_c$ is the carrier frequency of the signal and is generally taken to be the center frequency of the bandpass pre-filter. The case where $\omega_c$, the frequency of the received pulse is not the center frequency of the pre-filter is easily handled by a shift in the frequency of the complex envelope.


s(t) + n(t)
BANDPASS PRE-DETECTION FILTER H(f)
p(t) + w(t)
SQUARE-LAW DETECTOR $(\cdot)^2$
v(t)
LOW-PASS VIDEO FILTER G(f)
y(t)

a) Physical System



b) Equivalent Low-Pass Model

Figure 1. Block Diagram of Receiver Model

The noise process, n(t) can also be represented by a complex envelope, $\tilde{n}(t)$. The relationship is essentially the same, i.e.,

$$n(t) = \text{Re}\{\tilde{n}(t)\, e^{j\omega_c t}\} \qquad (2.3)$$

where $\tilde{n}(t)$ is a stationary complex Gaussian process. The power spectrum of n(t) is related to the power spectrum of $\tilde{n}(t)$ by the relationship,

$$S_n(f) = \frac{1}{4} S_{\tilde{n}}(f-f_c) + \frac{1}{4} S_{\tilde{n}}(-f-f_c). \qquad (2.4)$$

Thus, the power spectrum of $\tilde{n}(t)$ using Equation (2.1) is

$$S_{\tilde{n}}(f) = 2\, No \qquad (2.5)$$

over a bandwidth wider than the bandwidth of the low-pass equivalent of the pre-filter.

The transfer function of the bandpass pre-filter, H(f), and its low-pass equivalent, $\tilde{H}(f)$, are related by

$$H(f) = \tilde{H}(f-f_c) + \tilde{H}(-f-f_c). \qquad (2.6)$$

The low-pass equivalent transfer function, $\tilde{H}(f)$, has a corresponding low-pass equivalent impulse response, $\tilde{h}(t)$.

At the output of the pre-filter, the complex envelope is described by the sum $\tilde{p}(t) + \tilde{w}(t)$, where

$$\tilde{p}(t) = \tilde{s}(t) \star \tilde{h}(t) \qquad (2.7a)$$

and

$$\tilde{w}(t) = \tilde{n}(t) \star \tilde{h}(t). \qquad (2.7b)$$

The operation $\star$ denotes convolution.

The output of the square-law detector is a voltage, v(t), which is proportional to the magnitude squared of the complex envelope of the signal plus noise at the detector input. Examination of the model shown in Figure 1 reveals the output of the square-law detector to be

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

After the multiplication, this becomes

$$v(t) = |\tilde{p}(t)|^2 + \tilde{p}(t)\tilde{w}^*(t) + \tilde{p}^*(t)\tilde{w}(t) + |\tilde{w}(t)|^2. \quad (2.8)$$

The voltage v(t) can be interpreted as the sum of two components, the signal component of interest and the noise component. The signal of interest, $v_s(t)$, is the term of v(t) which contains no noise, i.e.

$$v_s(t) = |\tilde{p}(t)|^2 \quad (2.9)$$

The noise component $v_n(t)$ is the sum of the remaining terms, i.e.,

$$v_n(t) = \tilde{p}(t)\tilde{w}^*(t) + \tilde{p}^*(t)\tilde{w}(t) + |\tilde{w}(t)|^2. \quad (2.10)$$

This noise component is a nonstationary noise process because of the product terms involving signal and noise.

The output of the post-filter, y(t), is the voltage v(t) convolved with the impulse response of the post-filter, g(t). This output can be described as

$$y(t) = y_s(t) + y_n(t), \quad (2.11)$$

where

$$y_s(t) = v_s(t) \star g(t) \quad (2.12a)$$

and

$$y_n(t) = v_n(t) \star g(t). \quad (2.12b)$$

The term, $y_s(t)$, is the output of the receiver if the input is noise free. This desired output, $y_s(t)$, is examined in the next section. The statistics of the undesired output term, $y_n(t)$, a non-stationary noise process, are developed in Section IV.

## III. OUTPUT SIGNAL ANALYSIS

The goal of the analysis which follows is to develop an expression which may be used to compute the output signal, $y_s(t)$.

Working with the equivalent low-pass model of the receiver, the input signal is represented by its complex envelope, $\tilde{s}(t)$. From the construction of the low-pass model, the desired output, $y_s(t)$, can be factored as follows:

$$\begin{aligned} y_s(t) &= v_s(t) \star g(t) \\ y_s(t) &= \tilde{p}(t)\tilde{p}^*(t) \star g(t) \qquad (3.1) \\ y_s(t) &= \left\{[\tilde{s}(t) \star \tilde{h}(t)][\tilde{s}^*(t) \star \tilde{h}^*(t)]\right\} \star g(t). \end{aligned}$$

This gives $y_s(t)$ as a function of the impulse responses of the filters and the complex envelope of the input signal.

Using the transform relationships between convolution and multiplication, the desired output signal can be expressed through calculations in the frequency domain. Assuming the complex envelope $\tilde{s}(t)$ has a Fourier transform, the transform can be expressed as

$$\tilde{S}(f) = F[\tilde{s}(t)], \qquad (3.2)$$

where F[ ] is used to indicate a Fourier Transform operator. The output of the pre-filter in the frequency domain is then

$$\tilde{P}(f) = \tilde{S}(f)\tilde{H}(f). \qquad (3.3)$$

The complex envelope of the pre-filter output is then given by the inverse transform, viz.,

$$\tilde{p}(t) = F^{-1}[\tilde{S}(f)\tilde{H}(f)]. \qquad (3.4)$$

The Fourier transform of the signal output term of the square-law detector may be now found as

$$V_s(f) = F[\tilde{p}(t)\tilde{p}^*(t)]. \qquad (3.5)$$

Post-filtering $V_s(f)$ yields a frequency domain representation of the receiver output signal. $Y_s(f)$ is thus given by

$$Y_s(f) = V_s(f)\ G(f). \tag{3.6}$$

Taking the inverse transform gives the desired output signal,

$$y_s(t) = F^{-1}[Y_s(f)], \tag{3.7}$$

as a function of time.

All the Fourier transforms indicated in Equations 3.2-3.7 can be performed numerically using the Fast Fourier Transform (FFT) algorithm. These equations suggest a procedure for computing the output signal waveform. Of course, correct sampling of the complex envelope $\tilde{s}(t)$ and proper representation of the bandwidths of the filters must be taken into account in order to obtain valid results.

## IV. CALCULATING THE VARIANCE OF THE OUTPUT NOISE

The output noise mean is determined in Appendix A to be

$$\overline{y_n} = R_{\tilde{w}}(0)G(0). \tag{4.1}$$

Since only unity gain filters are being considered, the post-filter dc response is one. Thus the mean depends on the autocorrelation of the noise process after the pre-filter.

The transform relationship for the autocorrelation is

$$R_{\tilde{w}}(\tau) = F^{-1}\,[S_{\tilde{w}}(f)] = \int_{-\infty}^{\infty} S_{\tilde{w}}(f)\, e^{j\omega\tau} df \tag{4.2}$$

where $S_{\tilde{w}}(f)$ is the power spectral density of the noise process after the pre-filter. Using Equation 2.5 and substituting into Equation 4.1 yields

$$\overline{y_n} = 2\ N_o \int_{-\infty}^{\infty} |H(f)|^2\ dt. \tag{4.3}$$

The integral is equal to twice the equivalent noise bandwidth of the pre-filter, $B_{n_{pre}}$. Thus the output noise mean can be expressed as

$$\overline{y_n} = 4\ N_o\ B_{n_{pre}}\ . \tag{4.4}$$

The autocorrelation function of the output noise of the receiver is derived in Appendix A as

$$R_{y_n}(t,s) = R_{y_1}(t,s) + R_{y_1}^{*}(t,s) + R_{y_3}(t,s) + R_{\tilde{w}}^{2}(0)G^2(0), \tag{4.5}$$

where

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

$$R_{y_3}(t,s) = \iint_{-\infty}^{\infty} |R_{\tilde{w}}(\alpha-\beta)|^2\ g(t-\alpha)g(s-\beta)\ d\alpha d\beta.$$

The variance of the output noise can be expressed as

$$\sigma^2_{y_n}(t) = R_{y_n}(t,s)\Big|_{s=t} - \overline{y_n(t)}^{\,2} \quad . \qquad (4.6)$$

Substituting the results of Appendix A yields

$$\sigma^2_{y_n}(t) = R_{y_1}(t,t) + R_{y_1}^{*}(t,t) + R_{y_3}(t,t). \qquad (4.7)$$

First consider the term $Ry_1(t,t)$, where

$$R_{y_1}(t,t) = \iint_{-\infty}^{\infty} \tilde{p}(\alpha)\tilde{p}^*(\beta)\ R_{\tilde{w}}^*(\alpha-\beta)g(t-\alpha)g(t-\beta)\ d\alpha d\beta.$$

Letting

$$\nu = t-\alpha \quad \text{and} \quad \mu = t-\beta,$$

the substitution of variables gives

$$R_{y_1}(t,t) = \iint_{-\infty}^{\infty} \tilde{p}(t-\nu)\tilde{p}^*(t-\mu)R_{\tilde{w}}^*(\mu-\nu)\ g(\nu)g(\mu)\ d\nu d\mu. \qquad (4.8)$$

Using the following transform relationship of Equation 4.2, yields

$$R_{y_1}(t,t) = \iint_{-\infty}^{\infty} \tilde{p}(t-\nu)\tilde{p}^*(t-\mu)\left[\int_{-\infty}^{\infty} S_{\tilde{w}}(f)\ e^{j\omega(\mu-\nu)}\ df\right]^* g(\nu)g(\mu)\ d\nu d\mu \qquad (4.9)$$

Rearranging yields

$$R_{y_1}(t,t) = \int_{-\infty}^{\infty} S_{\tilde{w}}(f) \int_{-\infty}^{\infty} p(t-\nu)g(\nu)e^{j\omega\nu}d\nu \int_{-\infty}^{\infty} \tilde{p}^*(t-\mu)g(\mu)e^{-j\omega\mu}d\mu df \qquad (4.10)$$

The impulse response of the post-filter, g(t), is real; thus,

$$\int_{-\infty}^{\infty} \tilde{p}(t-\nu)g(\nu)e^{j\omega\nu}d\nu = \left[\int_{-\infty}^{\infty} \tilde{p}^*(t-\mu)g(\mu)e^{-j\omega\mu}d\mu\right]^* . \qquad (4.11a)$$

Furthermore,

$$\int_{-\infty}^{\infty} \tilde{p}^*(t-\mu)g(\mu)e^{-jw\mu}d\mu = F\left[\tilde{p}^*(t-\mu)g(\mu)\right] . \qquad (4.11b)$$

These two relationships can be sutstituted into the $R_{y_1}(t,t)$ to yield

$$R_{y_1}(t,t) = \int_{-\infty}^{\infty} S_{\tilde{w}}(f)\left|F\left[\tilde{p}^*(t-\mu)g(\mu)\right]\right|^2 df. \qquad (4.12)$$

$R_{y_1}(t,t)$ is observed to be a time varying function since the shift of the post-filtered signal, $\tilde{p}(t)$, is a function of time. A numerical solution for $R_{y_1}(t,t)$ as a function of time is needed to find the variance of the output noise.

## Numerical Integration of $Ry_1(t,t)$

The integration of $R_{y_1}(t,t)$ can be achieved approximatley by a discrete integration rule, known as the composite Simpson rule, which is

$$S_N = \frac{h}{6}\left[f_0 + f_N + 2\sum_{i=1}^{N-1} f_i + 4\sum_{i=1}^{N} f_{i-1/2}\right], \tag{4.13}$$

where

h is the spacing betweem sample points, $h = x_i - x_{i-1}$, and

$f_x$ is the value of the function at the discrete sample number x.

In order to apply the composite Simpson rule to Equation 4.8, the following procedure is used in PDETGAUS. The assumed periodic series $p(\mu)$ is changed to $p(-\mu)$, shifted by t seconds and multiplied by the time series $g(\mu)$. This new series is Fourier transformed and multiplied by the power spectrum $S_{\tilde{w}}(f)$ evaluated at the appropriate frequencies. The composite Simpson rule is then used to integrate the product of the Fourier transform and $S_{\tilde{w}}(f)$. Note that for each value of t, the procedure must be repeated.

## A Frequency Domain Expression for $R_{y_3}(t,t)$

Next, consider $R_{y_3}(t,t)$, which is given by

$$Ry_3(t,t) = \iint_{-\infty}^{\infty} \left|R_{\tilde{w}}(\alpha-\beta)\right|^2 g(t-\alpha)g(t-\beta)\, d\alpha d\beta \ .$$

Using the following substitution of variables,

$$\nu = t-\alpha \ ; \quad \mu = t-\beta$$

leads to

$$Ry_3(t,t) = \iint_{-\infty}^{\infty} \left| R_{\tilde{w}}(\mu-\nu) \right|^2 g(\nu)g(\mu)\; d\nu d\mu. \qquad (4.14)$$

However, since autocorrelation and Power Spectral Density are transform pairs, substitution gives

$$R_{y_3}(t,t) = \iiint_{-\infty}^{\infty} S_{\tilde{w}}(f)e^{j2\pi f_1(\mu-\nu)} df_1 \left[ \int_{-\infty}^{\infty} S_{\tilde{w}}(f_2)e^{j2\pi f_2(\mu-\nu)} df_2 \right]^* g(\nu)g(\mu)d\nu d\mu.$$

Power spectral density functions are always real, thus

$$R_{y_3}(t,t) = \iiiint_{-\infty}^{\infty} S_{\tilde{w}}(f_1)e^{j\pi f_1(\mu-\nu)} S_{\tilde{w}}(f_2)e^{-j2\pi f_2(\mu-\nu)} g(\nu)g(\mu)d\mu d\nu df_1 df_2.$$

Rearranging gives

$$R_{y_3}(t,t) = \iint_{-\infty}^{\infty} S_w(f_1)S_w(f_2) \int_{-\infty}^{\infty} g(\nu)e^{-j2\pi(f_1-f_2)\nu} d\nu \int_{-\infty}^{\infty} g(\nu)e^{j2\pi(f_1-f_2)\nu} d\mu df_1 df_2. \qquad (4.15)$$

Since g(t) is real, being the impulse response of the post-filter, it follows that

$$\int_{-\infty}^{\infty} g(\mu)e^{j2\pi(f_1-f_2)\mu}\, d\mu = \left[ \int_{-\infty}^{\infty} g(\mu)e^{-j2\pi(f_1-f_2)\mu}\, d\mu \right]^*. \qquad (4.16)$$

Substituting the Fourier transforms yields

$$R_{y_3}(t,t) = \iint_{-\infty}^{\infty} S_{\tilde{w}}(f_1)S_{\tilde{w}}(f_2) \left| G(f_1-f_2) \right|^2 df_2 df_1 \;. \qquad (4.17)$$

This expression for $R_{y_3}(t,t)$ does not depend on time, and thus is a constant and can be expressed as $R_{y_3}$. $R_{y_3}$ is the variance of the output noise process with only noise as the input to the receiver.

## A Simplification for the Case of Large Bandwidth Ratio

Performing the following substitution of variables,

$$\alpha = f_1 - f_2$$

yields

$$R_{y_3} = \iint_{-\infty}^{\infty} S_{\tilde{w}}(f_1) S_{\tilde{w}}(f_1-\alpha) \left|G(\alpha)\right|^2 df_1 d\alpha \tag{4.18}$$

Since $\tilde{w}(t)$ represents the noise at the pre-filter output, the power spectrum is readily shown to be

$$S_{\tilde{w}}(f) = 2No|\tilde{H}(f)|^2 \quad , \tag{4.19}$$

where 2No is the magnitude of the noise power spectral density, and $\tilde{H}(f)$ is the transfer function of the low pass equivalent pre-filter. Power spectral densities are always real and even, thus

$$S_{\tilde{w}}(f) = S_{\tilde{w}}(-f),$$

which may be substituted to obtain

$$R_{y_3} = \iint_{-\infty}^{\infty} S_{\tilde{w}}(f_1) S_{\tilde{w}}(\alpha-f_1) \left|G(\alpha)\right|^2 df_1 d\alpha \tag{4.20}$$

Rearranging the integrations gives,

$$R_{y_3} = \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} S_{\tilde{w}}(f_1) S_{\tilde{w}}(\alpha-f_1) df_1 \right] \left|G(\alpha)\right|^2 d\alpha \tag{4.21}$$

The inside integration is the convolution of the power spectral density with itself, thus

$$R_{y_3} = \int_{-\infty}^{\infty} \left[ S_{\tilde{w}}(f) \star S_{\tilde{w}}(f) \right] \left|G(\alpha)\right|^2 d\alpha \ . \tag{4.22}$$

Substituting in (4.19) yields

$$R_{y_3} = 4\ No^2 \int_{-\infty}^{\infty} \left[ |\tilde{H}(f)|^2 \star |\tilde{H}(f)|^2 \right] \left|G(\alpha)\right|^2 d\alpha \tag{4.21}$$

In this report the bandwidth of the pre-filter is assumed to be much larger than the bandwidth of the post-filter. The convolution of the magnitude squared of the pre-filter transfer function with itself will yield a smooth, gently decreasing function with twice the bandwidth of the pre-filter. Since the bandwidth of the convolution is twice that of the pre-filter and thus much larger than the post-filter, the values of $|G(f)|^2$ can be assumed to be multiplied by a constant. The constant value is the value of the convolution of Equation 4.18 at a frequency of zero. This value is determined to be the square of the low-pass equivalent noise density, multiplied by twice the noise bandwidth of the pre-filter, $B_{npre}$. Twice the noise bandwidth of the pre-filter is approximately the value of the convolution of Equation 4.21 at a frequency of zero. Figure (2) shows the comparison between the magnitude squared transfer functions of the post-filters and the convolution normalized.

Since the convolution is approximately constant over the bandwidth of the post-filter, $Ry_3$ can be approximated as,

$$Ry_3 \approx 8No^2 B_{npre} \int_{-\infty}^{\infty} |G(\alpha)|^2 \, d\alpha \; . \qquad (4.22)$$

The integration is twice the noise equivalent bandwidth for a unity gain filter, thus

$$Ry_3 \approx 8No^2 \, B_{npre}[2B_n]$$

or

$$Ry_3 \approx 16No^2 \, B_{npre} B_n \qquad (4.23)$$

where $B_n$ is the noise equivalent bandwidth of the post-filter.

Thus for the large post-filter to pre-filter bandwidth ratio case, the output noise variance due to only noise at the input of the receiver can be approximated by (4.23).

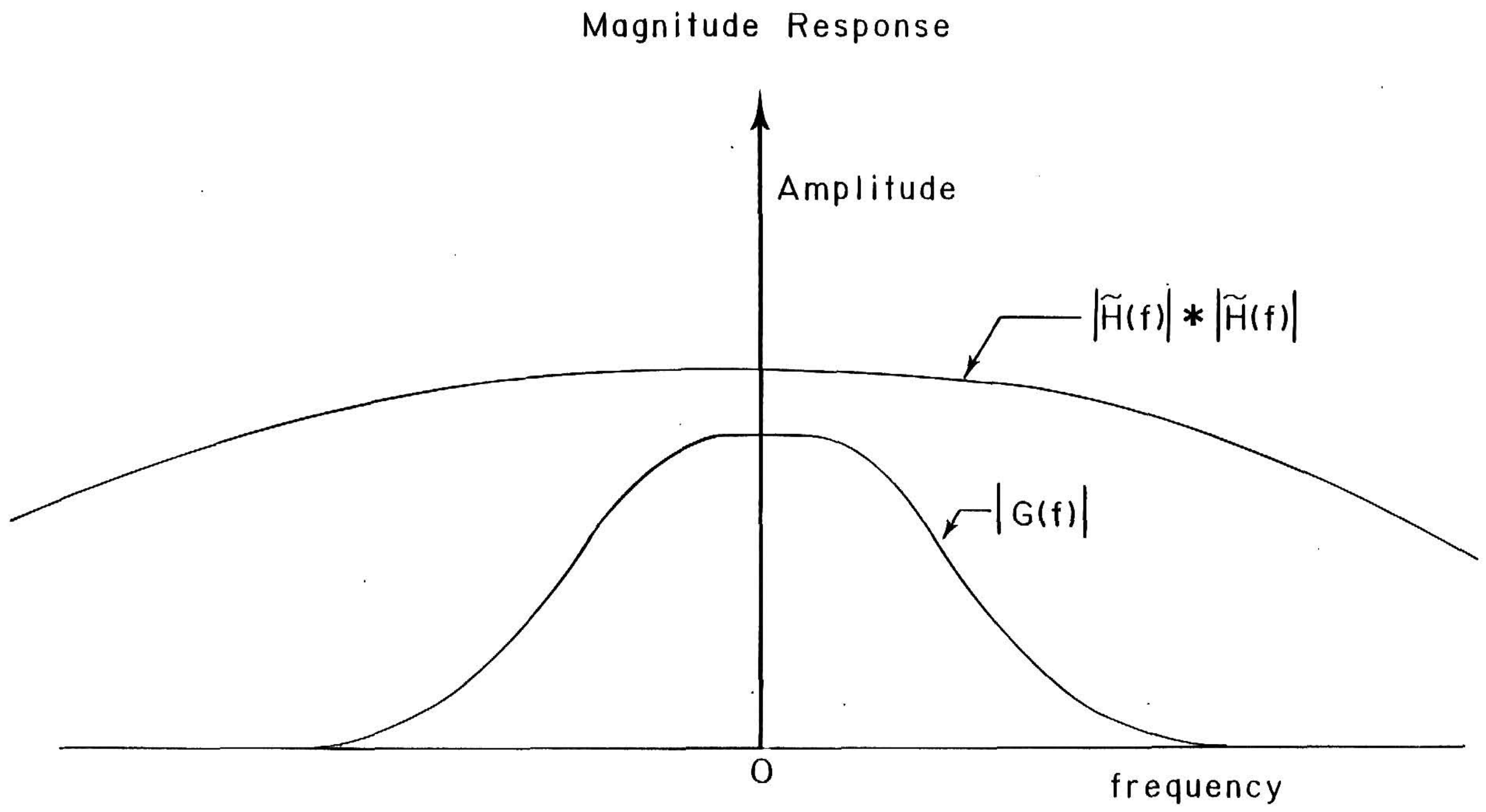


Figure 2. Comparison of Bandwidths in Noise Variance Equation

This approximation becomes more accurate as the bandwidth ratio increases. The acceptable value of bandwidth ratio depends on the desired degree of accuracy.

In this section, the variance was developed as the sum of two parts, a time varying term and a constant term. The time varying term is dependent on the input signal and is computed using numerical integration. The constant term depends only on the input noise process and is approximated using the large bandwidth ratio condition.

# V. PROBABILITY OF DETECTION CALCULATIONS

The goal of this work is to develop procedures for calculating probability of detection for wideband pulse receivers. The analysis which follows develops the necessary relationships for making the calculation based on the results obtained in previous sections for the output signal waveform and the mean and variance of the output noise process. A Gaussian assumption is a key factor in the development.

## Gaussian Assumption

The probability that a pulse type signal will be detected depends on how much the pulse is corrupted by the noise. Analysis of the noise at the output of the receiver can be difficult since a square-law detector is used in the receiver. However, this report deals with the case when the ratio of the bandwidth of the pre-filter to the bandwidth of the post-filter is large. The large ratio of bandwidth allows for the assumption that the process at the output of the receiver is Gaussian. This assumption is justified as follows.

Filter memory is inversly proportional to bandwidth. This property can be illustrated with a single-pole low-pass filter. The longer the decay time of the impulse response, the smaller bandwidth becomes. Applying this fact to the receiver means that the wideband pre-filter has a much shorter memory than the narrowband post-filter. Therefore, the output of the pre-filter has a relatively short time span between sample points that are for practical purposes statistically independent. These independent points in time are points far enough apart that the pre-filter response of an independent sample point has decayed to approximately zero by the time the next independent sample point is entering the pre-filter. These independent sample points are effectively added together by the longer memory of the post-filter. Each sample point at the output of the post-filter is thus a weighted sum of a large number of independent input samples.

The Central Limit Theorem states that the sum of independent random variables has a probability density function that approaches a Gaussian probability density function. Thus, each point in time at the output of the post-filter is a random variable which tends to have Gaussian characteristics as the bandwidth ratio grows large.

The Gaussian assumption is a significant simplifying assumption for this analysis. The question of how large a bandwidth ratio is needed for this assumption to hold depends on how much accuracy is needed in the calculations. In most cases, a bandwidth ratio of 100 or more will give sufficiently accurate results.

## Setting the Threshold

A signal pulse is detected when the post-filter output signal exceeds a preset threshold voltage. The output noise process may also exceed the thershold from time-to-time causing false alarms. It is customary to set the threshold for a specified false alarm rate. False alarm rate is the average number of times per second that the threshold is exceeded when only noise is present at the input of the receiver.

The false alarm rate can be used to determine the probability of the threshold being exceeded for a given sample with only the noise process at the input of the receiver. This is known as the probability of false alarm. The probability of the false alarm can be expressed as the average number of times a false alarms occurs per second divided by the number of independent opportunities for occurrence per second. In order to determine the independent opportunities, the post-filter output must be considered.

Let the equivalent noise bandwidth of the low pass post-filter be denoted as $B_n$. Approximating the post-filter as a rectangular filter with bandwidth $B_n$ will give a rectangular power spectral density when white noise is passed through the filter. The correlation function of the filter output, which is the transform of the power spectral density, will be a sinc function with zero crossings at intervals of $1/(2\ B_n)$ seconds. Points in time having this interval of time between them will be essentially uncorrelated. If the process is Gaussian, the points will also be independent. This approximation may be used to relate the probability of false alarm ($P_f$) to the false alarm rate, FAR, as

$$P_f = \frac{FAR}{2B_n} . \tag{5.1}$$

The probability of false alarm can also be determined using the probability density function (pdf) for a given sample.

Each sample at the output is a random variable. Using the Gaussian assumption gives the probability density function (pdf) for noise only at the output as

$$p(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(y-\overline{y})^2}{2\sigma^2}\right\} \tag{5.2}$$

where

$\sigma^2$ is $R_{y_3}$, the constant term of the output noise variance and $\overline{y_n}$ is the mean of the output noise.

The probability of false alarm is the probability that the output will exceed the threshold voltage. Expressing $P_f$ in terms of the Q function yields

$$P_f = P[y>V_t] = Q\left(\frac{V_t - \overline{y_n}}{\sqrt{R_{y_3}}}\right) \tag{5.3}$$

where $V_t$ is the threshold voltage. The Q function, $Q(\alpha)$, is defined as

$$Q(\alpha) = \int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx.$$

Combining the two equations for $P_f$ gives a means of determining the proper threshold voltage, $V_t$. It is necessary to make use of the readily available numerical algorithms for finding the inverse of $Q(\alpha)$. The computer program PDETGAUS determines the threshold voltage, $V_t$ by inverting Equation (5.3). The appropriate system parameters are taken into account by finding $\bar{y}_n$ and $R_{y_3}$ using the relationships (4.4) and (4.23), respectively.

## Calculating Probability of Detection

Once the threshold voltage has been set, the probability of detection, $P_d$, can be calculated.

In order to evaluate the probability of detection, the independent opportunities for the signal pulse to exceed threshold, i.e., the independent points in time when the signal is likely to exceed threshold, need to be determined. Formally, the difference in time between independent points in time at the output of the post-filter was determined to be approximately $1/(2\ B_n)$ seconds. For each of these independent samples, the probability that the sample voltage will be greater than the threshold needs to be determined, i.e.,

$$P[y>V_t].$$

If K independent opportunities occur for a given input pulse, the $P_d$ can be stated as

$$\begin{aligned} P_d &= P[\text{at least one sample exceeds } V_t] \\ &= 1 - P[\text{no samples exceed } V_t]. \end{aligned} \tag{5.4}$$

For ease of notation, define

$$P_i \triangleq P[y_i < V_t]$$

for the i-th independent sample. This probability can be expressed in terms of a probability density function (pdf) since the samples in time are random variables, viz.,

$$P_i = 1 - \int_{V_t}^{\infty} p(y_i)\, dy_i \ . \tag{5.5}$$

Using the Gaussian assumption, the pdf is

$$p(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(y_i - (y_{s_i} + \bar{y}_n))}{2\ \sigma_i^2}\right\} \tag{5.6}$$

where $\sigma_i^2$ is the variance of i-th sample of the output noise process, and $y_{s_i}$ is the output signal voltage at the $i^{th}$ sample time. In the procedures implemented in PDETGAUS, the needed values are obtained via numerical solution of Equations (3.7) and (4.8). The pdf is Gaussian and uses the fact that the output of the receiver, y(t), is the sum of the noise, $y_n(t)$, and signal, $y_s(t)$. The Q function can be used to express $P_i$ as,

$$P_i = 1 - Q\left(\frac{V_t - (y_{s_i} + \bar{y}_n)}{\sigma_i}\right) \tag{5.7}$$

Since the samples are statistically independent,

$$P[\text{no samples exceed } V_t] = P_1 \cdot P_2 \cdots P_K .$$

Applying this to $P_d$ gives

$$P_d = 1 - P_1 \cdot P_2 \cdots P_K . \tag{5.8}$$

In a numerical approach to determining the $P_d$, the independent $P_i$'s are determined by first finding the peak of the output pulse, which is used as one of the samples. Secondly, the other independent samples are determined by finding samples that are integer multiples of $1/(2\ B_n)$ seconds from the peak in both directions along the time axis. The desired output signal samples that are at an arbitary small percentage of the peak (for this report 10% is used) can be disregarded since their probability of not exceeding threshold is approximately one.

# VI. OPERATION OF PROGRAM

Section III provides a numerical method of calculating the output signal, and Section IV provides a numerical method of determining the nonstationary noise variance at the output. The necessary equations to compute the probability of detection were given in Section V. Using these previous results, the computer program PDETGAUS was developed. PDETGAUS is a menu driven program that computes the probability of detection under the assumption of a Gaussian noise process at the output of the receiver.

A listing of the Fortran program is supplied in Appendix B. A functional flow chart of the program is shown in Figure 3.

## Program Requirements and Limitations

The program PDETGAUS was developed on a Zenith Z-150 personal computer, (PC). This particular computer was equipped with 640 K-bytes of RAM, a 10 megabyte hard disk and an 8087 co-processor. The computer uses the MS DOS, version 2.11 operating system. PDETGAUS is written in Digital Research Fortran-77 language, however the compiled computer code may be executed on a compatible PC. Approximately 160 K-bytes of RAM are required to use PDETGAUS. Consequently, any IBM compatible PC that meets these memory requirements should be able to run PDETGAUS.

The approximate run times for PDETGAUS when creating new data files are 3 minutes for 128 sample points, 9 mintues for 256 sample points, and 38 minutes for 512 sample points. An important fact here is that the user has neither direct control over the the number of sample

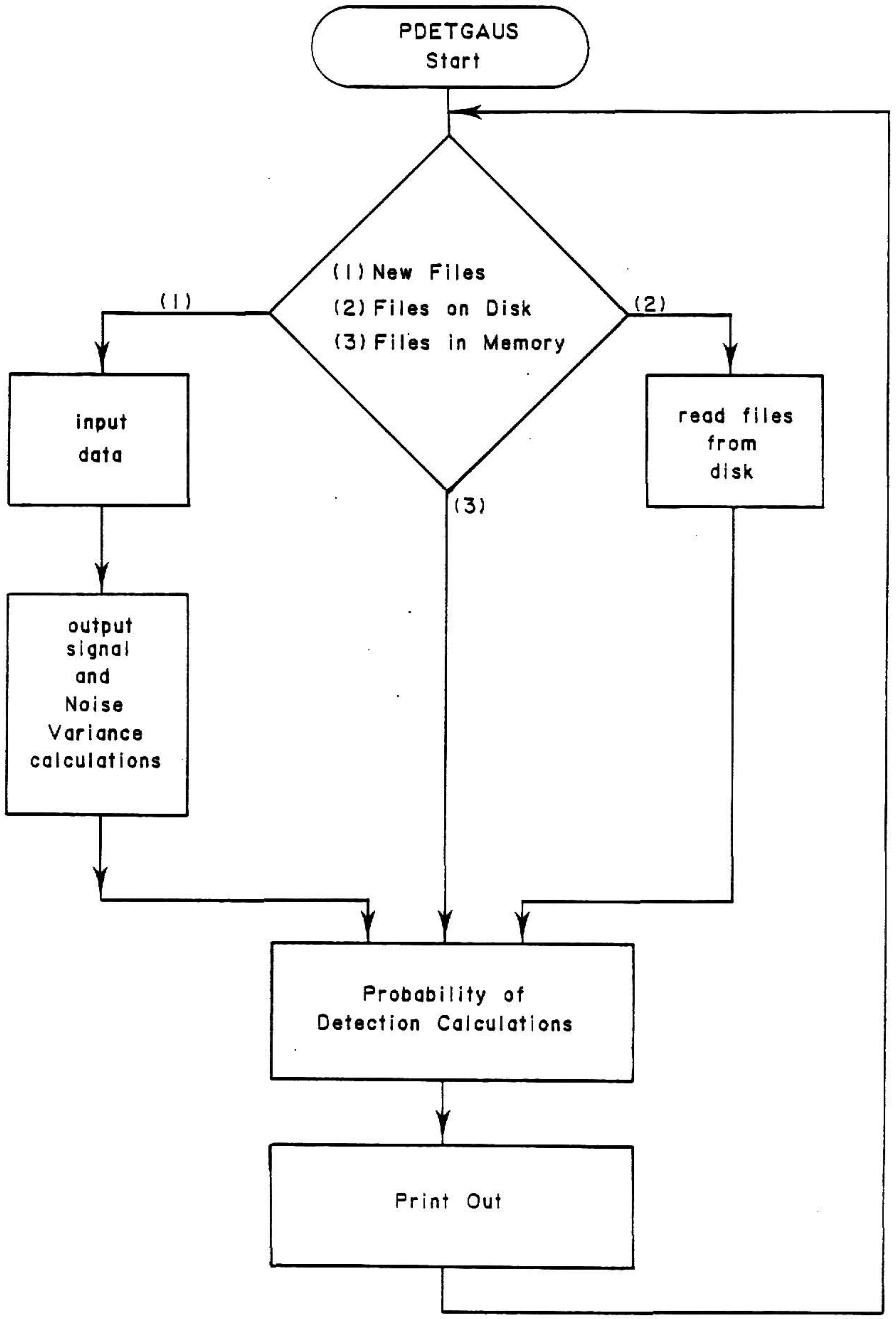


Figure 3. Functional Flow Diagram of PDETGAUS.

points, nor direct control of the fundamental frequency used to represent the signal pulses. PDETGAUS determines both of these parameters for the best determination of probability of detection.

The program was written for the case of a large bandwidth ratio between the pre-filter and the post-filter. PDETGAUS does not verify this rather ambiguous condition of a large bandwidth ratio. A possible but untested constraint is that the bandwidth ratio be 50 or greater. Larger ratios will lead to improved accuracy.

## Program Inputs

The program first asks for the source of the data files, input signal, output signal and noise variance needed to calculate probability of detection. The possible sources are: 1) A new set of files which will be created by the program, 2) files stored on disk, 3) files currently in memory. The option of creating a new set of files calls for parameter values to be input. These will be discussed later. The option of files stored on disk calls for the file names which were previously stored under PDETGAUS to be input so that the files can be read. The option of using files currently in memory is only offered after the program has calculated the probability of detection once. If one of the passive options, which are the files stored on disk and the files currently in memory, are chosen, the only parameters of the receiver that can be altered are the signal power to noise density ratio and the false alarm rate.

When creating new files, prompts will be displayed asking for the parameters of the pre-filter and post-filter. These parameters are the bandwidths and the number of poles in the filter response characteristics. In PDETGAUS, the available filter transfer functions are for unity gain Butterworth filters. Further prompts will ask for parameters of the desired input signal. PDETGAUS provides for the desired input signal to be a unity amplitude trapizoidal pulse with selectable parameters of rise time and 50% amplitude pulse width. The frequency offset, which is the frequency difference between the modulated input pulse and the center frequency of the band pass pre-filter may also be selected. Additional prompts are given for the signal power-to-noise density ratio, and the false alarm rate. Finally, the program will ask if any data files are to be stored on disk and if so, under what file names.

An important fact about the program is that it appends nine parameters of the receiver on the end of the input data file. This should be taken into account if the input data file is to be plotted, by not plotting the last six values of the file.

After a probability of detection has been computed, the option of exiting the program is offered. Also the user can terminate the program at any input prompt by entering CONTROL-C.

## Program Outputs

PDETGAUS must have access to a printer, since its main output is a hardcopy of input parameters, program related parameters and the probability of detection. The program related parameters are the fundamental frequency and the number of samples. The fundamental frequency is the fundamental frequency of the periodic extension of the signal pulse, and the number of samples is the number of discrete samples used to represent the input signal. These computer implemented parameters are determined under the constraints that the bandwdith of the post-filter and the input pulse to the pre-filter must be accurately represented. PDETGAUS also outputs the probability of detection on the monitor.

## Example Runs

The following are example runs of PDETGAUS. These examples show all important user prompts as well as sample user responses.

To run PDETGAUS, insert the disk containing the PDETGAUS.EXE file into disk drive A, then type A:PDETGAUS and return. The monitor will display a warning message about data entry errors. Following the warning message the file source menu will be displayed, as shown in Figure 4.

```
DO YOU WISH TO CALCULATE THE PROBABILITY OF DETECTION FROM:

1.   NEW INPUT SIGNAL, OUTPUT SIGNAL AND VARIANCE FILES?

2.   PREVIOUSLY DETERMINED INPUT SIGNAL, OUTPUT SIGNAL AND
     VARIANCE FILES STORED ON DISK?

ENTER NUMBER CORRESPONDING TO CHOICE.
```

Figure 4. PDETGAUS File Source Menu.

## Example 1

The first example deals with creating of new data files. Enter (1) from the file source menu. Figure 5 shows the prompts and responses, after a warning message is displayed about bandwidth ratio.

```
ENTER FOR PRE-FILTER THE:

          RF BANDWDITH (in MHz)        80

          NO. OF POLES (1 to 4 poles)  4

ENTER FOR POST-FILTER THE:

          BANDWDITH (in MHz)           1

          NO. OF POLES (1 to 4 poles)  1

ENTER FOR THE INPUT PULSE:

          50% AMP. PULSE WIDTH (in micro sec)  .5

          RISETIME (in nano sec)               0

          FREQUENCY OFFSET (Hz)                0

CHOOSE FORM OF SIGNAL POWER TO NOISE DENSITY RATIO:

          1)  as the value of the ratio, Ps/No

          2)  as SNR and the NOISE REFERENCE BANDWDITH

ENTER NUMBER CORRESPONDING TO CHOICE   1

ENTER SIGNAL POWER TO NOISE DENSITY RATIO,
Ps/No (in dB)   80

ENTER FALSE ALARM RATE
(false alarms per second)   1
```

Figure 5. Parameter Prompts and Responses

After the input of the system parameters, the program prompts the user to decide for each of the three data files, whether the file is to be saved or not. PDETGAUS provides for the storing of ASCII data files on disk. If the user does not wish to save a data file, he or she enters "no". If the user does wish to save a data file, he or she enters "yes". The program will then prompt for the filename of the data which must be entered in the form, "filename.DAT". Prompts for storing of data are shown in Figure 6. In the example the default disk is

```
DO YOU WISH TO SAVE THE INPUT SIGNAL?  YES

           enter file name IN.DAT

DO YOU WISH TO SAVE THE OUTPUT SIGNAL?  YES

           enter file name OUT.DAT

DO YOU WISH TO SAVE THE VARIANCE?  YES

           enter file name VAR.DAT
```

Figure 6. Prompts for Storing of Data Files

assumed to be C, and all files will be stored on disk C. If another disk, say disk A, is to store the data files, enter the filename in the form "A:filename.DAT".

PDETGAUS then displays the approximate amount of time for the probability of detection to be calculated. After the calcualtion time has passed, the output will be printed, as shown in Figure 7.

The user then has the option to run the program again. In this case, the file source menu will include an added option of using the data already in memory.

## Example 2

PDET allows for data files to be stored and then later retrieved. From the file source menu, choosing option 2 will cause the program to prompt for data filenames as shown in Figure 8.

Note that the files are assumed to be on the default disk. If they are not on the default disk, the appropriate disk prefix must be included before the file name, for example, "A:IN.DAT".

```
Square Law Detector ⟶ Probability of Detection Gaussian Assumption


              Butterworth Pre-Filter
                 RF Bandwidth            :  80.0000      MHz
                 Number of Poles         : 4


              Butterworth Post-Filter
                 Bandwidth               :  1.00000      MHz
                 Number of Poles         : 1


              Input Pulse
                50% Amplitude Width      : 0.500000E-06 s
                Rise Time                :  .000000      s
                Frequency Offset         :  .000000      Hz


              Miscellaneous
                Fundamental Frequency    : 100000.      Hz
                Number of Samples        :  256
                Signal/Noise Density     : 80.0000      dB
                False Alarm Rate         : 1.00000      FA/s


                    ********** RESULT **********

                 Probability of Detection :  80.93%
```

Figure 7. Printout of PDETGAUS.

```
ENTER FILE NAME THE CONTAINS:

    input signal    IN.DAT

    output signal   OUT.DAT

    variance file   VAR.DAT
```

Figure 8. Prompts for Retrieving Files from Disk.

Next, the program will ask for the signal power-to-noise density ratio and the false alarm rate and then calculate the probability of detection. Thus, the only parameters that can be varied when retrieving data from the disk are the signal power-to-noise density ratio and the false alarm rate. The printed output will be in the same form as Figure 7.

Using printed outputs, curves such as Figures 9, 10, and 11 can be sketched. The curves are probability paper graphs of probability of detection versus signal power-to-noise density ratio. For Figures 9, 10, and 11 the bandwidth ratios are 40, 60, and 100 respectively. All other parameters for the three cases are fixed.

Probability of Detection Versus Signal Power-to-Noise Density

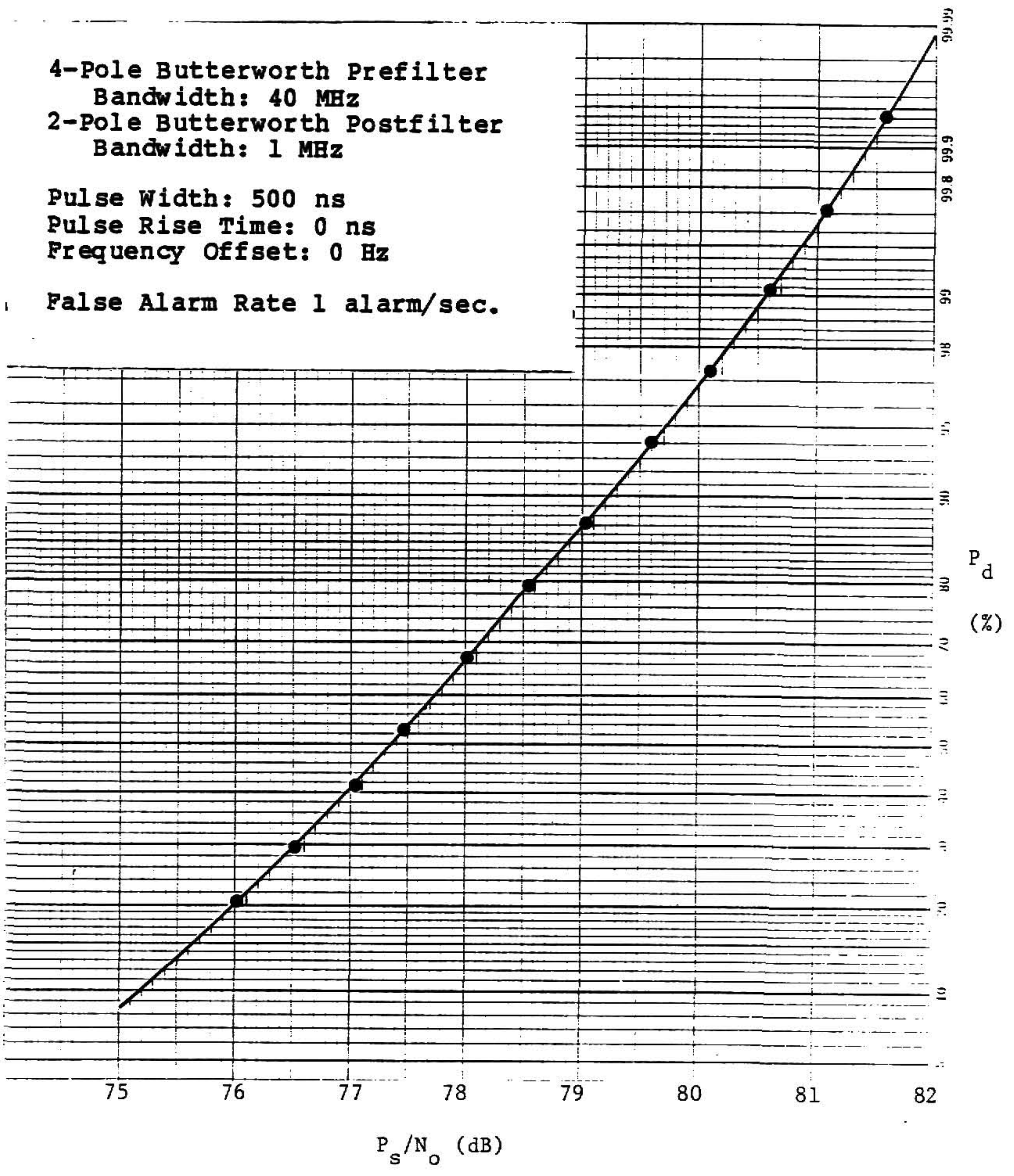


Figure 9.

Probability of Detection Versus Signal Power-to-Noise Density Ratio

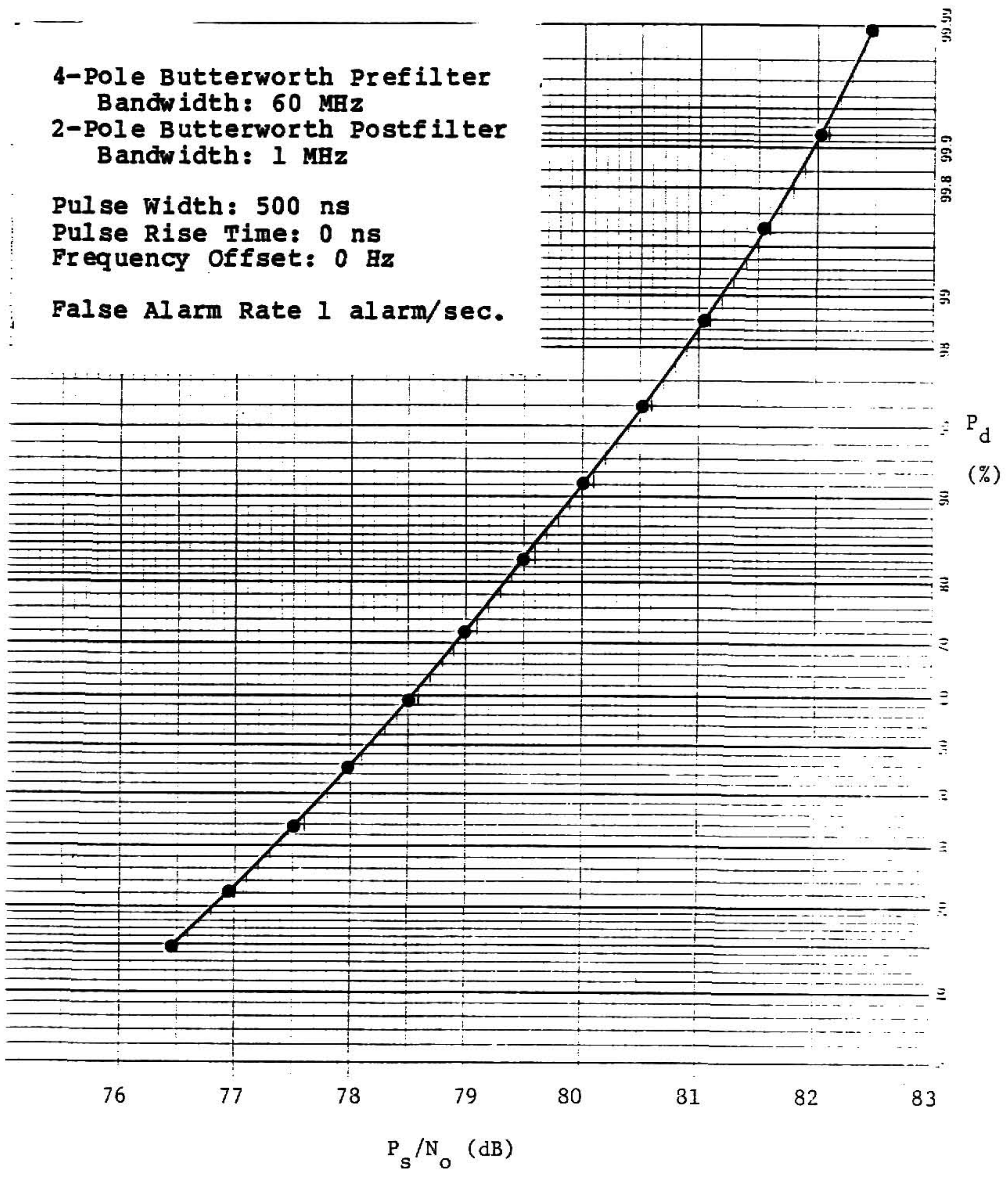


Figure 10.

Probability of Detection Versus Signal Power-to-Noise Density

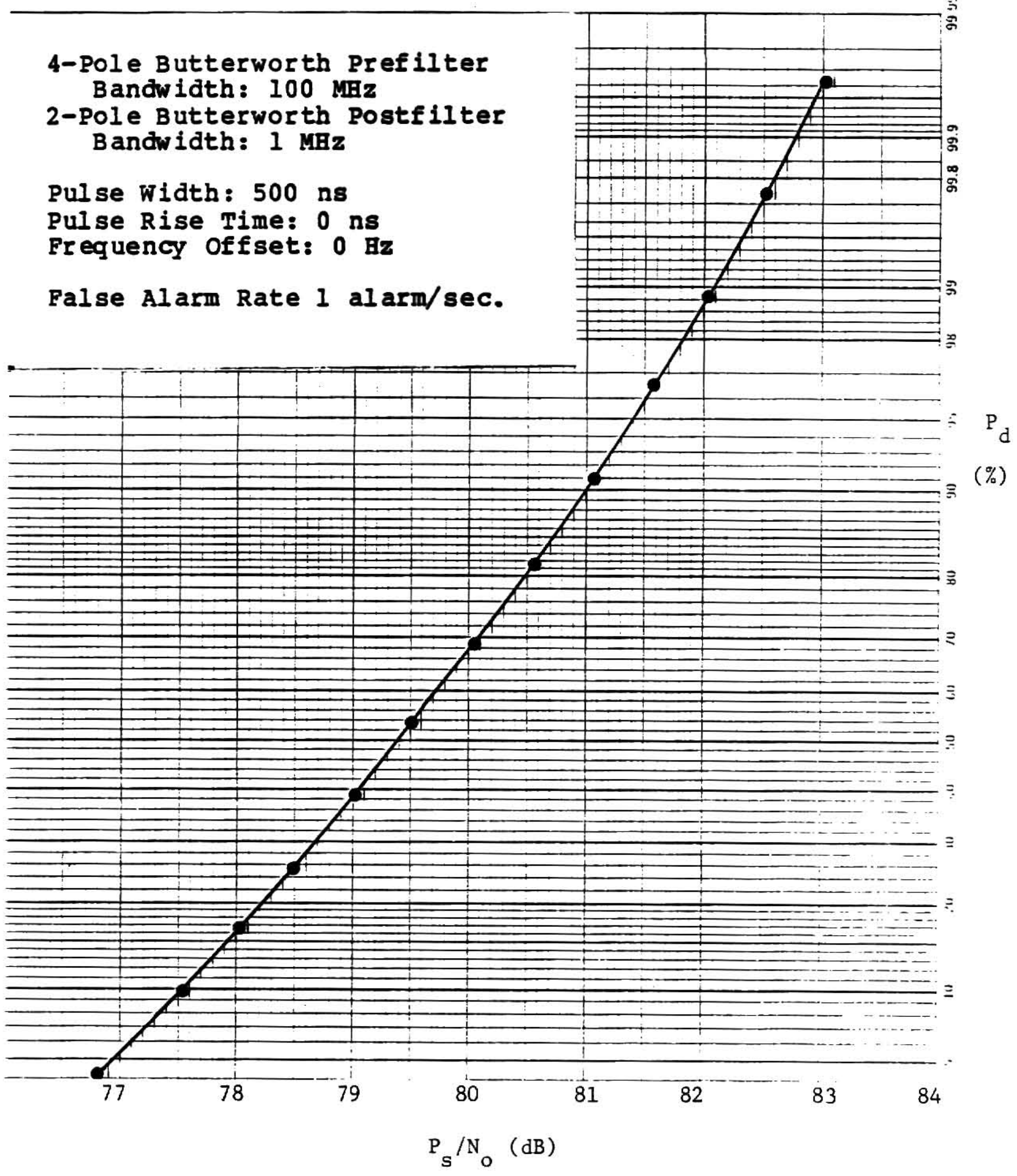


Figure 11.

## VII. CONCLUSIONS

This report deals with determining the probability of detection for a square-law receiver, under the constraint that the pre-filter to post-filter bandwidth ratio is large. The assumption that the output noise process of the receiver is Gaussian was justified by imposing this constraint. The main purpose of this report was to implement a computer program that numerically sloves for probability of detection. Except for the assumption of Gaussian statistics, the results are unconstrained and general. The procedures used take into account the important signal and filter parameters when making the calculation. The program should prove to be a useful tool for evaluating the performance of wideband pulse receivers.

## ACKNOWLEDGEMENTS

I wish to express my appreciation to my parents, friends and the members of my graduate committee, Dr. D. R. Hummels, Dr. S. A. Dyer and Dr. J. E. Boyer, Jr. A special thanks goes to Dr. B. K. Harms for his support.

# APPENDIX A

## MEAN AND AUTOCORRELATION OF OUTPUT NOISE

The noise component at the output of the square-law detector was found in Section II as

$$v_n(t) = \tilde{p}(t)\tilde{w}^*(t) + \tilde{p}^*(t)\tilde{w}(t) + |\tilde{w}(t)|^2. \tag{A.1}$$

Since there are product terms involving signal and noise in $v_n(t)$, it is a nonstationary noise process. The statistical properties of the noise process need to be determined in order to calculate the probability of detection for the input signal. The development which is taken from reference [3]. The first step is finding an expression for the mean of the output noise.

### The Mean of the Output Noise

The mean of the noise process is

$$\overline{v_n(t)} = E\{v_n(t)\}. \tag{A.2}$$

Using the property of linearity for expectation, the mean can be expressed as

$$\overline{v_n(t)} = E\{\tilde{p}(t)\tilde{w}^*(t)\} + E\{\tilde{p}^*(t)\tilde{w}(t)\} + E\{|\tilde{w}(t)|^2\} .$$

The signal components are deterministic; thus, for any given time, they are constants which can be taken outside the expectations, yielding

$$\bar{v}_n(t) = \tilde{p}(t)E\{\tilde{w}^*(t)\} + \tilde{p}^*(t)\ E\{\tilde{w}(t)\} + E\{|\tilde{w}(t)|^2\} . \tag{A.3}$$

The noise input to the receiver is zero-mean. Linear filtering of the noise, which yields $\tilde{w}(t)$, does not affect this property. Thus

$$E\{\tilde{w}(t)\} = E\{\tilde{w}^*(t)\} = 0,$$

which gives

$$\overline{v_n(t)} = E\{|\tilde{w}(t)|^2\}$$
$$= E\{\tilde{w}(t)\tilde{w}^*(t)\}. \tag{A.4}$$

The autocorrelation function of $\tilde{w}(t)$ is defined as

$$R_{\tilde{w}}(t,s) = E\{\tilde{w}(t)\tilde{w}^*(s)\}. \tag{A.5}$$

Thus, the mean of the noise component at the output of the square-law detector can be expressed as

$$\overline{v_n(t)} = R_{\tilde{w}}(t,t). \tag{A.6}$$

$\tilde{w}(t)$ is a stationary noise process since it is a linearly filtered stationary process. For stationary processes, the autocorrelation function is independent of absolute time, hence

$$\overline{v_n(t)} = R_{\tilde{w}}(0) = \overline{v_n}\ , \tag{A.7}$$

which is a constant.

Since $\overline{v_n(t)}$ is just the dc component at the detector output, it follows directly that the mean of the output noise, $y_n(t)$, is given by

$$\overline{y_n(t)} = R_{\tilde{w}}(0)\ G(0) = y_n\ . \tag{A.8}$$

## The Autocorrelation Function of the Output Noise

In determining the autocorrelation of the noise at the output of the receiver, the output of the square-law detector must be examined first. The noise output of the square-law detector, $v_n(t)$, is nonstationary, so the autocorrelation function is expressed as

$$R_{v_n}(t,s) = E\{v_n(t)\ v_n^*(s)\}\ . \tag{A.9}$$

Substitution, using (A.1) gives

$$R_{v_n}(t,s) = E\{[\tilde{p}(t)\tilde{w}^*(t) + \tilde{p}^*(t)\tilde{w}^*\tilde{w}(t)\tilde{w}^*(t)] \cdot [\tilde{p}^*(s)\tilde{w}(s) + \tilde{p}(s)\tilde{w}^*(s) + \tilde{w}^*(s)\tilde{w}(s)]\}.$$

Expanding the product results in

$$\begin{aligned} R_{v_n}(t,s) = E\{&\tilde{p}(t)\tilde{w}^*(t)\tilde{p}^*(s)\tilde{w}(s) + \tilde{p}(t)\tilde{w}^*(t)\tilde{p}(s)\tilde{w}^*(s) \\ &+ \tilde{p}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s) + \tilde{p}^*(s)\tilde{w}(t)\tilde{p}^*(s)\tilde{w}(s) \\ &+ \tilde{p}^*(t)\tilde{w}(t)\tilde{p}(s)\tilde{w}^*(s) + \tilde{p}^*(t)\tilde{w}(t)\tilde{w}^*(s)\tilde{w}(s) \\ &+ \tilde{w}(t)\tilde{w}^*(t)\tilde{p}^*(s)\tilde{w}(s) + \tilde{w}(t)\tilde{w}^*(t)\tilde{p}(s)\tilde{w}^*(s) \\ &+ \tilde{w}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s)\} . \end{aligned}$$

Since expectation is a linear operation, the individual product terms can be examined separately. An important fact used to simplify this expression is the fact that the expectation of a product having an odd number of zero means Gaussian random variables is always zero [2]. In the product, terms $\tilde{w}(t)$ and $\tilde{w}(s)$ are zero-mean random variables for a given t and s, respectively. Thus, evaluating the product terms with an odd number of noise processes yields

$$\begin{aligned} R_{v_n}(t,s) &= E\{\tilde{p}(t)\tilde{w}^*(t)\tilde{p}^*(s)\tilde{w}(s)\} \\ &+ E\{\tilde{p}(t)\tilde{w}^*(t)\tilde{p}(s)\tilde{w}^*(s)\} \\ &+ E\{\tilde{p}^*(t)\tilde{w}(s)\tilde{p}^*(s)\tilde{w}(s)\} \\ &+ E\{\tilde{p}^*(t)\tilde{w}(t)\tilde{p}(s)\tilde{w}^*(s)\} \\ &+ E\{\tilde{w}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s)\} . \end{aligned}$$

The signal term $p(\tau)$, is deterministic for any given index. Therefore they can be removed from the expectations, giving

$$\begin{aligned} R_{v_n}(t,s) &= \tilde{p}(t)\tilde{p}^*(s)\ E\{\tilde{w}^*(t)\tilde{w}(s)\} \\ &+ \tilde{p}(t)\tilde{p}(s)\ E\{\tilde{w}^*(t)\tilde{w}^*(s)\} \\ &+ \tilde{p}^*(t)\tilde{p}^*(s)\ E\{\tilde{w}(t)\tilde{w}(s)\} \\ &+ \tilde{p}^*(t)\tilde{p}(s)\ E\{\tilde{w}(t)\tilde{w}^*(s)\} \\ &+ E\{\tilde{w}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s)\} . \end{aligned} \tag{A.10}$$

Examining the noise process as

$$\tilde{w}(t) = \alpha(t) + j\beta(t),$$

where $\alpha(t)$ and $\beta(t)$ are zero-mean statistically independent random processes having identical correlation functions yields,

$$\begin{aligned} E\{\tilde{w}(t)\tilde{w}(s)\} &= E\{\alpha(t)\alpha(s)\} + j\, E\{\alpha(t)\}\, E\{\beta(s)\} \\ &\quad + j\, E\{\beta(t)\}\, E\{\alpha(s)\} - E\{\beta(t)\beta(s)\} . \end{aligned}$$

Taking into account the zero-mean and identical autocorrelation leads to

$$E\{\tilde{w}(t)\tilde{w}(s)\} = R_{\alpha}(t,s) - R_{\beta}(t,s) = 0. \tag{A.11a}$$

Likewise, it can be shown that

$$E\{\tilde{w}^*(t)\tilde{w}^*(s)\} = 0. \tag{A.11b}$$

Thus, the autocorrelation can be reduced to,

$$\begin{aligned} R_{v_n}(t,s) &= \tilde{p}(t)\tilde{p}^*(s)\, E\{\tilde{w}^*(t)\tilde{w}(s)\} \\ &\quad + \tilde{p}^*(t)\tilde{p}(s)\, E\{\tilde{w}(t)\tilde{w}^*(s)\} \\ &\quad + E\{\tilde{w}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s)\}. \end{aligned}$$

The following terms as autocorrelations

$$E\{\tilde{w}^*(t)\tilde{w}(s)\} = R_{\tilde{w}}(s,t)$$

$$E\{\tilde{w}(t)\tilde{w}^*(s)\} = R_{\tilde{w}}(t,s)$$

can be expressed as autocorrelation, using the fact that

$$R_{\tilde{w}}(t,s) = R_{\tilde{w}}^*(s,t).$$

The autocorrelation reduces to

$$\begin{aligned} R_{v_n}(t,s) &= [\tilde{p}(t)\tilde{p}^*(s) + \tilde{p}^*(t)\tilde{p}(s)]\, R_{\tilde{w}}(t,s) \\ &\quad + E\{\tilde{w}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s)\} . \end{aligned} \tag{A.12}$$

The last term of the autocorrelation can be expressed as [2],

$$\begin{aligned} E\{\tilde{w}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s)\} &= E\{\tilde{w}(t)\tilde{w}^*(t)\}\, E\{\tilde{w}^*(s)\tilde{w}(s)\} \\ &\quad + E\{\tilde{w}(t)\tilde{w}^*(s)\}\, E\{\tilde{w}^*(t)\tilde{w}(s)\} \\ &\quad + E\{\tilde{w}(t)\tilde{w}(s)\}\, E\{\tilde{w}^*(t)\tilde{w}^*(s)\} \end{aligned}$$

From previous work performed in this appendix, the expectation can be reduced to

$$E\{\tilde{w}(t)\tilde{w}^*(t)\tilde{w}^*(s)\tilde{w}(s)\} = R^2_{\tilde{w}}(0) + |R_{\tilde{w}}(t,s)|^2$$

Substituting this into (A.14) gives

$$R_{v_n}(t) = [\tilde{p}(t)\tilde{p}^*(s) + \tilde{p}^*(t)\tilde{p}(s)]\ R_{\tilde{w}}(t,s) + R^2_{\tilde{w}}(0) + |R_{\tilde{w}}(t,s)|^2 \qquad (A.13)$$

The autocorrelation of the noise output noise process is [2];

$$R_{y_n}(t,s) = \iint_{-\infty}^{\infty} R_{v_n}(t,s)\ g(t-\alpha)\ g(s-\beta)\ d\alpha d\beta \qquad (A.14)$$

Substituting equation (A.13) and breaking up the integration gives the autocorrelation as,

$$R_{y_n}(t,s) = R_{y_1}(t,s) + R_{y_2}(t,s) + R_{y_3}(t,s) + R_{y_4}(t,s) \qquad (A.15)$$

where

$$R_{y_1}(t,s) = \iint_{-\infty}^{\infty} \tilde{p}(\alpha)\ \tilde{p}^*(\beta)\ R^*_{\tilde{w}}(\alpha,\beta)\ g(t-\alpha)\ g(s-\beta)\ d\alpha d\beta\ ,$$

$$R_{y_2}(t,s) = \iint_{-\infty}^{\infty} \tilde{p}^*(\alpha)\tilde{p}(\beta)\ R_{\tilde{w}}(\alpha,\beta)\ g(t-\alpha)\ g(s-\beta)\ d\alpha d\beta\ ,$$

$$R_{y_3}(t,s) = \iint_{-\infty}^{\infty} R^2_{\tilde{w}}(0)\ g(t-\alpha)\ g(s-\beta)\ d\alpha d\beta\ ,$$

and

$$R_{y_4}(t,s) = \iint_{-\infty}^{\infty} |R_{\tilde{w}}(\alpha,\beta)|^2\ g(t-\alpha)\ g(s-\beta)\ d\alpha d\beta\ .$$

$R_{\tilde{w}}(\alpha,\beta)$ can be written as $R_{\tilde{w}}(\alpha-\beta)$ as it appears at the output of a linear filter wwith input of at least a wide sense stationary Gaussian random process. By using this fact, $R_{y_4}(t,s)$ can be shown that

$$R_{y_4}(t,s) = R^2_{\tilde{w}}(0)\ G^2(0). \qquad (A.16)$$

Since $R_{y_1}(t,s)$ and $R_{y_2}(t,s)$ are complex conjugates, the final form of the autocorrelation of the noise process at the output of the receiver is

$$R_{y_n}(t,s) = R_{y_1}(t,s) + R_{y_1}{}^*(t,s) + R_{y_3}(t,s) + R^2_{\tilde{w}}(0)\ G^2(0), \qquad \text{(A.17)}$$

where

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

and

$$R_{y_3}(t,s) = \iint_{-\infty}^{\infty} |R_{\tilde{w}}(\alpha-\beta)|^2\ g(t-\alpha)\ g(t-\beta)\ d\alpha d\beta\ .$$

## APPENDIX B

## COMPUTER PROGRAM: PDETGAUS

```
      PROGRAM PDETGAUS
C***********************************************************************
C                                                                      *
C                                                                      *
C      PDETGAUS -> PROBABILITY OF DETECTION (GAUSSIAN ASSUMPTION)      *
C                  PROGRAM                                             *
C                                                                      *
C      FORTRAN77 SOURSE FILENAME:                  PDETGAUS.EXE        *
C                                                                      *
C      DEPARTMENT OF ELECTRICAL ENGINEERING        KANSAS STATE        *
C                                                  UNIVERSITY          *
C                                                                      *
C      REVISION                  DATE              PROGRAMMER          *
C      --------                  ----              ----------          *
C                                                                      *
C        1.0                     1-10-86           SCOTT BRILES        *
C                                                                      *
C***********************************************************************
C
      CHARACTER*80 NAME1,NAME2,NAME3,NAME4,CSTRNG,NAMET1
      CHARACTER*48 QUEST1,QUEST2,QUEST3,QUEST4
      INTEGER ERROR1,ERROR2,ERROR3,CHOICE,LENGTH,LNGTH1
     &         INPDIS,YMAXNO,TOTSIPTS,NOFP,LNGTH2,MARK,CHOIC2
     &         NPOINT,NPPREF,NPPOSF,RUNTME
      LOGICAL SAVEI,SAVEO,SAVEV,AGAIN,CK
      REAL NFREQ,TRAP,F0,FILE(0:600),N0,OFFSET,PI,ARG,TEMP1
     &     FAR,ALPHAPF,VT,BNLP,ARB,TEMP
     &     DELTAF,YMAX,ARBLVL,TIME1,TPS,FREQ
     &     SD,APEAK,PS,FREQ,BN,YS,PSN0DB,PSN0
     &     RY3,BNPRE,YNMEAN,SIG,PF,QINVER,VARTV,SF
     &     BPF,BLP,RBPF1
      REAL*8 PJ,PLTVT,ALPHA,PROBD,PDPC,QFUNCT,DTEMP
      COMPLEX P(0:512),Y(0:512),CXARRY(0:512)
      COMPLEX BUTTER
      COMPLEX PE,RY1(0:513),LITG(0:512),CTEMP,CTEMP1
      COMPLEX TERM,TERM1,DCMPX,RY1W,RY1H,SWIG(0:512)
      COMMON /ASS/ P,Y,CXARRY,LITG,SWIG,RY1
      MARK = 0
C
C######################################################################
C
C      INPUT OF DATA FROM TERMINAL
C
C######################################################################
C
      CALL CLEARCRT(20)
      PRINT*,' '
      PRINT*,'******************  WARNING  ******************'
      PRINT*,' '
      PRINT*,'DUE TO THE NATURE OF THIS PROGRAMS RUN-TIME '
      PRINT*,'ENVIRONMENT, NO CORRECTIONS CAN BE MADE TO DATA'
      PRINT*,'AFTER IT HAS BEEN TYPED IN. '
      PRINT*,'DON''T PANIC!  YOU CAN MAKE CORRECTIONS LATER IN THE'
      PRINT*,'PROGRAM.'
      PRINT*,' '
      PAUSE '  TYPE ANY KEY TO CONTINUE'
1     CALL CLEARCRT(20)
      PRINT*,'      DO YOU WISH TO CALCULATE THE PROBABILITY OF'
      PRINT*,'      DETECTION FROM:'
      PRINT 603
603   FORMAT(' ')
      PRINT*,'         1.  NEW INPUT SIGNAL, OUTPUT SIGNAL '
      PRINT*,'             AND VARIANCE FILES ?'
      PRINT 603
      PRINT*,'         2.  PREVIOUSLY DETERMINED INPUT SIGNAL,'
      PRINT*,'             OUTPUT SIGNAL AND VARIANCE FILES'
```

```
        PRINT*,'            STORED ON DISK ?'
        PRINT 603
        IF (MARK.EQ.0) GOTO 8
        PRINT*,'         3.  DATA CURRENTLY IN MEMORY ?'
        PRINT 603
8       CONTINUE
        PRINT 610
610     FORMAT(1X,5X, 'ENTER NUMBER CORRESPONDING TO CHOICE   ')
        READ*, CHOICE
        PRINT 603
        GOTO (2,3,4) CHOICE
C
2       PRINT 603
        PRINT*,'      ********* CAUTION *********'
        PRINT*,'      FOR GOOD RESULTS AT LOW SNR, THE RATIO'
        PRINT*,'      OF PRE-FILTER/POST-FILTER BANDWIDTH'
        PRINT*,'      SHOULD BE 100 OR MORE.'
        PRINT 603
        PRINT*,'      ENTER FOR PRE-FILTER THE:'
        PRINT 603
        PRINT*,'            RF BANDWIDTH (in MHz)   '
        READ*, RBPF1
             RBPF = RBPF1 * 10.**6
             BPF = RBPF/2.0
        PRINT 603
        PRINT*,'            NO. OF POLES (1 to 4 poles)    '
        READ*, NPPREF
C
        PRINT 603
        PRINT*,'      ENTER FOR POST-FILTER THE:'
        PRINT 603
        PRINT*,'            BANDWIDTH (in MHz)   '
        READ*, BLP2
             BLP = BLP2 * 10.**6
        PRINT 603
        PRINT*,'            NO. OF POLES (1 to 4 poles)    '
        READ*, NPPOSF
        PRINT 603
C
        PRINT*,'      ENTER FOR THE INPUT PULSE:'
        PRINT 603
        PRINT*,'             50% AMP. PULSE WIDTH (in micro sec)    '
        READ*,  TEMP
             TAU = TEMP * 10.**(-6)
        PRINT 603
        PRINT*,'            RISETIME (in nano sec.)   '
        READ*,  TEMP
             RISTME = TEMP * 10.**(-9)
        PRINT 603
        PRINT*,'            FREQUENCY OFFSET (Hz)     '
        READ*,  OFFSET
        PRINT 603
        PRINT*,'PRE-FILTER'
        PRINT*,'      Bandwidth ',RBPF1,' MHz'
        PRINT*,'      No. of poles ',NPPREF
        PRINT*,'POST-FILTER'
        PRINT*,'      Bandwidth ',BLP2,' MHz'
        PRINT*,'      No. of poles ',NPPOSF
        PRINT*,'INPUT PULSE'
        PRINT*,'      Pulse width ',TAU,' s'
        PRINT*,'      Risetime ',RISTME,' s'
        PRINT*,'      Freq. offset ',OFFSET,' Hz'
        PRINT 603
        CALL QREPLY('     DO YOU WISH TO CORRECT DATA ?           ',CK)
        IF (CK) GOTO 1
        PRINT 603
```

```
C
C   Determine fundamental freq and number of points
        IF ((1.0/TAU).GE.BLP) THEN
                F0 = BLP / 10.0
        ELSE
                F0 = 1.0 / (10.0 * TAU)
        ENDIF
        NPOINT = 128
        SMPTME = 1.0 / (F0 * NPOINT)
        IF (SMPTME.GT.(TAU/10.)) THEN
                NPOINT = 256
                SMPTME = 1.0 / (F0 * NPOINT)
                IF (SMPTME.GT.(TAU/10.)) THEN
                        NPOINT = 512
                        SMPTME = 1.0 / (F0 * NPOINT)
                ENDIF
        ENDIF
        GOTO 4
C
C------------------------------------------------------
C
C       READ DATA FILE FROM DISK IF CHOSEN
C
C------------------------------------------------------
3       PRINT*,'     ENTER FILE NAME THAT CONTAINS:'
        PRINT 603
        CSTRNG = 'input signal     '
        CALL RNAME(CSTRNG,NAME1)
        CSTRNG = 'output signal    '
        CALL RNAME(CSTRNG,NAME2)
        CSTRNG = 'variance file    '
        CALL RNAME(CSTRNG,NAME3)
        PRINT 603
        CALL QREPLY('     DO YOU WISH TO CORRECT DATA ?           ',CK)
        IF (CK) GOTO 1
        CALL RDATA(LENGTH,FILE(0),NAME1)
C
        OFFSET = FILE(LENGTH - 1)
        RISTME = FILE(LENGTH - 2)
        TAU = FILE(LENGTH - 3)
        BLP = FILE(LENGTH - 4)
                BLP2 = BLP / (10.0 ** 6)
        NPPOSF = IFIX(FILE(LENGTH - 5))
        SMPTME = FILE(LENGTH - 6)
        RBPF = FILE(LENGTH - 7)
                RBPF1 = RBPF / (10.0 ** 6)
            BPF = RBPF / 2.0
        NPPREF = IFIX(FILE(LENGTH - 8))
        F0 = FILE(LENGTH - 9)
        CALL RDATA(LNGTH1,FILE(0),NAME2)
        NPOINT = LNGTH1
        DO 7 I=0,NPOINT-1
           Y(I) = CMPLX(FILE(I),0.0)
7       CONTINUE
C
        CALL RDATA(LNGTH2,FILE(0),NAME3)
        N0 = 4.E-9
        RY3 = 16.0 * (N0 ** 2) * BNLP * BNPRE
C
C---------------------------------------------
C
4       PRINT 603
        PRINT*,'     CHOOSE FORM OF SIGNAL POWER TO NOISE DENSITY'
        PRINT*,'     RATIO:'
        PRINT 603
        PRINT*,'        1)  as the value of the ratio, Ps/No'
        PRINT 603
        PRINT*,'        2)  as SNR and the NOISE REFERENCE BANDWIDTH'
        PRINT 603
        WRITE(6,610)
        READ*,  CHOIC2
        IF (CHOIC2.EQ.1) THEN
           PRINT 603
           PRINT*,'      ENTER SIGNAL POWER TO NOISE DENSITY '
           PRINT*,'      RATIO, Ps/No    (in dB)    '
           READ*,  PSNODB
              PSNO = 10.0 ** (PSNODB / 10.0)
        ELSE
           PRINT 603
           PRINT*,'      ENTER SNR (dB)     '
           READ*, SNR
           PRINT 603
           PRINT*,'      ENTER THE BANDWIDTH THAT THE SNR IS'
           PRINT*,'      REFERENCED TO (MHz)     '
           READ*, BREF1
              BREF = BREF1 * 10.**6
              PSNO = BREF * (10.0 ** (SNR / 10.0))
           PSNODB = 10.0 * ALOG10(PSNO)
        ENDIF
C
9       PRINT 603
        PRINT*,'      ENTER FALSE ALARM RATE'
        PRINT*,'      (false alarms per second)     '
        READ*,  FAR
        PRINT 603
        CALL QREPLY('     DO YOU WISH TO CORRECT DATA ?            ',CK)
        IF (CK) GOTO 4
C
C---***---***---***---***---***---***---***---***---***
C
C   COMMONLY USED VARIABLES
C
        BNLP = BN(BLP,NPPOSF)
        BNPRE = BN(BPF,NPPREF)
        PI = 3.1415926
        G = 1.E12
        NFIG = 1.0
        NO = 4.E-21 * NFIG * G
        PS = NO * PSNO
        APEAK = SQRT(2.0 * PS)
        IF((CHOICE.EQ.2).OR.(CHOICE.EQ.3)) GOTO 6
C
C----------------------------------------------------
C
170     QUEST1 = 'DO YOU WISH TO SAVE THE INPUT SIGNAL ?      '
        CALL QREPLY(QUEST1,SAVEI)
        IF(SAVEI) THEN
C
           CALL RNAME('enter file name  ',NAME1)
        ENDIF
C
        PRINT 603
        QUEST2 = 'DO YOU WISH TO SAVE THE OUTPUT SIGNAL ?     '
        CALL QREPLY(QUEST2,SAVEO)
        IF (SAVEO) THEN
C
           CALL RNAME('enter file name  ',NAME2)
        ENDIF
C
        PRINT 603
        QUEST3 = 'DO YOU WISH TO SAVE THE VARIANCE ?          '
        CALL QREPLY(QUEST3,SAVEV)
```

```
      IF (SAVEV) THEN
         CALL RNAME('enter file name  ',NAME3)
      ENDIF
      PRINT 603
      CALL QREPLY('     DO YOU WISH TO CORRECT DATA ?          ',CK)
      PRINT 603
      IF (CK) GOTO 170
C
C#################################################################
C
C      COMPUTE INPUT ARRAY ,s(n)
C
C#################################################################
C
      TPS = SMPTME * (NPOINT/2) -(0.5 * (TAU + RISTME))
      DO 10 I=0,NPOINT-1
         TIME = I * SMPTME
         IF(TIME.GE.TPS) THEN
            TIME1 = TIME - TPS
            TEMP1 = TRAP(RISTME,TAU,TIME1)
C           TEMP1 = TRAP(RISTME,TAU,TIME)
            ARG = 2.0 * PI * OFFSET * TIME
            CTEMP = CMPLX(COS(ARG),SIN(ARG))
            CTEMP1 = CMPLX(TEMP1,0.0) * CTEMP
         ELSE
            CTEMP1 = CMPLX(0.0,0.0)
         ENDIF
         CXARRY(I) = CTEMP1
         FILE(I) = REAL(CTEMP1)
10    CONTINUE
C
      IF (SAVEI) THEN
         FILE(NPOINT) = F0
         FILE(NPOINT + 1) = FLOAT(NPPREF)
         FILE(NPOINT + 2) = RBPF
         FILE(NPOINT + 3) = SMPTME
         FILE(NPOINT + 4) = FLOAT(NPPOSF)
         BILE(NPOINT + 5) = BLP
      FILE(NPOINT + 6) = TAU
      FILE(NPOINT + 7) = RISTME
      FILE(NPOINT + 8) = OFFSET
         I = NPOINT + 9
         CALL WDATA(I,FILE(0),NAME1)
      ENDIF
C
C#################################################################
C
C      COMPUTE SIGNAL PORTION OF OUTPUT , Ys(n)
C
C#################################################################
C
      CALL FFT(CXARRY,NPOINT,0)
C
      DO 20 I=0,NPOINT/2 - 1
         FREQ = F0 * I
         CTEMP = BUTTER(FREQ,NPPREF,BPF)
         P(I) = CXARRY(I) * CTEMP
         NFREQ = -1.0 * (I+1)  * F0
         CTEMP = BUTTER(NFREQ,NPPREF,BPF)
         P(NPOINT- 1 - I) = CXARRY(NPOINT-1-I) * CTEMP
20    CONTINUE
      P(NPOINT/2) = CMPLX(0.0,0.0)
C
      CALL FFT(P,NPOINT,1)
C
      DO 30 I=0,NPOINT-1
```

```
            CXARRY(I) = P(I) * CONJG(P(I))
30        CONTINUE
C
          CALL FFT(CXARRY,NPOINT,0)
C
          DO 40 I=0,NPOINT/2 -1
             FREQ = F0 * I
             LITG(I) = BUTTER(FREQ,NPPOSF,BLP)
             Y(I) = LITG(I) * CXARRY(I)
             NFREQ = -1. * (I+1) * F0
             LITG(NPOINT-1-I) = BUTTER(NFREQ,NPPOSF,BLP)
             Y(NPOINT-1-I) = CXARRY(NPOINT-1-I) * LITG(NPOINT-1-I)
40        CONTINUE
          Y(NPOINT/2) = CMPLX(0.0,0.0)
          FREQ = (NPOINT/2) * F0
          LITG(NPOINT/2) = BUTTER(FREQ,NPPOSF,BLP)
C
             CALL FFT(Y,NPOINT,1)
C
          IF (SAVEO) THEN
             DO 50 I=0,NPOINT-1
                FILE(I) = REAL(Y(I))
50           CONTINUE
             CALL WDATA(NPOINT,FILE(0),NAME2)
          ENDIF
C
C###############################################################
C
C         COMPUTE TIME VARYING PART OF OUTPUT NOISE
C         VARIANCE , Ry1(n)
C
C###############################################################
C
C    Determine PSD of noise after pre-filter
          DO 113 I=1,NPOINT/2
             FREQ = I * F0
             CTEMP = BUTTER (FREQ,NPPREF,BPF)
             CTEMP1 = CONJG(CTEMP)
             SWIG(I) = CMPLX(2.0 * N0 * CTEMP * CTEMP1)
             SWIG(NPOINT - I) = SWIG(I)
113       CONTINUE
          FREQ = 0.0
          CTEMP = BUTTER (FREQ,NPPREF,BPF)
          CTEMP1 = CONJG(CTEMP)
          SWIG(0) = CMPLX(2.0 * N0 * CTEMP * CTEMP1)
C
C    Determine impulse responce of post-filter
          CALL FFT(LITG(0),NPOINT,1)
C
          RUNTME = 35
          IF (NPOINT.EQ.128) RUNTME = 3
          IF (NPOINT.EQ.256) RUNTME = 7
          PRINT*
          PRINT*,'       PLEASE WAIT ',RUNTME,' minutes'
          PRINT*,'       ( Please make sure printer is on )'
C
C    Perform intergation
          DELTAF = F0 / 3.0
          DO 120 K=0,NPOINT-1
             RY1(K) =  CMPLX(0.0,0.0)
             DO 130 I=0,NPOINT-1
                INDX = K - I
               IF (INDX.LT.0) INDX = INDX + NPOINT
                CXARRY(I) = P(INDX) * LITG(I)
130          CONTINUE
C
```

```
            CALL FFT(CXARRY(0),NPOINT,0)
C
            RY1W = CMPLX(0.0,0.0)
            RY1H = CMPLX(0.0,0.0)
            DO 140 II=0,NPOINT-2,2
              CTEMP1 = CONJG(CXARRY(II))
               DCMPX = CMPLX(CXARRY(II) * CTEMP1)
               TERM = SWIG(II) * DCMPX
               IF(II.EQ.NPOINT/2) TERM = 0.5 * TERM
               CTEMP1 = CONJG(CXARRY(II+1))
               DCMPX = CMPLX(CXARRY(II+1) * CTEMP1)
               TERM1 = SWIG(II+1) * DCMPX
               RY1W = RY1W + TERM
               RY1H = RY1H + TERM1
140         CONTINUE
            RY1(K) = DELTAF * (2.0 * RY1W + 4.0 * RY1H)
120      CONTINUE
C
C############################################################
C
C
C     COMPUTE NON-TIME VARYING PART OF OUTPUT NOISE
C     VARIANCE, Ry3
C
C############################################################
C
         RY3 = 16.0 * (N0 ** 2) * BNLP * BNPRE
C
         DO 300 I=0,NPOINT-1
            VARTV = REAL(RY1(I))
            FILE(I) = VARTV + VARTV + RY3
300      CONTINUE
C
         IF(SAVEV)THEN
C
            CALL WDATA(NPOINT,FILE(0),NAME3)
         ENDIF
C############################################################
C
C    COMPUTE THRESHOLD VOLTAGE , Vt
C
C############################################################
C
6        PF = FAR/(2.0 * BNLP)
         SD = SQRT(RY3)
         YNMEAN = 4.0 * N0 * BNPRE
         ALPHAPF = QINVER(PF)
         VT = (ALPHAPF * SD) + YNMEAN
C
C   Determine independent sample distance
         INPDIS = NINT(1.0/(2.0 * BNLP * SMPTME))
         IF (INPDIS.EQ.0) INPDIS = 1
C
C    Determine maximum value of output signal
         YMAX = 0.0
         YMAXNO = -1
         DO 400 I=0,NPOINT-1
            IF(REAL(Y(I)).GT.YMAX) THEN
               YMAX = REAL(Y(I))
               YMAXNO = I
            ENDIF
400      CONTINUE
C
C###########################################################
C
C    COMPUTE P[Ys(j) < Vt]  for each independent
```

```
C   sample.   Also compute probability of detection ,Pd
C
C###########################################################
C
        TOTSIPTS = NPOINT/INPDIS
        ARB = 0.10
        ARBLVL = ARB * YMAX
C
        PLTVT = 1.0
        DO 500 I=0,TOTSIPTS
           NOFP = YMAXNO + (I * INPDIS)
           IF(NOFP.LE.NPOINT-1) THEN
              IF(REAL(Y(NOFP)).GE.ARBLVL) THEN
                 YS = REAL(Y(NOFP)) * (APEAK ** 2)
                 SIG = SQRT(((FILE(NOFP) -RY3)*(APEAK**2))+RY3)
                 ALPHA = DBLE((VT - (YS + YNMEAN))/SIG)
                 PJ = 1.0 - QFUNCT(ALPHA)
                 PLTVT = PLTVT * PJ
              ENDIF
           ENDIF
C
           NOFP = YMAXNO - ((I+1) * INPDIS)
           IF(NOFP.GE.0) THEN
              IF(REAL(Y(NOFP)).GE.ARBLVL) THEN
                 YS = REAL(Y(NOFP)) * (APEAK ** 2)
                 SIG = SQRT(((FILE(NOFP) -RY3) * (APEAK**2)) +RY3)
                 ALPHA = DBLE((VT - (YS + YNMEAN))/SIG)
                 PJ = 1.0 - QFUNCT(ALPHA)
                 PLTVT = PLTVT * PJ
              ENDIF
           ENDIF
500     CONTINUE
C
        PROBD = 1.0 - PLTVT
        PDPC = 100.0 * PROBD
        PRINT*
        PRINT*,'      Pd = ',PDPC,' %'
        PRINT*
C
C#############################################################
C
C    DOCUMENT RUN
C
C#############################################################
C
        OPEN(UNIT=4,FILE='LST:',FORM='FORMATTED')
        WRITE(4,763)
763     FORMAT(' '/////////' ')
        WRITE(4,761)'SQUARE LAW DETECTOR -> PROBABILITY OF DETECTION'
761     FORMAT(' ',15X,A47)
        WRITE(4,762)'GAUSSIAN ASSUMPTION'
762     FORMAT(' ',29X,A19)
        WRITE(4,*)' '
        WRITE(4,*)'                         BUTTERWORTH PRE-FILTER '
710     FORMAT(' ',15X,A30,G12.6,A5)
        WRITE(4,710)'      RF BANDWIDTH              : ',RBPF1,' MHz '
720     FORMAT(' ',15X,A30,I1)
        WRITE(4,720)'      NUMBER OF POLES           : ',NPPREF
        WRITE(4,*)' '
        WRITE(4,*)'                         BUTTERWORTH POST-FILTER '
        WRITE(4,710)'         BANDWIDTH              : ',BLP2,' MHz '
        WRITE(4,720)'      NUMBER OF POLES           : ',NPPOSF
        WRITE(4,*)' '
        WRITE(4,*)'                         INPUT PULSE '
        WRITE(4,710)'      50% AMPLITUDE WIDTH       : ',TAU,' s    '
        WRITE(4,710)'      RISE TIME                 : ',RISTME,' s    '
```

```
      WRITE(4,710)'     FREQUENCY OFFSET       : ',OFFSET,' Hz  '
      WRITE(4,*)' '
      WRITE(4,*)'                        MISCELLANEOUS '
      WRITE(4,710)'     FUNDAMENTAL FREQUENCY  : ',F0,' Hz  '
730   FORMAT(' ',15X,A30,I5)
      WRITE(4,730)'     NUMBER OF SAMPLES      : ',NPOINT
      WRITE(4,710)'     SIGNAL / NOISE DENSITY : ',PSNODB,' dB  '
      WRITE(4,710)'     FALSE ALARM RATE       : ',FAR,' FA/s'
      WRITE(4,*)' '
740   FORMAT(' ',15X,A36,A23)
      IF (SAVEI) THEN
      WRITE(4,740)'     INPUT SIGNAL STORED IN FILE  : ',NAME1
      ENDIF
      IF (SAVEO) THEN
      WRITE(4,740)'     OUTPUT SIGNAL STORED IN FILE : ',NAME2
      ENDIF
      IF (SAVEV) THEN
      WRITE(4,740)'     NOISE VARIANCE STORED IN FILE : ',NAME3
      ENDIF
      WRITE(4,*)' '
      WRITE(4,*)'                        ********** RESULT **********'
      WRITE(4,*)' '
750   FORMAT(' ',15X,A32,F6.2,A2)
      WRITE(4,750)'     PROBABILITY OF DETECTION   : ',PDPC,' %'
760   FORMAT('1')
      WRITE(4,760)
      CLOSE(4)
C
C#############################################################
C
C   END OF MAIN PROGRAM
C
C#############################################################
C
      MARK = 1
      QUEST4 = 'DO YOU WISH TO RUN THE PROGRAM AGAIN ?     '
      CALL QREPLY(QUEST4,AGAIN)
      IF (AGAIN) GOTO 1
      STOP
      END
C
C*******************************************************************
C*******************************************************************
C*******************************************************************
C
C       DEFINE TEAPIZIODAL PULSE
C
      REAL FUNCTION TRAP(RISTME,TAU,TIME)
      REAL RISTME,TAU,TIME
      IF (TIME.GE.RISTME) GOTO 11
      TRAP = TIME/RISTME
      GOTO 41
11    IF (TIME.GT.TAU) GOTO 21
      TRAP = 1.0
      GOTO 41
21    IF (TIME.GT. (RISTME+TAU)) GOTO 31
      TRAP = (TAU + RISTME - TIME)/RISTME
      GOTO 41
31    TRAP = 0.0
41    RETURN
      END
C
C*******************************************************************
C
C       DISCRETE FOURIER TRANSFORM OPERATION
C
```

```
        SUBROUTINE FFT(X,N,INV)
        COMPLEX X(512),W,T
        ITER = 0
        IREM = N
12      IREM = IREM/2
        IF (IREM.EQ.0) GOTO 22
        ITER = ITER + 1
        GOTO 12
22      CONTINUE
        S = -1
        IF (INV.EQ.1) S=1
        NXP2 = N
        DO 52 IT=1,ITER
        NXP = NXP2
        NXP2 = NXP/2
        WPWR = 3.141592/FLOAT(NXP2)
        DO 42 M=1,NXP2
        ARG = FLOAT(M-1) * WPWR
        W = CMPLX(COS(ARG),S*SIN(ARG))
        DO 42 MXP = NXP,N,NXP
        J1 = MXP - NXP + M
        J2 = J1 + NXP2
        T = X(J1) - X(J2)
        X(J1) = X(J1) + X(J2)
42      X(J2) = T * W
52      CONTINUE
        N2 = N/2
        N1 = N-1
        J = 1
        DO 65 I=1,N1
        IF (I.GE.J) GOTO 55
        T = X(J)
        X(J) = X(I)
        X(I) = T
55      K = N2
60      IF(K.GE.J) GOTO 65
        J = J - K
        K = K/2
        GOTO 60
65      J = J + K
        IF (INV.EQ.1) GOTO 75
        DO 70 I=1,N
70      X(I) = X(I)/FLOAT(N)
75      CONTINUE
        RETURN
        END
C
C*********************************************************************
C
C       EVALUATE BUTTERWORTH FILTER RESPONCE
C
        COMPLEX FUNCTION BUTTER(F,N,B)
        COMPLEX S,D1,D2
        REAL F,B,FB,S2
        INTEGER N
        FB = F/B
        S = CMPLX(0.0,FB)
        S2 = ((-1.0) * FB * FB)
        GOTO (13,23,33,43), N
13      BUTTER = CMPLX(1.0 / (S + 1.0))
        RETURN
23      BUTTER = CMPLX(1.0/(S2 + 1.414 * S + 1.0))
        RETURN
33      BUTTER = CMPLX(1.0/((S + 1.0) * (S2 + S +1.0)))
        RETURN
43      D1 = CMPLX(S2 + 0.765 * S + 1.0)
```

```
        D2 = CMPLX(S2 + 1.848 * S + 1.0)
        BUTTER = CMPLX(1.0/(D1 * D2))
        RETURN
        END
C
C*******************************************************************
C
C       READ FILENAME
C
        SUBROUTINE RNAME(ISTRING,NAME)
        CHARACTER ISTRING*17, NAME*17
        PRINT5, ISTRING
5       FORMAT (8X,A40)
        READ14, NAME
14      FORMAT(A17)
        PRINT*,' '
        RETURN
        END
C
C*****************************************************************
C
C       READ DATA FILE
C
        SUBROUTINE RDATA (LENGTH, VAR, NAME)
        CHARACTER NAME*17, A*1
        REAL VAR(520)
        LENGTH = 0
        OPEN (UNIT = 10, FILE = NAME)
        DO 101 LOOP = 1, 520
           READ(UNIT=10, FMT=111, END=121) VAR(LOOP)
           LENGTH = LENGTH + 1
101     CONTINUE
111     FORMAT(E16.8)
121     CLOSE(UNIT = 10)
        RETURN
        END
C
C******************************************************************
C
C       WRITE DATA FILE TO DISK
C
        SUBROUTINE WDATA(LENGTH, VAR, NAME)
        CHARACTER NAME*17, A*1
        REAL VAR(520)
        OPEN (UNIT=10,FILE = NAME,STATUS = 'UNKNOWN')
        DO 15 LOOP = 1,LENGTH
           WRITE(10, 25) VAR(LOOP)
15     CONTINUE
25      FORMAT(E16.8)
        CLOSE(UNIT=10)
        RETURN
        END
C
C******************************************************************
C
C       EVALUATE USER REPLY
C
        SUBROUTINE QREPLY(HEADER,RESULT)
        CHARACTER HEADER*43, TEST*1
        LOGICAL RESULT
        RESULT = .TRUE.
        PRINT 16,HEADER
16      FORMAT (5X,A43)
        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
C       CLEAR CRT
C
        SUBROUTINE CLEARCRT(N)
        INTEGER N
        DO 17 I=1,N
          PRINT*,' '
17      CONTINUE
        RETURN
        END
C
C*****************************************************************
        REAL FUNCTION QINVER (RQ)
C                                                               *
C       SUBROUTINE  QINVER                                      *
C                                                               *
C       PURPOSE:                                                *
C               TO COMPUTE THE VALUE OF X GIVEN THE Q-FUNCTION *
C               VALUE.                                          *
C                                                               *
C       ARGUMENTS SUPPLIED BY CALLING ROUTINE:                  *
C               RQ - THE VALUE OF THE Q-FUNCTION                *
C                                                               *
C       ARGUMENTS RETURNED TO CALLING ROUTINE:                  *
C               X - VALUE OF X CORRESPONDING TO Q(x)            *
C                                                               *
C*****************************************************************
C
C   INTIALIZE VARIABLES
C
        REAL X,RQ,Q,A,B,UBX,ESTQ,ERROR,SA,BOUND,DIF,MULT,DIR
        REAL LSTDIR,HOMING,PI,F,TQ
C
C**********************************************************************
C
C DETERMINE IF X IS POSITIVE OR NEGATIVE AND SET ROUTINE FOR POS. CASE
C
        IF (RQ.GT.0.5) THEN
                Q = 1.0 - RQ
                MULT = -1.0
        ELSE
                Q = RQ
                MULT = 1.0
        ENDIF
C
C**********************************************************************
C
C   INTINALIZE AND DEFINE VARIABLES FOR ESTIMATING X
C
        A = 0.339
        B = 5.510
        PI = 3.14159265
        SA = 1.0 - A
        F = 1.0 / (SQRT (2.0 * PI))
        LSTDIR = -1.0
        HOMING = 0.1
C
C**********************************************************************
C
C   COMPUTE ROUGH UPPER BOUND ESTIMATE OF X
C
```

```
      IF(Q.NE.0.0) THEN
         UBX = SQRT (ALOG (1.0 / (4.0 * Q**2)))
      ELSE
         UBX = 21.16518611
      ENDIF
C
C*********************************************************************
C   % ERROR ALLOWED IN Q(x) BETWEEN ACTUAL AND ESTIMATE OF X, Q(x)
      BOUND = 0.0001
C*********************************************************************
C
C   CALCULATING AND HOMING IN ON X
C
      TQ = Q
C
18    ESTQ=(1.0/(SA*UBX+A*SQRT(UBX**2+B)))*F*EXP(-1.0*(UBX**2)/2.0)
      DIF = Q - ESTQ
      IF(Q.EQ.0.0) THEN
         TQ = ESTQ
         IF(ESTQ.EQ.0.0) TQ = 1.0
      ENDIF
      ERROR = (ABS (DIF)) / TQ
      IF (ERROR.GT.BOUND) THEN
              IF (DIF.GE.0.0) THEN
                      DIR = -1.0
              ELSE
                      DIR = 1.0
              ENDIF
              IF(DIR.NE.LSTDIR) HOMING=HOMING*0.1
              LSTDIR = DIR
              UBX = UBX + (DIR * HOMING)
              GOTO 18
      ELSE
              QINVER = UBX * MULT
      ENDIF
C
C********************************************************************
C
C   RETURN TO CALLING ROUTINE
C
      RETURN
      END
C
C
C******************************************************************
C                                                                 *
      REAL*8 FUNCTION QFUNCT(RX)
C                                                                 *
C     FUNCTION: QFUNCT                                            *
C                                                                 *
C     PURPOSE:                                                    *
C     TO COMPUTE THE AN APPROXIMATION OF THE Q FUNCTION GIVEN     *
C     AN INPUT, X                                                 *
C                                                                 *
C******************************************************************
C
C   INTIALIZE VALUES USED IN CALCULATION
C
      REAL*8 A,B,PI,SA,F,X,XE,RX
      IF (RX.LT.0.0) THEN
              X = DABS(RX)
      ELSE
              X = RX
      ENDIF
      A = 0.339
      B = 5.510
      PI = 3.14159265
      SA = 1.0 - A
      F = 1.0 / (DSQRT(2.0 * PI))
C
C*******************************************************************
C
C   CALCULATE Q FUNCTION FOR X
C
      XE = DEXP(-1.0 * (X**2) / 2.0)
      QFUNCT = (1.0 / (SA * X + A * DSQRT(X**2 + B))) * F * XE
      IF(RX.LT.0.0) QFUNCT = 1.0 - QFUNCT
      RETURN
      END
C
C************************************************************
C
C   CALCULATES NOISE BANDWIDTH FOR BUTTERWORTH FILTERS
C
      REAL FUNCTION BN(B3,NPOLES)
      REAL B3,BN
      INTEGER NPOLES
      PI = 3.14159265
      GOTO (19,29,39,49) NPOLES
19       BN = (PI/2.0) * B3
         RETURN
29       BN = (PI/2.828) * B3
         RETURN
39       BN = (PI/3.0) * B3
         RETURN
49       BN = (PI/3.064) * B3
         RETURN
      END
```

## REFERENCES

[1] Simon Haykin, Communication Systems, John Wiley & Sons, Inc., New York, 1983.

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

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

[4] D. R. Hummels, C. Adams, and B. K. Harms, Filter Selection for Receivers Using Square-Law Detection, IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-19, 871-883, Novemeber 1983.

[5] B. K. Harms and D. R. Hummels, An Analysis and Comparison of the Channelized, Acoustoopic, and Frequency Compressive Intercept Receivers, Engineering Experiment Station, Project 2851, Manhattan, Kansas ,1985.

# A NUMERICAL PROCEDURE FOR COMPUTING PROBABILITY OF DETECTION FOR A WIDEBAND PULSE RECEIVER

by

SCOTT D. BRILES

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

1986

## ABSTRACT

This report describes the development of a computer program which can be used to determine probability of detection for a square-law pulse receiver. The report is intended to be used as a manual for using the program. The numerical procedures implemented in the program are based on the condition of a large pre-filter to post-filter bandwidth ratio which allows for the assumption of a Gaussian noise process at the output of the receiver. The advantage of the Gaussian assumption is a decrease in the difficulty of obtaining solutions for the probability of detection.