2.1. Detection in the Presence of Noise

A simplified block diagram of a radar receiver that employs an envelope detector followed by a threshold decision is shown in Fig. 2.1. The input signal to the receiver is composed of the radar echo signal $s(t)$ and additive zero mean white Gaussian noise $n(t)$, with variance $\psi^2$. The input noise is assumed to be spatially incoherent and uncorrelated with the signal.

The output of the bandpass IF filter is the signal $v(t)$, which can be written as

\begin{align*}
 v(t) &= v_I(t)\cos \omega_0 t + v_Q(t)\sin \omega_0 t = r(t)\cos(\omega_0 t - \phi(t)) \\
 v_I(t) &= r(t)\cos \phi(t) \\
 v_Q(t) &= r(t)\sin \phi(t)
\end{align*}

where $\omega_0 = 2\pi f_0$ is the radar operating frequency, $r(t)$ is the envelope of $v(t)$, the phase is $\phi(t) = \text{atan}(v_Q/v_I)$, and the subscripts $I, Q$, respectively, refer to the in-phase and quadrature components.

A target is detected when $r(t)$ exceeds the threshold value $V_T$, where the decision hypotheses are

\begin{align*}
 s(t) + n(t) &> V_T & \text{Detection} \\
 n(t) &> V_T & \text{False alarm}
\end{align*}
The case when the noise subtracts from the signal (while a target is present) to make $r(t)$ smaller than the threshold is called a miss. Radar designers seek to maximize the probability of detection for a given probability of false alarm.

The IF filter output is a complex random variable that is composed of either noise alone or noise plus target return signal (sine wave of amplitude $A$). The quadrature components corresponding to the first case are

$$ v_I(t) = n_I(t) $$
$$ v_Q(t) = n_Q(t) $$

and for the second case,

$$ v_I(t) = A + n_I(t) = r(t) \cos \varphi(t) \Rightarrow n_I(t) = r(t) \cos \varphi(t) - A $$
$$ v_Q(t) = n_Q(t) = r(t) \sin \varphi(t) $$

where the noise quadrature components $n_I(t)$ and $n_Q(t)$ are uncorrelated zero mean low pass Gaussian noise with equal variances, $\psi^2$. The joint Probability Density Function (pdf) of the two random variables $n_I, n_Q$ is

$$ f(n_I, n_Q) = \frac{1}{2\pi\psi^2} \exp\left( \frac{-n_I^2 + n_Q^2}{2\psi^2} \right) $$

$$ = \frac{1}{2\pi\psi^2} \exp\left( \frac{-(r \cos \varphi - A)^2 + (r \sin \varphi)^2}{2\psi^2} \right) $$

The pdfs of the random variables $r(t)$ and $\varphi(t)$, respectively, represent the modulus and phase of $v(t)$. The joint pdf for the two random variables $r(t); \varphi(t)$ is given by

$$ f(r, \varphi) = f(n_I, n_Q)|J| $$

where $|J|$ is a matrix of derivatives defined by

\[\text{Figure 2.1. Simplified block diagram of an envelope detector and threshold receiver.}\]
The determinant of the matrix of derivatives is called the Jacobian, and in this case it is equal to

$$|J| = r(t)$$

Substituting Eqs. (2.4) and (2.7) into Eq. (2.5) and collecting terms yield

$$f(r, \varphi) = \frac{r}{2\pi \psi^2} \exp\left(\frac{r^2 + A^2}{2\psi^2}\right) \exp\left(\frac{rA \cos \varphi}{\psi^2}\right)$$

The pdf for \( r \) alone is obtained by integrating Eq. (2.8) over \( \varphi \)

$$f(r) = \int_0^{2\pi} f(r, \varphi) d\varphi = \frac{r}{\psi^2} \exp\left(-\frac{r^2 + A^2}{2\psi^2}\right) \frac{1}{2\pi} \int_0^{2\pi} \exp\left(\frac{rA \cos \varphi}{\psi^2}\right) d\varphi$$

where the integral inside Eq. (2.9) is known as the modified Bessel function of zero order,

$$I_0(\beta) = \frac{1}{2\pi} \int_0^{2\pi} e^{\beta \cos \theta} d\theta$$

Thus,

$$f(r) = \frac{r}{\psi^2} I_0\left(\frac{rA}{\psi^2}\right) \exp\left(-\frac{r^2 + A^2}{2\psi^2}\right)$$

which is the Rician probability density function. If \( A/\psi^2 = 0 \) (noise alone), then Eq. (2.11) becomes the Rayleigh probability density function

$$f(r) = \frac{r}{\psi^2} \exp\left(-\frac{r^2}{2\psi^2}\right)$$

Also, when \( (A/\psi^2) \) is very large, Eq. (2.11) becomes a Gaussian probability density function of mean \( A \) and variance \( \psi^2 \):
Fig. 2.2 shows plots for the Rayleigh and Gaussian densities. For this purpose, use MATLAB program “fig2_2.m” given in Listing 2.1 in Section 2.11. This program uses MATLAB functions “normpdf.m” and “raylpdf.m”. Both functions are part of the MATLAB Statistics toolbox. Their associated syntax is as follows

\[
\text{normpdf}(x, \mu, \sigma) \\
\text{raylpdf}(x, \sigma) 
\]

“\(x\)” is the variable, “\(\mu\)” is the mean, and “\(\sigma\)” is the standard deviation.

The density function for the random variable \(\varphi\) is obtained from

\[
f(\varphi) = \int_0^r f(r, \varphi) \, dr 
\]  \hspace{1cm} (2.14)

While the detailed derivation is left as an exercise, the result of Eq. (2.14) is

\[
f(\varphi) = \frac{1}{2\pi} \exp\left(-\frac{A^2}{2\psi^2}\right) + \frac{A \cos \varphi}{\sqrt{2\pi}\psi^2} \exp\left(-\frac{(A \sin \varphi)^2}{2\psi^2}\right) F\left(\frac{A \cos \varphi}{\psi}\right) 
\]  \hspace{1cm} (2.15)
where

\[ F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\xi^2/2} d\xi \]  \hspace{1cm} (2.16)

The function \( F(x) \) can be found tabulated in most mathematical formula reference books. Note that for the case of noise alone \( A = 0 \), Eq. (2.15) collapses to a uniform pdf over the interval \( \{0, 2\pi\} \).

One excellent approximation for the function \( F(x) \) is

\[ F(x) = 1 - \left( \frac{1}{0.661x + 0.339 \sqrt{x^2 + 5.51}} \right) \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \hspace{1cm} x \geq 0 \]  \hspace{1cm} (2.17)

and for negative values of \( x \)

\[ F(-x) = 1 - F(x) \]  \hspace{1cm} (2.18)

**MATLAB Function “que_func.m”**

The function “que_func.m” computes \( F(x) \) using Eqs. (2.17) and (2.18) and is given in Listing 2.2 in Section 2.11. The syntax is as follows:

\[ fofx = \text{que_func}(x) \]

### 2.2. Probability of False Alarm

The probability of false alarm \( P_{fa} \) is defined as the probability that a sample \( R \) of the signal \( r(t) \) will exceed the threshold voltage \( V_T \) when noise alone is present in the radar,

\[ P_{fa} = \int_{V_T}^{\infty} \frac{R}{\Psi^2} \exp \left( -\frac{R^2}{2\Psi^2} \right) dr = \exp \left( -\frac{V_T^2}{2\Psi^2} \right) \]  \hspace{1cm} (2.19a)

\[ V_T = \sqrt{2\Psi^2 \ln \left( \frac{1}{P_{fa}} \right)} \]  \hspace{1cm} (2.19b)

Fig. 2.3 shows a plot of the normalized threshold versus the probability of false alarm. It is evident from this figure that \( P_{fa} \) is very sensitive to small changes in the threshold value. This figure can be reproduced using MATLAB program “fig2_3.m” given in Listing 2.3 in Section 2.11.

The false alarm time \( T_{fa} \) is related to the probability of false alarm by
where $t_{\text{int}}$ represents the radar integration time, or the average time that the output of the envelope detector will pass the threshold voltage. Since the radar operating bandwidth $B$ is the inverse of $t_{\text{int}}$, then by substituting Eq. (2.19) into Eq. (2.20) we can write $T_{fa}$ as

$$T_{fa} = t_{\text{int}} / P_{fa} \quad (2.20)$$

Minimizing $T_{fa}$ means increasing the threshold value, and as a result the radar maximum detection range is decreased. Therefore, the choice of an acceptable value for $T_{fa}$ becomes a compromise depending on the radar mode of operation.

Fehlner\(^1\) defines the false alarm number as

$$n_{fa} = \frac{-\ln(2)}{\ln(1 - P_{fa})} \approx \frac{\ln(2)}{P_{fa}} \quad (2.22)$$

---


© 2004 by Chapman & Hall/CRC CRC Press LLC
Other slightly different definitions for the false alarm number exist in the literature, causing a source of confusion for many non-expert readers. Other than the definition in Eq. (2.22), the most commonly used definition for the false alarm number is the one introduced by Marcum (1960). Marcum defines the false alarm number as the reciprocal of $P_{fa}$. In this text, the definition given in Eq. (2.22) is always assumed. Hence, a clear distinction is made between Marcum’s definition of the false alarm number and the definition in Eq. (2.22).

### 2.3. Probability of Detection

The probability of detection $P_D$ is the probability that a sample $R$ of $r(t)$ will exceed the threshold voltage in the case of noise plus signal,

$$P_D = \int_{V_T}^{\infty} \frac{r}{\psi^2} I_0\left(\frac{rA}{2\psi}\right) \exp\left(-\frac{r^2 + A^2}{2\psi^2}\right) dr$$  

(2.23)

If we assume that the radar signal is a sine waveform with amplitude $A$, then its power is $A^2/2$. Now, by using $SNR = A^2/2\psi^2$ (single-pulse SNR) and $V_T^2/2\psi^2 = \ln(1/P_{fa})$, then Eq. (2.23) can be rewritten as

$$P_D = \int_{\sqrt{2\psi^2 \ln(1/P_{fa})}}^{\infty} \frac{r}{\psi^2} I_0\left(\frac{rA}{2\psi}\right) \exp\left(-\frac{r^2 + A^2}{2\psi^2}\right) dr =$$

(2.24)

$$Q\left[A^2/\psi^2, \sqrt{2\ln\left(1/P_{fa}\right)}\right]$$

$$Q[\alpha, \beta] = \int_{\beta}^{\infty} \zeta I_0(\alpha\zeta) e^{-(\zeta^2 + \alpha^2)/2} \, d\zeta$$  

(2.25)

$Q$ is called Marcum’s Q-function. When $P_{fa}$ is small and $P_D$ is relatively large so that the threshold is also large, Eq. (2.24) can be approximated by

$$P_D \approx F\left(\frac{A}{\psi} - \sqrt{2\ln\left(1/P_{fa}\right)}\right)$$  

(2.26)

where $F(x)$ is given by Eq. (2.16). Many approximations for computing Eq. (2.24) can be found throughout the literature. One very accurate approximation presented by North (see bibliography) is given by

$$P_D \approx 0.5 \times \text{erfc}(\sqrt{-\ln P_{fa} - \sqrt{SNR + 0.5}})$$  

(2.27)

where the complementary error function is
MATLAB Function “marcumsq.m”

The integral given in Eq. (2.24) is complicated and can be computed using numerical integration techniques. Parl¹ developed an excellent algorithm to numerically compute this integral. It is summarized as follows:

\[ Q[a, b] = \begin{cases} \frac{\alpha_n}{2\beta_n} \exp\left(\frac{(a-b)^2}{2}\right) & a < b \\ 1 - \left(\frac{\alpha_n}{2\beta_n} \exp\left(\frac{(a-b)^2}{2}\right)\right) & a \geq b \end{cases} \]  

(2.29)

\[ \alpha_n = d_n + \frac{2n}{ab} \alpha_{n-1} + \alpha_{n-2} \]  

(2.30)

\[ \beta_n = 1 + \frac{2n}{ab} \beta_{n-1} + \beta_{n-2} \]  

(2.31)

\[ d_{n+1} = d_n d_1 \]  

(2.32)

\[ \alpha_0 = \begin{cases} 1 & a < b \\ 0 & a \geq b \end{cases} \]  

(2.33)

\[ d_1 = \begin{cases} a/b & a < b \\ b/a & a \geq b \end{cases} \]  

(2.34)

\[ \alpha_{-1} = 0.0, \beta_0 = 0.5, \text{ and } \beta_{-1} = 0. \] The recursive Eqs. (2.30) through (2.32) are computed continuously until \( \beta_n > 10^p \) for values of \( p \geq 3 \). The accuracy of the algorithm is enhanced as the value of \( p \) is increased. The MATLAB function “marcumsq.m” given in Listing 2.4 in Section 2.11 implements Parl’s algorithm to calculate the probability of detection defined in Eq. (2.24). The syntax is as follows:

\[ Pd = \text{marcumsq}(alpha, beta) \]

where \( alpha \) and \( beta \) are from Eq. (2.25). Fig. 2.4 shows plots of the probability of detection, \( P_D \), versus the single pulse SNR, with the \( P_{fa} \) as a parameter. This figure can be reproduced using the MATLAB program “prob_snrl.m” given in Listing 2.5 in Section 2.11.

---

2.4. Pulse Integration

Pulse integration was discussed in Chapter 1 in the context of radar measurements. In this section a more comprehensive analysis of this topic is introduced in the context of radar detection. The overall principles and conclusions presented earlier will not change; however, the mathematical formulation and specific numerical values will change. Coherent integration preserves the phase relationship between the received pulses, thus achieving a build up in the signal amplitude. Alternatively, pulse integration performed after the envelope detector (where the phase relation is destroyed) is called non-coherent or post-detection integration.

2.4.1. Coherent Integration

In coherent integration, if a perfect integrator is used (100% efficiency), then integrating \( n_p \) pulses would improve the SNR by the same factor. Otherwise, integration loss occurs which is always the case for non-coherent integration. In order to demonstrate this signal buildup, consider the case where the radar return signal contains both signal plus additive noise. The \( m^{th} \) pulse is
where $s(t)$ is the radar return of interest and $n_m(t)$ is white uncorrelated additive noise signal. Coherent integration of $n_P$ pulses yields

$$z(t) = \frac{1}{n_P} \sum_{m=1}^{n_P} y_m(t) = \frac{1}{n_P} [s(t) + n_m(t)] = s(t) + \sum_{m=1}^{n_P} \frac{1}{n_P} n_m(t)$$

The total noise power in $z(t)$ is equal to the variance. More precisely,

$$\psi_{nz}^2 = E \left[ \sum_{m=1}^{n_P} \frac{1}{n_P} n_m(t) \sum_{l=1}^{n_P} \frac{1}{n_P} n_l(t)^* \right]$$

where $E[ ]$ is the expected value operator. It follows that

$$\psi_{nz}^2 = \frac{1}{2} \sum_{m,l=1}^{n_P} E[n_m(t)n_l^*(t)] = \frac{1}{2} \sum_{m,l=1}^{n_P} \psi_{ny}^2 \delta_{ml} = \frac{1}{n_P} \psi_{ny}^2$$

where $\psi_{ny}^2$ is the single pulse noise power and $\delta_{ml}$ is equal to zero for $m \neq l$ and unity for $m = l$. Observation of Eqs. (2.36) and (2.38) shows that the desired signal power after coherent integration is unchanged, while the noise power is reduced by the factor $1/n_P$. Thus, the SNR after coherent integration is improved by $n_P$.

Denote the single pulse SNR required to produce a given probability of detection as $(SNR)_1$. Also, denote $(SNR)_{n_P}$ as the SNR required to produce the same probability of detection when $n_P$ pulses are integrated. It follows that

$$(SNR)_{n_P} = \frac{1}{n_P} (SNR)_1$$

The requirements of knowing the exact phase of each transmitted pulse as well as maintaining coherency during propagation is very costly and challenging to achieve. Thus, radar systems would not utilize coherent integration during search mode, since target micro-dynamics may not be available.

2.4.2. Non-Coherent Integration

Non-coherent integration is often implemented after the envelope detector, also known as the quadratic detector. A block diagram of radar receiver utilizing a square law detector and non-coherent integration is illustrated in Fig. 2.5. In practice, the square law detector is normally used as an approximation to the optimum receiver.
The pdf for the signal \( r(t) \) was derived earlier and it is given in Eq. (2.11). Define a new dimensionless variable \( y \) as

\[
y_n = \frac{r_n}{\psi} \tag{2.40}
\]

and also define

\[
\mathbb{R}_p = \frac{A^2}{\psi^2} = 2SNR \tag{2.41}
\]

It follows that the pdf for the new variable is then given by

\[
f(y_n) = f(r_n) \frac{dr_n}{dy_n} = y_n I_0(y_n \sqrt{\mathbb{R}_p}) \exp\left(\frac{-(y_n^2 + \mathbb{R}_p)}{2}\right) \tag{2.42}
\]

The output of a square law detector for the \( n^{th} \) pulse is proportional to the square of its input, which, after the change of variable in Eq. (2.40), is proportional to \( y_n \). Thus, it is convenient to define a new change variable,

\[
x_n = \frac{1}{2} y_n^2 \tag{2.43}
\]

The pdf for the variable at the output of the square law detector is given by

\[
f(x_n) = f(y_n) \frac{dy_n}{dx_n} = \exp\left(\left(-\frac{x_n + \mathbb{R}_p}{2}\right)\right) I_0(\sqrt{2x_n \mathbb{R}_p}) \tag{2.44}
\]

Non-coherent integration of \( n_p \) pulses is implemented as

\[
z = \sum_{n=1}^{n_p} x_n \tag{2.45}
\]

Since the random variables \( x_n \) are independent, the pdf for the variable \( z \) is
The operator \( \bullet \) symbolically indicates convolution. The characteristic functions for the individual pdfs can then be used to compute the joint pdf in Eq. (2.46). The details of this development are left as an exercise. The result is

\[
f(z) = f(x_1) \bullet f(x_2) \bullet \ldots \bullet f(x_{np})
\]  

(2.46)

The characteristic functions for the individual pdfs can then be used to compute the joint pdf in Eq. (2.46). The details of this development are left as an exercise. The result is

\[
f(z) = \left( \frac{2z}{npR_p} \right)^{(np-1)/2} \exp \left( -z - \frac{1}{2} npR_p \right) I_{np-1} \left( \sqrt{2npzR_p} \right)
\]  

(2.47)

\( I_{np-1} \) is the modified Bessel function of order \( np - 1 \). Therefore, the probability of detection is obtained by integrating \( f(z) \) from the threshold value to infinity. Alternatively, the probability of false alarm is obtained by letting \( R_p \) be zero and integrating the pdf from the threshold value to infinity. Closed form solutions to these integrals are not easily available. Therefore, numerical techniques are often utilized to generate tables for the probability of detection.

**Improvement Factor and Integration Loss**

Denote the SNR that is required to achieve a specific \( P_D \) given a particular \( P_{fa} \) when \( np \) pulses are integrated non-coherently by \((SNR)_{NCI}\). And thus, the single pulse SNR, \((SNR)_1\), is less than \((SNR)_{NCI}\). More precisely,

\[
(SNR)_{NCI} = (SNR)_1 \times I(n_p)
\]  

(2.48)

where \( I(n_p) \) is called the integration improvement factor. An empirically derived expression for the improvement factor that is accurate within 0.8dB is reported in Peebles\(^1\) as

\[
[I(n_p)]_{dB} = 6.79(1 + 0.235P_D) \left( 1 + \frac{\log(1/P_{fa})}{46.6} \right) \log(n_p)
\]  

(2.49)

Fig. 2.6a shows plots of the integration improvement factor as a function of the number of integrated pulses with \( P_D \) and \( P_{fa} \) as parameters, using Eq. (2.49). This plot can be reproduced using the MATLAB program “fig2_6a.m” given in Listing 2.6 in Section 2.11. Note this program uses the MATLAB function “improv_fac.m”, which is given in Listing 2.7 in Section 2.11.

**MATLAB Function “improv_fac.m”**

The function “improv_fac.m” calculates the improvement factor using Eq. (2.49). It is given in Listing 2.7 in Section 2.11. The syntax is as follows:

\[
[impr_of_np] = improv_fac(np, pfa, pd)
\]

---

Figure 2.6a. Improvement factor versus number of non-coherently integrated pulses.

Figure 2.6b. Integration loss versus number of non-coherently integrated pulses.
where

$$L_{NCI} = \frac{n_p}{I(n_p)}$$  \hspace{1cm} (2.50)

Figure 2.6b shows a plot of the integration loss versus $n_p$. This figure can be reproduced using MATLAB program "fig2_6b.m" given in Listing 2.8 in Section 2.11. It follows that, when non-coherent integration is utilized, the corresponding SNR required to achieve a certain $P_D$ given a specific $P_{fa}$ is now given by

$$\text{(SNR)}_{NCI} = \frac{(n_p \times \text{(SNR)}_1)}{L_{NCI}}$$  \hspace{1cm} (2.51)

which is very similar to Eq. (1.86) derived in Chapter 1.

2.4.3. Mini Design Case Study 2.1

An L-band radar has the following specifications: operating frequency $f_0 = 1.5 \text{GHz}$, operating bandwidth $B = 2 \text{MHz}$, noise figure $F = 8 \text{dB}$, system losses $L = 4 \text{dB}$, time of false alarm $T_{fa} = 12 \text{ minutes}$, detection range $R = 12 \text{Km}$, the minimum required SNR is $\text{SNR} = 13.85 \text{dB}$, antenna gain $G = 5000$, and target RCS $\sigma = 1 \text{m}^2$. (a) Determine the PRF $f_r$, the pulsewidth $\tau$, the peak power $P_t$, the probability of false alarm $P_{fa}$, the corresponding $P_D$, and the minimum detectable signal level $S_{min}$. (b) How can you reduce the transmitter power to achieve the same performance when 10 pulses are integrated non-coherently? (c) If the radar operates at a shorter range in the single pulse mode, find the new probability of detection when the range decreases to 9Km.

A Solution

Assume that the maximum detection corresponds to the unambiguous range. From that the PRF is computed as
The pulsewidth is proportional to the inverse of the bandwidth,

\[ f_r = \frac{c}{2R_u} = \frac{3 \times 10^8}{2 \times 12000} = 12.5 \text{KHz} \]

The probability of false alarm is

\[ \tau = \frac{1}{B} = \frac{1}{2 \times 10^6} = 0.5 \mu s \]

The probability of false alarm is

\[ P_{fa} = \frac{1}{BT_{fa}} = \frac{1}{2 \times 10^6 \times 12 \times 60} = 6.94 \times 10^{-10} \]

It follows that by using MATLAB function “marcumsq.m” the probability of detection is calculated from

\[
Q\left(\frac{\sqrt{A^2/\psi^2} \cdot \sqrt{2 \ln \left(\frac{1}{P_{fa}}\right)}}{\ln 6.494}\right)
\]

with the following syntax

\[
\text{marcumsq}(\alpha, \beta)
\]

where

\[
\alpha = \sqrt{2} \times \sqrt{10^{13.85/10}} = 6.9665
\]

\[
\beta = \sqrt{2 \ln \left(\frac{1}{6.94 \times 10^{-10}}\right)} = 6.494
\]

Remember that \(A^2/\psi^2 = 2\text{SNR}\). Thus, the detection probability is

\[
P_D = \text{marcumsq}(6.9665, 6.944) = 0.508
\]

Using the radar equation one can calculate the radar peak power. More precisely,

\[
P_t = \text{SNR} \frac{(4\pi)^3 R^4 k T_0 B F L}{G^2 \lambda^2 \sigma} \Rightarrow
\]

\[
P_t = 10^{1.385} \frac{(4\pi)^3 \times 12000^4 \times 1.38 \times 10^{-23} \times 290 \times 2 \times 10^6 \times 6.309 \times 2.511}{5000^2 \times 0.2^2 \times 1}
\]

\[
= 126.61 \text{Watts}
\]

And the minimum detectable signal is
When 10 pulses are integrated non-coherently, the corresponding improvement factor is calculated from the MATLAB function “improv_fac.m” using the following syntax

\[
\text{improv\_fac}(10,1e-11,0.5)
\]

which yields \( I(10) = 6 \Rightarrow 7.78\,\text{dB} \). Consequently, by keeping the probability of detection the same (with and without integration) the SNR can be reduced by a factor of almost 6 dB (13.85 - 7.78). The integration loss associated with a 10-pulse non-coherent integration is calculated from Eq. (2.50) as

\[
L_{\text{NCI}} = \frac{n_p}{I(10)} = \frac{10}{6} = 1.67 \Rightarrow 2.2\,\text{dB}
\]

Thus the net single pulse SNR with 10-pulse non-coherent integration is

\[
(SNR)_{\text{NCI}} = 13.85 - 7.78 + 2.2 = 8.27\,\text{dB}.
\]

Finally, the improvement in the SNR due to decreasing the detection range to 9 Km is

\[
(SNR)_{9\,\text{Km}} = 10\log\left(\frac{12000}{9000}\right)^4 + 13.85 = 18.85\,\text{dB}.
\]

### 2.5. Detection of Fluctuating Targets

So far the probability of detection calculations assumed a constant target cross section (non-fluctuating target). This work was first analyzed by Marcum.\(^1\) Swerling\(^2\) extended Marcum’s work to four distinct cases that account for variations in the target cross section. These cases have come to be known as Swerling models. They are: Swerling I, Swerling II, Swerling III, and Swerling IV. The constant RCS case analyzed by Marcum is widely known as Swerling 0 or equivalently Swerling V. Target fluctuation lowers the probability of detection, or equivalently reduces the SNR.

---

Swerling I targets have constant amplitude over one antenna scan; however, a Swerling I target amplitude varies independently from scan to scan according to a Chi-square probability density function with two degrees of freedom. The amplitude of Swerling II targets fluctuates independently from pulse to pulse according to a Chi-square probability density function with two degrees of freedom. Target fluctuation associated with a Swerling III model is similar to Swerling I, except in this case the target power fluctuates independently from pulse to pulse according to a Chi-square probability density function with four degrees of freedom. Finally, the fluctuation of Swerling IV targets is from pulse to pulse according to a Chi-square probability density function with four degrees of freedom. Swerling showed that the statistics associated with Swerling I and II models apply to targets consisting of many small scatterers of comparable RCS values, while the statistics associated with Swerling III and IV models apply to targets consisting of one large RCS scatterer and many small equal RCS scatterers. Non-coherent integration can be applied to all four Swerling models; however, coherent integration cannot be used when the target fluctuation is either Swerling II or Swerling IV. This is because the target amplitude decorrelates from pulse to pulse (fast fluctuation) for Swerling II and IV models, and thus phase coherency cannot be maintained.

The Chi-square pdf with $2N$ degrees of freedom can be written as

$$f(\sigma) = \frac{N}{(N-1)!} \left(\frac{N\sigma}{\bar{\sigma}}\right)^{N-1} \exp\left(-\frac{N\sigma}{\bar{\sigma}}\right)$$

(2.52)

where $\bar{\sigma}$ is the average RCS value. Using this equation, the pdf associated with Swerling I and II targets can be obtained by letting $N = 1$, which yields a Rayleigh pdf. More precisely,

$$f(\sigma) = \frac{1}{\sigma} \exp\left(-\frac{\sigma}{\bar{\sigma}}\right) \quad \sigma \geq 0$$

(2.53)

Letting $N = 2$ yields the pdf for Swerling III and IV type targets,

$$f(\sigma) = \frac{4\sigma}{\sigma^2} \exp\left(-\frac{2\sigma}{\bar{\sigma}}\right) \quad \sigma \geq 0$$

(2.54)

The probability of detection for a fluctuating target is computed in a similar fashion to Eq. (2.23), except in this case $f(r)$ is replaced by the conditional pdf $f(r/\sigma)$. Performing the analysis for the general case (i.e., using Eq. (2.47)) yields

$$f(z/\sigma) = \left(\frac{2z}{n_p\sigma^2/\psi^2}\right)^{(n_p-1)/2} \exp\left(-z - \frac{1}{2} n_p \frac{\sigma^2}{\psi^2}\right) J_{n_p-1}\left(2n_p z \frac{\sigma^2}{\psi^2}\right)$$

(2.55)
To obtain \( f(z) \) use the relations
\[
f(z, \sigma) = f(z/\sigma)f(\sigma) \quad (2.56)
\]
\[
f(z) = \int f(z, \sigma)d\sigma \quad (2.57)
\]
Finally, using Eq. (2.56) in Eq. (2.57) produces
\[
f(z) = \int f(z/\sigma)f(\sigma)d\sigma \quad (2.58)
\]
where \( f(z/\sigma) \) is defined in Eq. (2.55) and \( f(\sigma) \) is in either Eq. (2.53) or (2.54). The probability of detection is obtained by integrating the pdf derived from Eq. (2.58) from the threshold value to infinity. Performing the integration in Eq. (2.58) leads to the incomplete Gamma function.

### 2.5.1. Threshold Selection

When only a single pulse is used, the detection threshold \( V_T \) is related to the probability of false alarm \( P_{fa} \) as defined in Eq. (2.19). DiFranco and Rubin\(^1\) derived a general form relating the threshold and \( P_{fa} \) for any number of pulses when non-coherent integration is used. It is
\[
P_{fa} = 1 - \Gamma_1 \left( \frac{V_T}{\sqrt{n_p}}, n_p - 1 \right) \quad (2.59)
\]
where \( \Gamma_1 \) is used to denote the incomplete Gamma function. It is given by
\[
\Gamma_1 \left( \frac{V_T}{\sqrt{n_p}}, n_p - 1 \right) = \int_0^{V_T/\sqrt{n_p}} e^{-\gamma} \frac{\gamma^{(n_p-1)-1}}{(n_p-1)!} d\gamma \quad (2.60)
\]
Note that the limiting values for the incomplete Gamma function are
\[
\Gamma_1(0, N) = 0 \quad \Gamma_1(\infty, N) = 1 \quad (2.61)
\]
For our purposes, the incomplete Gamma function can be approximated by
\[
\Gamma_1 \left( \frac{V_T}{\sqrt{n_p}}, n_p - 1 \right) = 1 - \frac{V_T^{n_p-1}}{(n_p-1)!} \left[ 1 + \frac{n_p-1}{V_T} + \frac{(n_p-1)(n_p-2)}{V_T^2} + \ldots + \frac{(n_p-1)!}{V_T^{n_p-1}} \right] \quad (2.62)
\]

---

The threshold value \( V_T \) can then be approximated by the recursive formula used in the Newton-Raphson method. More precisely,

\[
V_{T,m} = V_{T,m-1} - \frac{G(V_{T,m-1})}{G'(V_{T,m-1})}; \quad m = 1, 2, 3, \ldots
\]  

(2.63)

The iteration is terminated when \( |V_{T,m} - V_{T,m-1}| < V_{T,m-1}/10000.0 \). The functions \( G \) and \( G' \) are

\[
G(V_{T,m}) = (0.5)^{n_p/n_{fa}} - \Gamma_I(V_T, n_P)
\]  

(2.64)

\[
G'(V_{T,m}) = -\frac{e^{-V_T} V_T^{n_p-1}}{(n_p-1)!}
\]  

(2.65)

The initial value for the recursion is

\[
V_{T,0} = n_P - \sqrt{n_P} + 2.3 \sqrt{-\log P_{fa}} (\sqrt{-\log P_{fa}} + \sqrt{n_P} - 1)
\]  

(2.66)

**MATLAB Function “incomplete_gamma.m”**

In general, the incomplete Gamma function for some integer \( N \) is

\[
\Gamma_I(x, N) = \int_0^x e^{-v} v^{N-1} \frac{dv}{(N-1)!}
\]  

(2.67)

The function “incomplete_gamma.m” implements Eq. (2.67). It is given in Listing 2.9 in Section 2.11. Note that this function uses the MATLAB function “factor.m” which is given in Listing 2.10. The function “factor.m” calculates the factorial of an integer. Fig. 2.7 shows the incomplete Gamma function for \( N = 1, 3, 6, 10 \). This figure can be reproduced using the MATLAB program “fig2_7.m” given in Listing 2.11. The syntax for this function is as follows:

\[
[value] = incomplete\_gamma \ (x, \ N)
\]

where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>variable input to ( \Gamma_I(x, N) )</td>
<td>units of ( x )</td>
<td>input</td>
</tr>
<tr>
<td>( N )</td>
<td>variable input to ( \Gamma_I(x, N) )</td>
<td>none / integer</td>
<td>input</td>
</tr>
<tr>
<td>value</td>
<td>( \Gamma_I(x, N) )</td>
<td>none</td>
<td>output</td>
</tr>
</tbody>
</table>
MATLAB Function “threshold.m”

The function “threshold.m” calculates the threshold using the recursive formula used in the Newton-Raphson method. It is given in Listing 2.12 in Section 2.11. The syntax is as follows:

\[ [pfa, vt] = \text{threshold}(nfa, np) \]

where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfa</td>
<td>Marcum’s false alarm number</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>np</td>
<td>number of integrated pulses</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>pfa</td>
<td>probability of false alarm</td>
<td>none</td>
<td>output</td>
</tr>
<tr>
<td>vt</td>
<td>threshold value</td>
<td>none</td>
<td>output</td>
</tr>
</tbody>
</table>

Fig. 2.8 shows plots of the threshold value versus the number of integrated pulses for several values of \( n_{fa} \); remember that \( P_{fa} \approx \ln(2)/n_{fa} \). This figure can be reproduced using MATLAB program “fig2_8.m” given in Listing 2.13. This program uses both “threshold.m” and “incomplete_gamma”. 
2.6. Probability of Detection Calculation

Marcum defined the probability of false alarm for the case when \( n_p > 1 \) as

\[
P_{fa} \approx \ln(2) \frac{n_p}{n_{fa}} \tag{2.68}
\]

The single pulse probability of detection for non-fluctuating targets is given in Eq. (2.24). When \( n_p > 1 \), the probability of detection is computed using the Gram-Charlier series. In this case, the probability of detection is

\[
P_D \approx \text{erfc}\left(\frac{V}{\sqrt{2}}\right) \frac{e^{-V^2/2}}{\sqrt{2\pi}} \left[ C_3(V^2 - 1) + C_4V(3 - V^2) ight. \\
\left. - C_6V(V^4 - 10V^2 + 15) \right] \tag{2.69}
\]

where the constants \( C_3, C_4, \) and \( C_6 \) are the Gram-Charlier series coefficients, and the variable \( V \) is

\[
V = \frac{V_T - n_p(1 + \text{SNR})}{\varpi} \tag{2.70}
\]

In general, values for \( C_3, C_4, C_6, \) and \( \varpi \) vary depending on the target fluctuation type.

Figure 2.8. Threshold \( V_T \) versus \( n_p \) for several values of \( n_{fa} \).
2.6.1. Detection of Swerling V Targets

For Swerling V (Swerling 0) target fluctuations, the probability of detection is calculated using Eq. (2.69). In this case, the Gram-Charlier series coefficients are

$$C_3 = \frac{\text{SNR} + 1/3}{\sqrt{n_p(2\text{SNR} + 1)^{1.5}}}$$

(2.71)

$$C_4 = \frac{\text{SNR} + 1/4}{n_p(2\text{SNR} + 1)^2}$$

(2.72)

$$C_6 = C_3^2/2$$

(2.73)

$$\varpi = \sqrt{n_p(2\text{SNR} + 1)}$$

(2.74)

**MATLAB Function “pd_swerling5.m”**

The function “pd_swerling5.m” calculates the probability of detection for Swerling V targets. It is given in Listing 2.14. The syntax is as follows:

$$[pd] = \text{pd}_\text{sw}erling5 (input1, \text{indicator}, np, \text{snr})$$

where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>input1</td>
<td>$P_{fa}$ or $n_{fa}$</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>indicator</td>
<td>1 when input1 = $P_{fa}$</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td></td>
<td>2 when input1 = $n_{fa}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>np</td>
<td>number of integrated pulses</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>snr</td>
<td>$\text{SNR}$</td>
<td>dB</td>
<td>input</td>
</tr>
<tr>
<td>pd</td>
<td>probability of detection</td>
<td>none</td>
<td>output</td>
</tr>
</tbody>
</table>

Fig. 2.9 shows a plot for the probability of detection versus SNR for cases $n_p = 1, 10$. This figure can be reproduced using the MATLAB program “fig2_9.m”. It is given in Listing 2.15 in Section 2.11.

Note that it requires less SNR, with ten pulses integrated non-coherently, to achieve the same probability of detection as in the case of a single pulse. Hence, for any given $P_D$ the SNR improvement can be read from the plot. Equivalently, using the function “improv_fac.m” leads to about the same result. For example, when $P_D = 0.8$ the function “improv_fac.m” gives an SNR improvement factor of $I(10) \approx 8.55\text{dB}$. Fig. 2.9 shows that the ten pulse SNR is about $6.03\text{dB}$. Therefore, the single pulse SNR is about (from Eq. (2.49)) $14.5\text{dB}$, which can be read from the figure.
2.6.2. Detection of Swerling I Targets

The exact formula for the probability of detection for Swerling I type targets was derived by Swerling. It is

\[
P_D = e^{-\frac{V_T}{1 + \text{SNR}}}; \quad n_p = 1
\]  

\[
P_D = 1 - \Gamma \left(V_T, n_p - 1\right) + \left(1 + \frac{1}{n_p \text{SNR}}\right)^{n_p - 1} \Gamma \left(\frac{V_T}{1 + \frac{1}{n_p \text{SNR}}}, n_p - 1\right) \times e^{-\frac{V_T}{1 + n_p \text{SNR}}}; \quad n_p > 1
\]

**MATLAB Function “pd_swerling1.m”**

The function “pd_swerling1.m” calculates the probability of detection for Swerling I type targets. It is given in Listing 2.16 in Section 2.11. The syntax is as follows:

\[
[pd] = \text{pd}_\text{swerling1}(nfa, np, snr)
\]
where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfa</td>
<td>Marcum’s false alarm number</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>np</td>
<td>number of integrated pulses</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>snr</td>
<td>SNR</td>
<td>dB</td>
<td>input</td>
</tr>
<tr>
<td>pd</td>
<td>probability of detection</td>
<td>none</td>
<td>output</td>
</tr>
</tbody>
</table>

Fig. 2.10 shows a plot of the probability of detection as a function of SNR for \( n_p = 1 \) and \( P_{fa} = 10^{-9} \) for both Swerling I and V type fluctuations. Note that it requires more SNR, with fluctuation, to achieve the same \( P_D \) as in the case with no fluctuation. This figure can be reproduced using MATLAB program “fig2_10.m” given in Listing 2.17.

Fig. 2.11a shows a plot of the probability of detection versus SNR for \( n_p = 1, 10, 50, 100 \), where \( P_{fa} = 10^{-8} \). Fig. 2.11b is similar to Fig. 2.11a; in this case \( P_{fa} = 10^{-11} \). These figures can be reproduced using MATLAB program “fig2_11ab.m” given in Listing 2.18.
Figure 2.11a. Probability of detection versus SNR. Swerling I. $P_{fa} = 10^{-8}$.

Figure 2.11b. Probability of detection versus SNR. Swerling I. $P_{fa} = 10^{-11}$.
2.6.3. Detection of Swerling II Targets

In the case of Swerling II targets, the probability of detection is given by

\[ P_D = 1 - \Gamma_i \left( \frac{V_T}{(1 + SNR)} , n_p \right) ; \quad n_p \leq 50 \]  

(2.77)

For the case when \( n_p > 50 \) Eq. (2.69) is used to compute the probability of detection. In this case,

\[ C_3 = \frac{1}{3 \sqrt{n_p}} , \quad C_6 = \frac{C_3^2}{2} \]  

(2.78)

\[ C_4 = \frac{1}{4n_p} \]  

(2.79)

\[ \varpi = \sqrt{n_p} \left( 1 + SNR \right) \]  

(2.80)

**MATLAB Function “pd_swerling2.m”**

The function “pd_swerling2.m” calculates \( P_D \) for Swerling II type targets. It is given in Listing 2.19 in Section 2.11. The syntax is as follows:

\[ [pd] = pd_swerling2 \left( nfa, np, snr \right) \]

where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfa</td>
<td>Marcum’s false alarm number</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>np</td>
<td>number of integrated pulses</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>snr</td>
<td>SNR</td>
<td>dB</td>
<td>input</td>
</tr>
<tr>
<td>pd</td>
<td>probability of detection</td>
<td>none</td>
<td>output</td>
</tr>
</tbody>
</table>

Fig. 2.12 shows a plot of the probability of detection as a function of SNR for \( n_p = 1, 10, 50, 100 \), where \( P_{fa} = 10^{-10} \). This figure can be reproduced using MATLAB program “fig2_12.m” given in Listing 2.20.

2.6.4. Detection of Swerling III Targets

The exact formulas, developed by Marcum, for the probability of detection for Swerling III type targets when \( n_p = 1, 2 \) is

\[ P_D = \exp \left( \frac{-V_T}{1 + n_pSNR/2} \right) \left( 1 + \frac{2}{n_pSNR} \right)^{n_p-2} \times K_0 \]  

(2.81)

\[ K_0 = 1 + \frac{V_T}{1 + n_pSNR/2} - \frac{2}{n_pSNR} (n_p - 2) \]
For $n_p > 2$ the expression is

$$P_D = \frac{V_T^{n_p-1} e^{-V_T}}{(1 + n_p \text{SNR}/2)(n_p-2)!} + 1 - \Gamma_i(V_T, n_p-1) + K_0$$

$$\times \Gamma_i \left( \frac{V_T}{1 + 2/n_p \text{SNR}}, n_p-1 \right)$$

(2.82)

**MATLAB Function “pd_swerling3.m”**

The function “pd_swerling3.m” calculates $P_D$ for Swerling III type targets. It is given in Listing 2.21 in Section 2.11. The syntax is as follows:

```
[pd] = pd_swerling3 (nfa, np, snr)
```

where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfa</td>
<td>Marcum’s false alarm number</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>np</td>
<td>number of integrated pulses</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>snr</td>
<td>SNR</td>
<td>dB</td>
<td>input</td>
</tr>
<tr>
<td>pd</td>
<td>probability of detection</td>
<td>none</td>
<td>output</td>
</tr>
</tbody>
</table>
Fig. 2.13 shows a plot of the probability of detection as a function of SNR for $n_P = 1, 10, 50, 100$, where $P_{fa} = 10^{-9}$. This figure can be reproduced using MATLAB program “fig2_13.m” given in Listing 2.22.

![Figure 2.13. Probability of detection versus SNR. Swerling III. $P_{fa} = 10^{-9}$.](image)

### 2.6.5. Detection of Swerling IV Targets

The expression for the probability of detection for Swerling IV targets for $n_P < 50$ is

$$P_D = 1 - \left[ \gamma_0 + \left( \frac{SNR}{2} \right) n_P \gamma_1 + \left( \frac{SNR}{2} \right)^2 n_P (n_P - 1) \frac{\gamma_2}{2!} + \cdots \right]$$

$$+ \left( \frac{SNR}{2} \right)^{n_P} \frac{\gamma_{n_P}}{n_P!} \left( 1 + \frac{SNR}{2} \right)^{-n_P}$$

(2.83)

where

$$\gamma_i = \Gamma_i \left( \frac{V_T}{1 + (SNR)/2}, n_P + i \right)$$

(2.84)

By using the recursive formula

$$\Gamma_i(x, i + 1) = \Gamma_i(x, i) - \frac{x^i}{i! \exp(x)}$$

(2.85)

then only $\gamma_0$ needs to be calculated using Eq. (2.84) and the rest of $\gamma_i$ are calculated from the following recursion:
For the case when \( n_p \geq 50 \), the Gram-Charlier series and Eq. (2.69) can be used to calculate the probability of detection. In this case,

\[
\gamma_i = \gamma_{i-1} - A_i \quad ; \quad i > 0
\]

\[
A_i = \frac{V_T/(1 + (SNR)/2)}{n_p + i - 1} A_{i-1} \quad ; \quad i > 1
\]

\[
A_1 = \frac{(V_T/(1 + (SNR)/2))^{n_p}}{n_p! \exp(V_T/(1 + (SNR)/2))}
\]

\[
\gamma_0 = \Gamma_i \left( \frac{V_T}{1 + (SNR)/2}, n_p \right)
\]

MATLAB Function “pd_swerling4.m”

The function “pd_swerling4.m” calculates \( P_D \) for Swerling IV type targets. It is given in Listing 2.23 in Section 2.11. The syntax is as follows:

\[
[pd] = pd_swerling4 \left( nfa, np, snr \right)
\]

where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfa</td>
<td>Marcum’s false alarm number</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>np</td>
<td>number of integrated pulses</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>snr</td>
<td>SNR</td>
<td>dB</td>
<td>input</td>
</tr>
<tr>
<td>pd</td>
<td>probability of detection</td>
<td>none</td>
<td>output</td>
</tr>
</tbody>
</table>

Figure 2.14 shows a plot of the probability of detection as a function of SNR for \( n_p = 1, 10, 50, 100 \), where \( P_{fa} = 10^{-9} \). This figure can be reproduced using MATLAB program “fig2_14.m” given in Listing 2.24.
2.7. The Radar Equation Revisited

The radar equation developed in Chapter 1 assumed a constant target RCS and did not account for integration loss. In this section, a more comprehensive form of the radar equation is introduced. In this case, the radar equation is given by

\[
R^4 = \frac{P_{av} G_t G_r \lambda^2 \sigma I(n_p)}{(4\pi)^3 k T_e F B \tau f_r L_t L_f (SNR)_1}
\]

(2.94)

where \(P_{av} = P_t \tau f_r\) is the average transmitted power, \(P_t\) is the peak transmitted power, \(\tau\) is pulsewidth, \(f_r\) is PRF, \(G_t\) is transmitting antenna gain, \(G_r\) is receiving antenna gain, \(\lambda\) is wavelength, \(\sigma\) is target cross section, \(I(n_p)\) is improvement factor, \(n_p\) is the number of integrated pulses, \(k\) is Boltzman’s constant, \(T_e\) is effective noise temperature, \(F\) is the system noise figure, \(B\) is receiver bandwidth, \(L_t\) is total system losses including integration loss, \(L_f\) is loss due to target fluctuation, and \((SNR)_1\) is the minimum single pulse SNR required for detection.

The fluctuation loss, \(L_f\), can be viewed as the amount of additional SNR required to compensate for the SNR loss due to target fluctuation, given a specific \(P_D\) value. This was demonstrated for a Swerling I fluctuation in Fig.
2.10. Kanter developed an exact analysis for calculating the fluctuation loss. In this text the authors will take advantage of the computational power of MATLAB and the MATLAB functions developed for this text to numerically calculate the amount of fluctuation loss with an accuracy of $0.005\, dB$ or better. For this purpose the MATLAB function “fluct_loss.m” was developed. It is given in Listing 2.25 in Section 2.11. Its syntax is as follows:

$$[Lf, Pd_{Sw5}] = \text{fluct} \_\text{loss}(pd, pfa, np, sw\_case)$$

where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>pd</td>
<td>desired probability of detection</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>pfa</td>
<td>probability of false alarm</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>np</td>
<td>number of pulses</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>sw_case</td>
<td>1, 2, 3, or 4 depending on the desired Swerling case</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>Lf</td>
<td>fluctuation loss</td>
<td>dB</td>
<td>output</td>
</tr>
<tr>
<td>Pd_Sw5</td>
<td>Probability of detection corresponding to a Swerling V case</td>
<td>none</td>
<td>output</td>
</tr>
</tbody>
</table>

For example, using the syntax

$$[Lf, Pd_{Sw5}] = \text{fluct} \_\text{loss}(0.65, 1e-9, 10, 1)$$

will calculate the SNR corresponding to both Swerling V and Swerling I fluctuation when the desired probability of detection $P_D = 0.65$ and probability of false alarm $P_{fa} = 10^{-9}$ and 10 pulses of non-coherent integration. The following is a reprint of the output:

$$PD_{SW5} = 0.65096989459928 \quad \text{SNR}_{SW5} = 5.52499999999990$$  
$$PD_{SW1} = 0.65019653294095 \quad \text{SNR}_{SW1} = 8.32999999999990$$  
$$Lf = 2.80500000000000$$

Note that a negative value for $Lf$ indicates a fluctuation SNR gain instead of loss. Finally, it must be noted that the function “fluct_loss.m” always assumes non-coherent integration. Fig. 2.15 shows a plot for the additional SNR (or fluctuation loss) required to achieve a certain probability of detection. This figure can be reproduced using MATLAB program “fig2_16.m” given in Listing 2.26 in Section 2.11.

2.8. Cumulative Probability of Detection

Denote the range at which the single pulse SNR is unity (0 dB) as \( R_0 \), and refer to it as the reference range. Then, for a specific radar, the single pulse SNR at \( R_0 \) is defined by the radar equation and is given by

\[
(SNR)_{R_0} = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 kT_0 BFLR_0^4} = 1 \tag{2.95}
\]

The single pulse SNR at any range \( R \) is

\[
SNR = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 kT_0 BFLR^4} \tag{2.96}
\]

Dividing Eq. (2.96) by Eq. (2.95) yields

\[
\frac{SNR}{(SNR)_{R_0}} = \left(\frac{R_0}{R}\right)^4 \tag{2.97}
\]

Therefore, if the range \( R_0 \) is known then the SNR at any other range \( R \) is
Also, define the range $R_{50}$ as the range at which $P_D = 0.5 = P_{50}$. Normally, the radar unambiguous range $R_u$ is set equal to $2R_{50}$.

The cumulative probability of detection refers to detecting the target at least once by the time it is at range $R$. More precisely, consider a target closing on a scanning radar, where the target is illuminated only during a scan (frame). As the target gets closer to the radar, its probability of detection increases since the SNR is increased. Suppose that the probability of detection during the $n$th frame is $P_{D_n}$; then, the cumulative probability of detecting the target at least once during the $n$th frame (see Fig. 2.16) is given by

$$ P_{C_n} = 1 - \prod_{i=1}^{n} (1 - P_{D_i}) \quad (2.99) $$

$P_{D_i}$ is usually selected to be very small. Clearly, the probability of not detecting the target during the $n$th frame is $1 - P_{C_n}$. The probability of detection for the $i$th frame, $P_{D_i}$, is computed as discussed in the previous section.

![Figure 2.16. Detecting a target in many frames.](image)

### 2.8.1. Mini Design Case Study 2.2

A radar detects a closing target at $R = 10Km$, with probability of detection $P_D$ equal to 0.5. Assume $P_{fa} = 10^{-7}$. Compute and sketch the single look probability of detection as a function of normalized range (with respect to $R = 10Km$), over the interval $(2 - 20)Km$. If the range between two successive frames is $1Km$, what is the cumulative probability of detection at $R = 8Km$?
A Solution:

From the function “marcumsq.m” the SNR corresponding to $P_D = 0.5$ and $P_{fa} = 10^{-7}$ is approximately 12dB. By using a similar analysis to that which led to Eq. (2.98), we can express the SNR at any range $R$ as

$$(SNR)_R = (SNR)_{10} + 40 \log\frac{10}{R} = 52 - 40 \log R$$

By using the function “marcumsq.m” we can construct the following table:

<table>
<thead>
<tr>
<th>R Km</th>
<th>(SNR) dB</th>
<th>$P_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>39.09</td>
<td>0.999</td>
</tr>
<tr>
<td>4</td>
<td>27.9</td>
<td>0.999</td>
</tr>
<tr>
<td>6</td>
<td>20.9</td>
<td>0.999</td>
</tr>
<tr>
<td>8</td>
<td>15.9</td>
<td>0.999</td>
</tr>
<tr>
<td>9</td>
<td>13.8</td>
<td>0.9</td>
</tr>
<tr>
<td>10</td>
<td>12.0</td>
<td>0.5</td>
</tr>
<tr>
<td>11</td>
<td>10.3</td>
<td>0.25</td>
</tr>
<tr>
<td>12</td>
<td>8.8</td>
<td>0.07</td>
</tr>
<tr>
<td>14</td>
<td>6.1</td>
<td>0.01</td>
</tr>
<tr>
<td>16</td>
<td>3.8</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>20</td>
<td>0.01</td>
<td>$\varepsilon$</td>
</tr>
</tbody>
</table>

where $\varepsilon$ is very small. A sketch of $P_D$ versus normalized range is shown in Fig. 2.17.

The cumulative probability of detection is given in Eq. (2.95), where the probability of detection of the first frame is selected to be very small. Thus, we can arbitrarily choose frame 1 to be at $R = 16$Km. Note that selecting a different starting point for frame 1 would have a negligible effect on the cumulative probability (we only need $P_{D_1}$ to be very small). Below is a range listing for frames 1 through 9, where frame 9 corresponds to $R = 8$Km. The cumulative probability of detection at 8 Km is then

$$P_{C_9} = 1 - (1 - 0.999)(1 - 0.9)(1 - 0.5)(1 - 0.25)(1 - 0.07)(1 - 0.01)(1 - \varepsilon)^2 \approx 0.9998$$
2.9. Constant False Alarm Rate (CFAR)

The detection threshold is computed so that the radar receiver maintains a constant pre-determined probability of false alarm. Eq. (2.19b) gives the relationship between the threshold value \( V_T \) and the probability of false alarm \( P_{fa} \), and for convenience is repeated here as Eq. (2.100):

\[
V_T = \sqrt{\frac{2\psi^2 \ln \left( \frac{1}{P_{fa}} \right)}{\rho}}
\]

If the noise power \( \psi^2 \) is assumed to be constant, then a fixed threshold can satisfy Eq. (2.100). However, due to many reasons this condition is rarely true. Thus, in order to maintain a constant probability of false alarm the threshold value must be continuously updated based on the estimates of the noise variance. The process of continuously changing the threshold value to maintain a constant probability of false alarm is known as Constant False Alarm Rate (CFAR).

Three different types of CFAR processors are primarily used. They are adaptive threshold CFAR, nonparametric CFAR, and nonlinear receiver techniques. Adaptive CFAR assumes that the interference distribution is known and approximates the unknown parameters associated with these distributions. Nonparametric CFAR processors tend to accommodate unknown interference distributions. Nonlinear receiver techniques attempt to normalize the root mean square amplitude of the interference. In this book only analog Cell-Averaging CFAR (CA-CFAR) technique is examined. The analysis presented in this section closely follows Urkowitz\(^1\).

---

2.9.1. Cell-Averaging CFAR (Single Pulse)

The CA-CFAR processor is shown in Fig. 2.18. Cell averaging is performed on a series of range and/or Doppler bins (cells). The echo return for each pulse is detected by a square law detector. In analog implementation these cells are obtained from a tapped delay line. The Cell Under Test (CUT) is the central cell. The immediate neighbors of the CUT are excluded from the averaging process due to a possible spillover from the CUT. The output of $M$ reference cells ($M/2$ on each side of the CUT) is averaged. The threshold value is obtained by multiplying the averaged estimate from all reference cells by a constant $K_0$ (used for scaling). A detection is declared in the CUT if

$$Y_1 \geq K_0 Z$$  \hspace{1cm} (2.101)

Cell-averaging CFAR assumes that the target of interest is in the CUT and all reference cells contain zero mean independent Gaussian noise of variance $\psi^2$. Therefore, the output of the reference cells, $Z$, represents a random variable with gamma probability density function (special case of the Chi-square) with $2M$ degrees of freedom. In this case, the gamma pdf is

$$f(z) = \frac{z^{(M/2) - 1} e^{-z/2\psi^2}}{2^{M/2} \psi^M \Gamma(M/2)} ; \ z > 0$$ \hspace{1cm} (2.102)

Figure 2.18. Conventional CA-CFAR.

The probability of false alarm corresponding to a fixed threshold was derived earlier. When CA-CFAR is implemented, then the probability of false
alarm can be derived from the conditional false alarm probability, which is averaged over all possible values of the threshold in order to achieve an unconditional false alarm probability. The conditional probability of false alarm when $y = V_T$ can be written as

$$P_{fa}(V_T = y) = e^{-y/\psi^2} \quad (2.103)$$

It follows that the unconditional probability of false alarm is

$$P_{fa} = \int_0^\infty P_{fa}(V_T = y)f(y)dy \quad (2.104)$$

where $f(y)$ is the pdf of the threshold, which except for the constant $K_0$ is the same as that defined in Eq. (2.102). Therefore,

$$f(y) = \frac{y^{M-1}e^{-y/(2K_0\psi^2)}}{(2K_0\psi^2)^M \Gamma(M)} \quad ; \quad y \geq 0 \quad (2.105)$$

Performing the integration in Eq. (2.104) yields

$$P_{fa} = \frac{1}{(1 + K_0)^M} \quad (2.106)$$

Observation of Eq. (2.106) shows that the probability of false alarm is now independent of the noise power, which is the objective of CFAR processing.

2.9.2. Cell-Averaging CFAR with Non-Coherent Integration

In practice, CFAR averaging is often implemented after non-coherent integration, as illustrated in Fig. 2.19. Now, the output of each reference cell is the sum of $n_p$ squared envelopes. It follows that the total number of summed reference samples is $Mn_p$. The output $Y_1$ is also the sum of $n_p$ squared envelopes. When noise alone is present in the CUT, $Y_1$ is a random variable whose pdf is a gamma distribution with $2n_p$ degrees of freedom. Additionally, the summed output of the reference cells is the sum of $Mn_p$ squared envelopes. Thus, $Z$ is also a random variable which has a gamma pdf with $2Mn_p$ degrees of freedom.

The probability of false alarm is then equal to the probability that the ratio $Y_1/Z$ exceeds the threshold. More precisely,

$$P_{fa} = Prob\{Y_1/Z > K_1\} \quad (2.107)$$
Eq. (2.107) implies that one must first find the joint pdf for the ratio $Y_1/Z$. However, this can be avoided if $P_{fa}$ is first computed for a fixed threshold value $V_T$, then averaged over all possible values of the threshold. Therefore, let the conditional probability of false alarm when $y = V_T$ be $P_{fa}(V_T = y)$. It follows that the unconditional false alarm probability is given by

$$P_{fa} = \int_{0}^{\infty} P_{fa}(V_T = y)f(y)dy$$

(2.108)

where $f(y)$ is the pdf of the threshold. In view of this, the probability density function describing the random variable $K_1Z$ is given by

$$f(y) = \frac{(y/K_1)^{Mn_p-1}e^{-y/2K_0\psi^2}}{(2\psi^2)^{Mn_p}K_1\Gamma(Mn_p)}; \quad y \geq 0$$

(2.109)

It can be shown that in this case the probability of false alarm is independent of the noise power and is given by

$$P_{fa} = \frac{1}{(1 + K_1)^{Mn_p}} \sum_{k=0}^{nP-1} \frac{\Gamma(Mn_p + k)(K_1)^k}{\Gamma(Mn_p)(1 + K_1)^k}$$

(2.110)

---

Figure 2.19. Conventional CA-CFAR with non-coherent integration.
which is identical to Eq. (2.106) when $K_1 = K_0$ and $n_P = 1$.

### 2.10. “MyRadar” Design Case Study - Visit 2

#### 2.10.1. Problem Statement

Modify the design introduced in Chapter 1 for the “MyRadar” design case study so that the effects of target RCS fluctuations are taken into account. For this purpose modify the design such that: The aircraft and missile target types follow Swerling I and Swerling III fluctuations, respectively. Also assume that a $P_D \geq 0.995$ is required at maximum range with $P_{fa} = 10^{-7}$ or better. You may use either non-coherent integration or cumulative probability of detection. Also, modify any other design parameters if needed.

#### 2.10.2. A Design

The missile and the aircraft detection ranges were calculated in Chapter 1. They are $R_a = 90Km$ for the aircraft and $R_m = 55Km$ for the missile. First, determine the probability of detection for each target type with and without the 7-pulse non-coherent integration. For this purpose, use MATLAB program “myradar_visit2_1.m” given in Listing 2.27. This program first computes the improvement factor and the associated integration loss. Second it calculates the single pulse SNR. Finally it calculates the SNR when non-coherent integration is utilized. Executing this program yields:

- $SNR_{single\_pulse\_missile} = 5.5998\ dB$
- $SNR_{7\_pulse\_NCI\_missile} = 11.7216\ dB$
- $SNR_{single\_pulse\_aircraft} = 6.0755\ dB$
- $SNR_{7\_pulse\_NCI\_aircraft} = 12.1973\ dB$

Using these values in functions “pd_swerling1.m” and “pd_swerling3.m” yields

- $Pd_{single\_pulse\_missile} = 0.013$
- $Pd_{7\_pulse\_NCI\_missile} = 0.9276$
- $Pd_{single\_pulse\_aircraft} = 0.038$
- $Pd_{7\_pulse\_NCI\_aircraft} = 0.8273$

Clearly in all four cases, there is not enough SNR to meet the design requirement of $P_D \geq 0.995$.

---

1. Please read disclaimer in Section 1.9.1.
Instead, resort to accomplishing the desired probability of detection by using cumulative probabilities. The single frame increment for the missile and aircraft cases are

\[
\begin{align*}
R_{\text{Missile}} &= \text{scan rate} \times v_m = 2 \times 150 = 300m \\
R_{\text{Aircraft}} &= \text{scan rate} \times v_a = 2 \times 400 = 800m
\end{align*}
\]

(2.111) (2.112)

2.10.2.1 Single Pulse (Per Frame) Design Option

As a first design option, consider the case where during each frame only a single pulse is used for detection (i.e., no integration). Consequently, if the single pulse detection does not achieve the desired probability of detection at 90 Km for the aircraft or at 55 Km for the missile, then non-coherent integration of a few pulses per frame can then be utilized. Keep in mind that only non-coherent integration can be used in the cases of Swerling type I and III fluctuations (see Section 2.4).

Assume that the first frame corresponding to detecting the aircraft is 106 Km. This assumption is arbitrary and it provides the designer with 21 frames. It follows that the first frame, when detecting the missile, is at 61 Km. Furthermore, assume that the SNR at \( R = 90Km \) is \((SNR)_{\text{aircraft}} = 8.5dB\), for the aircraft case. And, for the missile case assume that at \( R = 55Km \) the corresponding SNR is \((SNR)_{\text{missile}} = 9dB\). Note that these values are simply educated guesses, and the designer may be required to perform several iterations in order to accomplish the desired cumulative probability of detection, \( P_D \geq 0.995 \). In order to calculate the cumulative probability of detection at a certain range, the MATLAB program “myradar_visit2_2.m” was developed. This program is given in Listing 2.28 in Section 2.11.

Initialization of the program “myradar_visit2_2.m” includes entering the following inputs: The desired \( P_{fa} \); the number of pulses to be used for non-coherent integration per frame; the range at which the desired cumulative operability of detection must be achieved; the frame size; and finally the target fluctuation type. For notational purposes, denote the range at which the desired cumulative probability of detection must be achieved as \( R_0 \). Then for each frame, the following list includes the outputs of this program: SNR, probability of detection, fluctuation loss, and cumulative probability of detection.

The logic used by this program for calculating the proper probability of detection at each frame and for computing the cumulative probability of detection is described as follows:

1. Initialize the program, by entering the desired input values. Assume Swerling V fluctuation and use Eq. (2.98) to calculate the frame-SNR, \((SNR)_i\).
1.1. For the “MyRadar” design case study, use $n_P = 1$, $R_0 = 90\text{Km}$, and $(SNR_0)_{\text{aircraft}} = 8.5\text{dB}$. Alternatively use $R_0 = 55\text{Km}$ and $(SNR)_{\text{missile}} = 9\text{dB}$ for the missile case. Note that the selected SNR values are best estimates or educated guesses, and it may require going through few iterations before finally selecting an acceptable set.

2. The program will then calculate the number of frames and their associated ranges. The program uses the function “fluct_loss.m” to calculate the Swerling V $P_D$ at each frame and the additional SNR required to accomplish the same probability of detection when target fluctuation is included.

3. Depending on the fluctuation type, the program will then use the proper MATLAB function to calculate the probability of detection for each frame, $P_{D_i}$.

3.1. For the “MyRadar” design case study, these functions are “pd_swerling1.m” and “pd_swerling 3.m”.

4. Finally, the program uses Eq. (2.99) to compute the cumulative probability of detection, $P_{D_C}$.

A Graphical User Interface (GUI) has been developed for this program; Fig. 2.20 shows its associated GUI workspace. To use this GUI, from the MATLAB command window type “myradar_visit2_2_gui”. Executing the program “myradar_visit2_2.m” using the input values stated above yields the following cumulative probabilities of detection for the aircraft and missile cases,

- $P_{D_C,\text{Missile}} = 0.99872$
- $P_{D_C,\text{Aircraft}} = 0.99687$

These results clearly satisfy the design requirement of $P_D \geq 0.995$. However, one must re-validate the peak power requirement for the design. To do that, go back to Eqs. (1.107) and (1.108), and replace the SNR values used in Chapter 1 by the values adopted in this chapter (i.e., $(SNR_0)_{\text{aircraft}} = 8.5\text{dB}$ and $(SNR)_{\text{missile}} = 9\text{dB}$). It follows that the corresponding single pulse energy for the missile and the aircraft cases are respectively given by

$$E_m = 0.1658 \times \frac{10^{0.9}}{10^{0.56}} = 0.36273\text{Joules} \quad (2.113)$$

$$E_a = 0.1487 \times \frac{10^{0.85}}{10^{0.56}} = 0.28994\text{Joules} \quad (2.114)$$
This indicates that the stressing single pulse peak power requirement (i.e., missile detection) exceeds $362\, KW$. This value for the single pulse peak power is high for a mobile ground based air defense radar and practical constraints would require using less peak power.

In order to bring the single pulse peak power requirement down, one can use non-coherent integration of a few pulses per frame prior to calculating the frame probability of detection. For this purpose, the program "myradar_visit2_2.m" can be used again. However, in this case $n_p > 1$. This is analyzed in the next section.

### 2.10.2.2. Non-Coherent Integration Design Option

The single frame probability of detection can be improved significantly when pulse integration is utilized. One may use coherent or non-coherent integration to improve the frame cumulative probability of detection. In this case, caution should be exercised since coherent integration would not be practical

---

**Figure 2.20. GUI workspace associated with program “myradar_visit2_2_gui.m”**

This indicates that the stressing single pulse peak power requirement (i.e., missile detection) exceeds $362\, KW$. This value for the single pulse peak power is high for a mobile ground based air defense radar and practical constraints would require using less peak power.

In order to bring the single pulse peak power requirement down, one can use non-coherent integration of a few pulses per frame prior to calculating the frame probability of detection. For this purpose, the program “myradar_visit2_2.m” can be used again. However, in this case $n_p > 1$. This is analyzed in the next section.

### 2.10.2.2. Non-Coherent Integration Design Option

The single frame probability of detection can be improved significantly when pulse integration is utilized. One may use coherent or non-coherent integration to improve the frame cumulative probability of detection. In this case, caution should be exercised since coherent integration would not be practical.
when the target fluctuation type is either Swerling I or Swerling III. Alternatively, using non-coherent integration will always reduce the minimum required SNR.

Rerun the MATLAB program “myradar_visit2_2_gui”. Use \( n_p = 4 \) and use \( SNR = 4dB \) (single pulse) for both the missile and aircraft single pulse SNR\(^1\) at their respective reference ranges, \( R_{0_{\text{missile}}} = 55Km \) and \( R_{0_{\text{aircraft}}} = 90Km \). The resulting cumulative probabilities of detection are

\[
P_{DC_{\text{missile}}} = 0.99945
\]
\[
P_{DC_{\text{aircraft}}} = 0.99812
\]

which are both within the desired design requirements. It follows that the corresponding minimum required single pulse energy for the missile and the aircraft cases are now given by

\[
E_m = 0.1658 \times \frac{10^{0.4}}{10^{0.56}} = 0.1147\text{Joules} \tag{2.115}
\]
\[
E_d = 0.1487 \times \frac{10^{0.4}}{10^{0.56}} = 0.1029\text{Joules} \tag{2.116}
\]

Thus, the minimum single pulse peak power (assuming the same pulsewidth as that given in Section1.9.2) is

\[
P_t = \frac{0.1147}{1 \times 10^{-6}} = 114.7KW \tag{2.117}
\]

Note that the peak power requirement will be significantly reduced while maintaining a very fine range resolution when pulse compression techniques are used. This will be discussed in a subsequent chapter.

Fig. 2.21 shows a plot of the SNR versus range for both target types. This plot assumes 4-pulse non-coherent integration. It can be reproduced using MATLAB program “fig2_21.m”. It is given in Listing 2.29 in Section 2.11.

### 2.11. MATLAB Program and Function Listings

This section presents listings for all MATLAB programs/functions used in this chapter. The user is advised to rerun these programs with different input parameters.

---

1. Again these values are educated guesses. The designer may be required to go through a few iterations before arriving at an acceptable set of design parameters.
% This program can be used to reproduce Figure 2.2 of the text
clear all
xg = linspace(-6,6,1500); % random variable between -6 and 6
xr = linspace(0,6,1500); % random variable between 0 and 6
mu = 0; % zero mean Gaussian pdf mean
sigma = 1.5; % standard deviation (sqrt(variance))
ynorm = normpdf(xg,mu,sigma); % use MATLAB function normpdf
yray = raylpdf(xr,sigma); % use MATLAB function raylpdf
plot(xg,ynorm,'k',xr,yray,'k-.'); % use MATLAB function raylpdf
grid
legend('Gaussian pdf','Rayleigh pdf')
xlabel('Detection range - Km')

Figure 2.21. SNR versus detection range for both target types. The 4-pulse NCI curves correspond to 21 frame cumulative detection with the last frame at: 55 Km for the missile and 90 Km for the aircraft.
Listing 2.2. MATLAB Function “que_func.m”

function fofx = que_func(x)
% This function computes the value of the Q-function
% listed in Eq.(2.16). It uses the approximation in Eqs. (2.17) and (2.18)
if (x >= 0)
    denom = 0.661 * x + 0.339 * sqrt(x^2 + 5.51);
    expo = exp(-x^2 /2.0);
    fofx = 1.0 - (1.0 / sqrt(2.0 * pi)) * (1.0 / denom) * expo;
else
    denom = 0.661 * x + 0.339 * sqrt(x^2 + 5.51);
    expo = exp(-x^2 /2.0);
    value = 1.0 - (1.0 / sqrt(2.0 * pi)) * (1.0 / denom) * expo;
    fofx = 1.0 - value;
end

Listing 2.3. MATLAB Program “fig2_3.m”

% This program generates Figure 2.3.
close all
clear all
logpfa = linspace(.01,250,1000);
var = 10.^((logpfa ./ 10.0);
vtlnorm = sqrt( log (var));
semilogx(logpfa, vtnorm,'k')
grid

Listing 2.4. MATLAB Function “marcumsq.m”

function Pd = marcumsq (a,b)
% This function uses Parl's method to compute PD
max_test_value = 5000.;
if (a < b)
    alphan0 = 1.0;
    dn = a / b;
else
    alphan0 = 0.;
    dn = b / a;
end
alphan_1 = 0.;
betan0 = 0.5;
betan\_1 = 0.;
D1 = dn;
n = 0;
ratio = 2.0 / (a * b);
r1 = 0.0;
betan = 0.0;
alphan = 0.0;
while betan < 1000.,
    n = n + 1;
    alphan = dn + ratio * n * alphan0 + alphan;
    betan = 1.0 + ratio * n * betan0 + betan;
    alphan\_1 = alphan0;
    alphan0 = alphan;
    betan\_1 = betan0;
    betan0 = betan;
    dn = dn * D1;
end
PD = (alphan0 / (2.0 * betan0)) * exp( -(a-b)^2 / 2.0);
if ( a >= b)
    PD = 1.0 - PD;
end
return

---

**Listing 2.5. MATLAB Program “prob_snr1.m”**

% This program is used to produce Fig. 2.4
close all
clear all
for nfa = 2:2:12
    b = sqrt(-2.0 * log(10^(-nfa)));
    index = 0;
    hold on
    for snr = 0:.1:18
        index = index +1;
        a = sqrt(2.0 * 10^(.1*snr));
        pro(index) = marcumsq(a,b);
    end
    x = 0:.1:18;
    set(gca,'ytick',[.1 .2 .3 .4 .5 .6 .7 .75 .8 .85 .9 ... .95 .9999])
    set(gca,'xtick',[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18])
    loglog(x, pro,'k');
end
hold off
xlabel ('Single pulse SNR - dB')
ylabel ('Probability of detection')
grid

---

**Listing 2.6. MATLAB program “fig2_6a.m”**

```matlab
% This program is used to produce Fig. 2.6a
% It uses the function "improv_fac"
clear all
close all
pfa1 = 1.0e-2;
pfa2 = 1.0e-6;
pfa3 = 1.0e-10;
pfa4 = 1.0e-13;
pd1 = .5;
pd2 = .8;
pd3 = .95;
pd4 = .999;
index = 0;
for np = 1:1:1000
    index = index + 1;
    I1(index) = improv_fac (np, pfa1, pd1);
    I2(index) = improv_fac (np, pfa2, pd2);
    I3(index) = improv_fac (np, pfa3, pd3);
    I4(index) = improv_fac (np, pfa4, pd4);
end
np = 1:1:1000;
semilogx (np, I1, 'k', np, I2, 'k--', np, I3, 'k-.', np, I4, 'k:');
xlabel ('Number of pulses');
ylabel ('Improvement factor I - dB')
legend ('pd=.5, nfa=e+2','pd=.8, nfa=e+6','pd=.95, nfa=e+10','pd=.999, nfa=e+13');
grid
```

---

**Listing 2.7. MATLAB Function “improv_fac.m”**

```matlab
function impr_of_np = improv_fac (np, pfa, pd)
% This function computes the non-coherent integration improvement
% factor using the empirical formula defined in Eq. (2.49)
fact1 = 1.0 + log10( 1.0 / pfa) / 46.6;
fact2 = 6.79 * (1.0 + 0.235 * pd);
fact3 = 1.0 - 0.14 * log10(np) + 0.0183 * (log10(np))^2;
```
impr_of_np = fact1 * fact2 * fact3 * log10(np);
return

Listing 2.8. MATLAB Program “fig2_6b.m”

% This program is used to produce Fig. 2.6b
% It uses the function "improv_fac".
clear all
close all
pfa1 = 1.0e-12;
pfa2 = 1.0e-12;
pfa3 = 1.0e-12;
pfa4 = 1.0e-12;
pd1 = .5;
pd2 = .8;
pd3 = .95;
pd4 = .99;
index = 0;
for np = 1:1:1000
    index = index+1;
    I1 = improv_fac (np, pfa1, pd1);
    i1 = 10.^(0.1*I1);
    L1(index) = -1*10*log10(i1 ./ np);
    I2 = improv_fac (np, pfa2, pd2);
    i2 = 10.^(0.1*I2);
    L2(index) = -1*10*log10(i2 ./ np);
    I3 = improv_fac (np, pfa3, pd3);
    i3 = 10.^(0.1*I3);
    L3(index) = -1*10*log10(i3 ./ np);
    I4 = improv_fac (np, pfa4, pd4);
    i4 = 10.^(0.1*I4);
    L4 (index) = -1*10*log10(i4 ./ np);
end
np = 1:1:1000;
semilogx (np, L1, 'k', np, L2, 'k--', np, L3, 'k-.', np, L4, 'k:')
axis tight
xlabel ('Number of pulses');
ylabel ('Integration loss - dB')
legend ('pd=.5, nfa=e+12','pd=.8, nfa=e+12','pd=.95, nfa=e+12','pd=.99, nfa=e+12');
grid
Listing 2.9. MATLAB Function “incomplete_gamma.m”

function [value] = incomplete_gamma ( vt, np)
% This function implements Eq. (2.67) to compute the Incomplete Gamma Function
% This function needs "factor.m" to run
format long
eps = 1.000000001;
% Test to see if np = 1
if (np == 1)
    value1 = vt * exp(-vt);
    value = 1.0 - exp(-vt);
    return
end
sumold = 1.0;
sumnew = 1.0;
calc1 = 1.0;
calc2 = np;
xx = np * log(vt+0.0000000001) - vt - factor(calc2);
temp1 = exp(xx);
temp2 = np / (vt+0.0000000001);
diff = .0;
ratio = 1000.0;
if (vt >= np)
    while (ratio >= eps)
        diff = diff + 1.0;
        calc1 = calc1 * (calc2 - diff) / vt;
        sumnew = sumold + calc1;
        ratio = sumnew / sumold;
        sumold = sumnew;
    end
    value = 1.0 - temp1 * sumnew * temp2;
    return
else
    diff = 0.;
    sumold = 1.;
    ratio = 1000.;
    calc1 = 1.;
    while(ratio >= eps)
        diff = diff + 1.0;
        calc1 = calc1 * vt / (calc2 + diff);
        sumnew = sumold + calc1;
        ratio = sumnew / sumold;
        sumold = sumnew;
    end
end
end

Listing 2.10. MATLAB Function “factor.m”

function [val] = factor(n)
% Compute the factorial of n using logarithms to avoid overflow.
format long
n = n + 9.0;
n2 = n * n;
temp = (n-1) * log(n) - n + log(sqrt(2.0 * pi * n)) ... 
+ ((1.0 - (1.0/30. + (1.0/105)/n2)/n2) / 12) / n;
val = temp - log((n-1)*(n-2)*(n-3)*(n-4)*(n-5)*(n-6) ... 
*(n-7)*(n-8));
return

Listing 2.11. MATLAB Program “fig2_7.m”

% This program can be used to reproduce Fig. 2.7
close all
clear all
format long
ii = 0;
for x = 0:.1:20
    ii = ii+1;
    val1(ii) = incomplete_gamma(x, 1);
    val2(ii) = incomplete_gamma(x, 3);
    val = incomplete_gamma(x, 6);
    val3(ii) = val;
    val = incomplete_gamma(x, 10);
    val4(ii) = val;
end
xx = 0:.1:20;
plot(xx,val1,'k',xx,val2,'k:',xx,val3,'k--',xx,val4,'k-.')
legend(’N = 1’,’N = 3’,’N = 6’,’N = 10’) 
xlabel(’x’) 
ylabel(’Incomplete Gamma function (x,N)’) 
grid

Listing 2.12. MATLAB Function “threshold.m”

function [pfa, vt] = threshold (nfa, np)
% This function calculates the threshold value from nfa and np.
% The Newton-Raphson recursive formula is used (Eqs. (2-63) through (2-66))
% This function uses "incomplete_gamma.m".
delmax = .00001;
eps = 0.000000001;
delta = 10000.;
pfa = np * log(2) / nfa;
sqrtctx = sqrt(-log10(pfa));
sqrtctx = sqrt(np);
vt0 = np - sqrtctx + 2.3 * sqrtctx * (sqrtctx + sqrtctx - 1.0);
vt = vt0;
while (abs(delta) >= vt0)
  igf = incomplete_gamma(vt0, np);
  num = 0.5^(np/nfa) - igf;
  temp = (np-1) * log(vt0+eps) - vt0 - factor(np-1);
  deno = exp(temp);
  vt = vt0 + (num / (deno+eps));
  delta = abs(vt - vt0) * 10000.0;
  vt0 = vt;
end

Listing 2.13. MATLAB Program “fig2_8.m”

% Use this program to reproduce Fig. 2.8 of text
clear all
for n= 1: 1:150
  [pfa1 y1(n)] = threshold(1000, n);
  [pfa2 y3(n)] = threshold(10000, n);
  [pfa3 y4(n)] = threshold(500000, n);
end
n = 1:1:150;
loglog(n,y1,'k',n,y3,'k--',n,y4,'k-.');
axis([0 200 1 300])
xlabel('Number of pulses');
ylabel('Threshold');
legend('nfa=1000','nfa=10000','nfa=500000')
grid

Listing 2.14. MATLAB Function “pd_swerling5.m”

function pd = pd_swerling5 (input1, indicator, np, snrbar)
% This function is used to calculate the probability of
% for Swerling 5 or 0 targets for np>1.
if(np == 1)
  'Stop, np must be greater than 1'
  return
format long
snrb = 10.0.^(snrbar./10.);
eps = 0.00000001;
delmax = .00001;
delta =10000.;
% Calculate the threshold Vt
if (indicator ~=1)
    nfa = input1;
    pfa = np * log(2) / nfa;
else
    pfa = input1;
    nfa = np * log(2) / pfa;
end
sqrtpfa = sqrt(-log10(pfa));
sqrtnp = sqrt(np);
vt0 = np - sqrtnp + 2.3 * sqrtpfa * (sqrtpfa + sqrtnp - 1.0);
vt = vt0;
while (abs(delta) >= vt0)
    igf = incomplete_gamma(vt0,np);
    num = 0.5^(np/nfa) - igf;
    temp = (np-1) * log(vt0+eps) - vt0 - factor(np-1);
    deno = exp(temp);
    vt = vt0 + (num / (deno+eps));
    delta = abs(vt - vt0) * 10000.0;
    vt0 = vt;
end
% Calculate the Gram-Charlier coefficients
temp1 = 2.0 .* snrbar + 1.0;
omegabar = sqrt(np .* temp1);
c3 = -(snrbar + 1.0 / 3.0) ./ (sqrt(np) .* temp1.^1.5);
c4 = (snrbar + 0.25) ./ (np .* temp1.^2.);
c6 = c3 .* c3 ./2.0;
V = (vt - np .* (1.0 + snrbar)) ./ omegabar;
Vsq = V .*V;
val1 = exp(-Vsq ./ 2.0) ./ sqrt( 2.0 * pi);
val2 = c3 .* (V.^2 -1.0) + c4 .* V .* (3.0 - V.^2) - ... 
c6 .* V .* (V.^4 - 10. .* V.^2 + 15.0);
q = 0.5 .* erfc (V/sqrt(2.0));
pd = q - val1 .* val2;
Listing 2.15. MATLAB Program “fig2_9.m”

% This program is used to produce Fig. 2.9
close all
clear all
pfa = 1e-9;
nfa = log(2) / pfa;
b = sqrt(-2.0 * log(pfa));
index = 0;
for snr = 0:.1:20
    index = index + 1;
    a = sqrt(2.0 * 10^(.1*snr));
    pro(index) = marcumsq(a,b);
    prob205(index) = pd_swerling5 (pfa, 1, 10, snr);
end
x = 0:.1:20;
plot(x, pro,'k',x,prob205,'k:');
axis([0 20 0 1])
xlabel ('SNR - dB')
ylabel ('Probability of detection')
legend('np = 1','np = 10')
grid

Listing 2.16. MATLAB Function “pd_swerling1.m”

function pd = pd_swerling1 (nfa, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 1 targets.
format long
snrbar = 10.0^(snrbar/10.);
eps = 0.000000001;
delmax = .00001;
delta = 10000.;
% Calculate the threshold Vt
pfa = np * log(2) / nfa;
sqrtgamma = sqrt(-log10(pfa));
sqrtnp = sqrt(np);
vto = np - sqrtnp + 2.3 * sqrtgamma * (sqrtgamma + sqrtnp - 1.0);
vt = vto;
while (abs(delta) >= vto)
    igf = incomplete_gamma(vto,np);
    num = 0.5^(np/nfa) - igf;
    temp = (np-1) * log(vto+eps) - vto - factor(np-1);
    deno = exp(temp);
    delta = 0.5 / deno;
end
v = vto - delta;
p = pd_swerling1(v, np, snrbar);

end
\[ vt = vt0 + \frac{\text{num}}{\text{deno} + \text{eps}}; \]
\[ \text{delta} = \left| vt - vt0 \right| \times 10000.0; \]
\[ vt0 = vt; \]
end

if (np == 1)
    \[ \text{temp} = -\frac{vt}{1.0 + \text{snrbar}}; \]
    \[ \text{pd} = \exp(\text{temp}); \]
    return
end

\[ \text{temp1} = 1.0 + \text{np} \times \text{snrbar}; \]
\[ \text{temp2} = \frac{1.0}{(\text{np} \times \text{snrbar})}; \]
\[ \text{temp} = 1.0 + \text{temp2}; \]
\[ \text{val1} = \text{temp}^{\text{np} - 1}; \]
\[ \text{igf1} = \text{incomplete\_gamma}(vt,\text{np} - 1); \]
\[ \text{igf2} = \text{incomplete\_gamma}(vt/\text{temp},\text{np} - 1); \]
\[ \text{pd} = 1.0 - \text{igf1} + \text{val1} \times \text{igf2} \times \exp(-vt/\text{temp1}); \]
Listing 2.18. MATLAB Program “fig2_11ab.m”

% This program is used to produce Fig. 2.11a&b

clear all
pfa = 1e-11;
nfa = log(2) / pfa;
index = 0;
for snr = -10:.5:30
    index = index +1;
    prob1(index) = pd_swerling1(nfa, 1, snr);
    prob10(index) = pd_swerling1(nfa, 10, snr);
    prob50(index) = pd_swerling1(nfa, 50, snr);
    prob100(index) = pd_swerling1(nfa, 100, snr);
end

x = -10:.5:30;
plot(x, prob1,'k',x,prob10,'k:',x,prob50,'k--', ...
     x, prob100,'k-');
axis([-10 30 0 1])
xlabel ('SNR - dB')
ylabel ('Probability of detection')
legend('np = 1','np = 10','np = 50','np = 100')
grid

Listing 2.19. MATLAB Function “pd_swerling2.m”

function pd = pd_swerling2(nfa, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 2 targets.

format long

snrbar = 10.0^(snrbar/10.);
eps = 0.00000001;
delmax = .00001;
delta = 10000.;

% Calculate the threshold Vt

pfa = np * log(2) / nfa;
srtpfa = sqrt(-log10(pfa));
srtnp = sqrt(np);

vt0 = np - srtnp + 2.3 * srtpfa * (srtpfa + srtnp - 1.0);

vt = vt0;
while (abs(delta) >= vt0)
    igf = incomplete_gamma(vt0,np);
    num = 0.5^(np/nfa) - igf;
    temp = (np-1) * log(vt0+eps) - vt0 - factor(np-1);
    deno = exp(temp);
    vt = vt0 + (num / (deno+eps));
end
delta = abs(vt - vt0) * 10000.0;
v0 = vt;
end
if (np <= 50)
    temp = vt / (1.0 + snrbar);
pd = 1.0 - incomplete_gammad(temp,np);
    return
else
    temp1 = snrbar + 1.0;
    omegabar = sqrt(np) * temp1;
c3 = -1.0 / sqrt(9.0 * np);
c4 = 0.25 / np;
c6 = c3 * c3 /2.0;
V = (vt - np * temp1) / omegabar;
Vsqrt = V *V;
val1 = exp(-Vsqrt / 2.0) / sqrt( 2.0 * pi);
val2 = c3 * (V^2 -1.0) + c4 * V * (3.0 - V^2) - ...
c6 * V * (V^4 - 10. * V^2 + 15.0);
q = 0.5 * erfc (V/sqrt(2.0));
pd = q - val1 * val2;
end

Listing 2.20. MATLAB Program “fig2_12.m”
% This program is used to produce Fig. 2.12
clear all
pfa = 1e-10;
nfa = log(2) / pfa;
index = 0;
for snr = -10:.5:30
    index = index +1;
    prob1(index) = pd_swerling2 (nfa, 1, snr);
    prob10(index) = pd_swerling2 (nfa, 10, snr);
    prob50(index) = pd_swerling2 (nfa, 50, snr);
    prob100(index) = pd_swerling2 (nfa, 100, snr);
end
x = -10:.5:30;
plot(x, prob1, 'k',x,prob10,'k:',x,prob50,'k--', ...
x, prob100, 'k-.');
axis([-10 30 0 1])
xlabel ('SNR - dB')
ylabel ('Probability of detection')
legend('np = 1','np = 10','np = 50','np = 100')
grid
Listing 2.21. MATLAB Function “pd_swerling3.m”

function pd = pd_swerling3 (nfa, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 3 targets.
format long
snrbar = 10.0^(snrbar/10.);
eps = 0.00000001;
delmax = .00001;
delta = 10000.;
% Calculate the threshold Vt
pfa = np * log(2) / nfa;
sqrtpfa = sqrt(-log10(pfa));
sqrtnp = sqrt(np);
vt0 = np - sqrtnp + 2.3 * sqrtpfa * (sqrtpfa + sqrtnp - 1.0);
vt = vt0;
while (abs(delta) >= vt0)
    igf = incomplete_gamma(vt0,np);
    num = 0.5^(np/nfa) - igf;
    temp = (np-1) * log(vt0+eps) - vt0 - factor(np-1);
    deno = exp(temp);
    vt = vt0 + (num / (deno+eps));
    delta = abs(vt - vt0) * 10000.0;
    vt0 = vt;
end

Listing 2.22. MATLAB Program “fig2_13.m”

% This program is used to produce Fig. 2.13
clear all
pfa = 1e-9;
nfa = log(2) / pfa;
index = 0;
for snr = -10:.5:30
   index = index +1;
   prob1(index) = pd_swerling3(nfa, 1, snr);
   prob10(index) = pd_swerling3(nfa, 10, snr);
   prob50(index) = pd_swerling3(nfa, 50, snr);
   prob100(index) = pd_swerling3(nfa, 100, snr);
end
x = -10:.5:30;
plot(x, prob1,'k',x,prob10,'k:',x,prob50,'k--', ...
    x, prob100,'k-.');
axis([-10 30 0 1])
xlabel ('SNR - dB')
ylabel ('Probability of detection')
legend('np = 1','np = 10','np = 50','np = 100')
grid

Listing 2.23. MATLAB Function “pd_swerling4.m”

function pd = pd_swerling4 (nfa, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 4 targets.
format long
snrbar = 10.0^(snrbar/10.);
eps = 0.00000001;
delmax = .00001;
delta =10000.;
% Calculate the threshold Vt
pfa = np * log(2) / nfa;
sqrtpfa = sqrt(-log10(pfa));
sqrtnp = sqrt(np);
vt0 = np - sqrtnp + 2.3 * sqrtpfa * (sqrtpfa + sqrtnp - 1.0);
vt = vt0;
while (abs(delta) >= vt0)
   igf = incomplete_gamma(vt0,np);
   num = 0.5^(np/nfa) - igf;
   temp = (np-1) * log(vt0+eps) - vt0 - factor(np-1);
   deno = exp(temp);
   vt = vt0 + (num / (deno+eps));
   delta = abs(vt - vt0) * 10000.0;
   vt0 = vt;
end
h8 = snrbar /2.0;
\[\beta = 1.0 + h8;\]
\[\beta2 = 2.0 \times \beta^2 - 1.0;\]
\[\beta3 = 2.0 \times \beta^3;\]

if (np >= 50)
\[\text{temp1} = 2.0 \times \beta - 1;\]
\[\text{omegabar} = \sqrt{\text{np} \times \text{temp1}};\]
\[c3 = (\beta3 - 1.) / 3.0 / \beta2 / \text{omegabar};\]
\[c4 = (\beta3 \times \beta3 - 1.0) / 4. / \text{np} / \beta2 / \beta2;\]
\[c6 = c3 \times c3 / 2.0;\]
\[V = (\text{vt} - \text{np} \times (1.0 + \text{snrbar})) / \text{omegabar};\]
\[V\text{sqr} = V \times V;\]
\[\text{val1} = \exp(-V\text{sqr} / 2.0) / \sqrt{2.0 \times \pi};\]
\[\text{val2} = c3 \times (V^2 - 1.0) + c4 \times V \times (3.0 - V^2) - ...\]
\[c6 \times V \times (V^4 - 10. \times V^2 + 15.0);\]
\[q = 0.5 \times \text{erfc}(V / \sqrt{2.0});\]
\[pd = q - \text{val1} \times \text{val2};\]
return

else
\[\text{snr} = 1.0;\]
\[\text{gamma0} = \text{incomplete}_\gamma(\text{vt} / \beta, \text{np});\]
\[a1 = (\text{vt} / \beta)^n / (\exp(\text{factor}(\text{np})) \times \exp(\text{vt} / \beta));\]
\[\text{sum} = \text{gamma0};\]
for i = 1:1:np
\[\text{temp1} = 1;\]
if (i == 1)
\[\text{ai} = a1;\]
else
\[\text{ai} = (\text{vt} / \beta) \times a1 / (\text{np} + i - 1);\]
end
\[a1 = ai;\]
\[\text{gammai} = \text{gamma0} - ai;\]
\[\text{gamma0} = \text{gammai};\]
\[a1 = ai;\]
for ii = 1:1:i
\[\text{temp1} = \text{temp1} \times (\text{np} + 1 - ii);\]
end
\[\text{term} = (\text{snrbar} / 2.0)^i \times \text{gammai} \times \text{temp1} / \exp(\text{factor}(i));\]
\[\text{sum} = \text{sum} + \text{term};\]
end
\[pd = 1.0 - \text{sum} / \beta^n;\]
end
\[pd = \text{max}(pd, 0.);\]
Listing 2.24. MATLAB Program “fig2_14.m”

% This program is used to produce Fig. 2.14

clear all
pfa = 1e-9;
afa = log(2) / pfa;
index = 0;
for snr = -10:.5:30
    index = index +1;
    prob1(index) = pd_swerling4(afa, 1, snr);
    prob10(index) = pd_swerling4(afa, 10, snr);
    prob50(index) = pd_swerling4(afa, 50, snr);
    prob100(index) = pd_swerling4(afa, 100, snr);
end
x = -10:.5:30;
plot(x, prob1,'k',x,prob10,'k:',x,prob50,'k--', ...
    x, prob100,'k-.');
axis([-10 30 0 1.1])
xlabel ('SNR - dB')
ylabel ('Probability of detection')
legend('np = 1','np = 10','np = 50','np = 100')
grid
axis tight

Listing 2.25. MATLAB Function “fluct_loss.m”

function [Lf, Pd_Sw5] = fluct_loss(pd, pfa, np, sw_case)
% This function calculates the SNR fluctuation loss for Swerling models
% A negative Lf value indicates SNR gain instead of loss

format long
% compute the false alarm number
afa = log(2) / pfa;
% *************** Swerling 5 case ***************
% check to make sure that np>1
if (np ==1)
    b = sqrt(-2.0 * log(pfa));
    Pd_Sw5 = 0.001;
    snr_inc = 0.1 - 0.005;
    while(Pd_Sw5 <= pd)
        snr_inc = snr_inc + 0.005;
        a = sqrt(2.0 * 10^(.1*snr_inc));
        Pd_Sw5 = marcumsq(a,b);
    end
PD_SW5 = Pd_Sw5

© 2004 by Chapman & Hall/CRC CRC Press LLC
SNR_SW5 = snr_inc
else
    % np > 1 use MATLAB function pd_swerling5.m
    snr_inc = 0.1 - 0.005;
    Pd_Sw5 = 0.001;
    while(Pd_Sw5 <= pd)
        snr_inc = snr_inc + 0.005;
        Pd_Sw5 = pd_swerling5(pfa, 1, np, snr_inc);
    end
    PD_SW5 = Pd_Sw5
    SNR_SW5 = snr_inc
end
if sw_case == 5
    Lf = 0.
    return
end
% ****************** End Swerling 5 case ***************
% ****************** Swerling 1 case *******************
if (sw_case == 1)
    Pd_Sw1 = 0.001;
    snr_inc = 0.1 - 0.005;
    while(Pd_Sw1 <= pd)
        snr_inc = snr_inc + 0.005;
        Pd_Sw1 = pd_swerling1(nfa, np, snr_inc);
    end
    PD_SW1 = Pd_Sw1
    SNR_SW1 = snr_inc
    Lf = SNR_SW1 - SNR_SW5
end
% ****************** End Swerling 1 case ***************
% ****************** Swerling 2 case *******************
if (sw_case == 2)
    Pd_Sw2 = 0.001;
    snr_inc = 0.1 - 0.005;
    while(Pd_Sw2 <= pd)
        snr_inc = snr_inc + 0.005;
        Pd_Sw2 = pd_swerling2(nfa, np, snr_inc);
    end
    PD_SW2 = Pd_Sw2
    SNR_SW2 = snr_inc
    Lf = SNR_SW2 - SNR_SW5
end
% ****************** End Swerling 2 case ***************
% ****************** Swerling 3 case *******************
if (sw_case == 3)
    Pd_Sw3 = 0.001;
    snr_inc = 0.1 - 0.005;
    while(Pd_Sw3 <= pd)
        snr_inc = snr_inc + 0.005;
        Pd_Sw3 = pd_swerling3(nfa, np, snr_inc);
    end
    PD_SW3 = Pd_Sw3
    SNR_SW3 = snr_inc
    Lf = SNR_SW3 - SNR_SW5
end
% ******************* End Swerling 3 case ***************
% ******************* Swerling 4 case *******************
if (sw_case == 4)
    Pd_Sw4 = 0.001;
    snr_inc = 0.1 - 0.005;
    while(Pd_Sw4 <= pd)
        snr_inc = snr_inc + 0.005;
        Pd_Sw4 = pd_swerling4(nfa, np, snr_inc);
    end
    PD_SW4 = Pd_Sw4
    SNR_SW4 = snr_inc
    Lf = SNR_SW4 - SNR_SW5
end
% ******************* End Swerling 4 case ***************
return

Listing 2.26. MATLAB Program “fig2_15.m”

% Use this program to reproduce Fig. 2.15 of text
clear all
index = 0.;
for pd = 0.01:.05:1
    index = index + 1;
    [Lf,Pd_Sw5] = fluct_loss(pd, 1e-9,1,1);
    Lf1(index) = Lf;
    [Lf,Pd_Sw5] = fluct_loss(pd, 1e-9,1,4);
    Lf4(index) = Lf;
end
pd = 0.01:.05:1;
figure (2)
plot(pd, Lf1, 'k',pd, Lf4,'K:')
xlabel('Probability of detection')
ylabel('Fluctuation loss - dB')
legend('Swerling I & II','Swerling III & IV')
title('Pfa=1e-9, np=1')
grid

---

**Listing 2.27. MATLAB Program “myradar_visit2_1.m”**

```matlab
% Myradar design case study visit 2_1
close all
clear all
pfa = 1e-7;
pd = 0.995;
np = 7;
pt = 165.8e3; % peak power in Watts
freq = 3e+9; % radar operating frequency in Hz
g = 34.5139; % antenna gain in dB
sigmam = 0.5; % missile RCS m squared
sigmaa = 4; % aircraft RCS m squared
te = 290.0; % effective noise temperature in Kelvins
b = 1.0e+6; % radar operating bandwidth in Hz
nf = 6.0; % noise figure in dB
loss = 8.0; % radar losses in dB
% compute the improvement factor due to 7-pulse non-coherent integration
Improv = improv_fac(np, pfa, pd);
% calculate the integration loss
lossnci = 10*log10(np) - Improv;
% calculate net gain in SNR due to integration
SNR_net = Improv - lossnci;
loss_total = loss + lossnci;
rangem = 55e3;
rangea = 90e3;
SNR_single_pulse_missile = radar_eq(pt, freq, g, sigmam, te, b, nf, loss, rangem)
SNR_7_pulse_NCI_missile = SNR_single_pulse_missile + SNR_net
SNR_single_pulse_aircraft = radar_eq(pt, freq, g, sigmaa, te, b, nf, loss, rangea)
SNR_7_pulse_NCI_aircraft = SNR_single_pulse_aircraft + SNR_net
```

---

**Listing 2.28. MATLAB Program “myradar_visit2_2.m”**

```matlab
%clear all
% close all
% swid = 3;
```
% pfa = 1e-7;
% np = 1;
% R_1st_frame = 61e3; % Range for first frame
% R0 = 55e3; % range to last frame
% SNR0 = 9; % SNR at R0
% frame = 0.3e3; % frame size
nfa = log(2) / pfa;
range_frame = R_1st_frame:-frame:R0; % Range to each frame
% implement Eq. (2.98)
SNRI = SNR0 + 40 .* log10((R0 ./ range_frame));
% calculate the Swerling 5 Pd at each frame
b = sqrt(-2.0 * log(pfa));
if np == 1
    for frame = 1:1:size(SNRI,2)
        a = sqrt(2.0 * 10^(.1*SNRI(frame)));
        pd5(frame) = marcumsq(a,b);
    end
else
    [pd5] = pd_swerling5(pfa, 1, np, SNRI);
end
% compute additional SNR needed due to fluctuation
for frame = 1:1:size(SNRI,2)
    [Lf(frame),Pd_Sw5] = fluct_loss(pd5(frame), pfa, np, swid);
end
% adjust SNR at each frame
SNRI = SNRI - Lf;
% compute the frame Pd
for frame = 1:1:size(SNRI,2)
    if(swid==1)
        Pdi(frame) = pd_swerling1(nfa, np, SNRI(frame));
    end
    if(swid==2)
        Pdi(frame) = pd_swerling2(nfa, np, SNRI(frame));
    end
    if(swid==3)
        Pdi(frame) = pd_swerling3(nfa, np, SNRI(frame));
    end
    if(swid==4)
        Pdi(frame) = pd_swerling4(nfa, np, SNRI(frame));
    end
    if(swid==5)
        Pdi(frame) = pd5(frame);
    end
end
\[ P_{dc}(1: \text{size}(\text{SNR}_i, 2)) = 0; \]
\[ P_{dc}(1) = 1 - P_{di}(1); \]
\% compute the cumulative \( P_d \)
\for\ frame = 2:1: \text{size}(\text{SNR}_i, 2)
    \[ P_{dc}(\text{frame}) = (1 - P_{di}(\text{frame})) \times P_{dc}(\text{frame}-1); \]
\end
\[ P_{DC} = 1 - P_{dc}(21) \]

**Listing 2.29. MATLAB Program “fig2_21.m”**

% Use this program to reproduce Fig. 2.20 of text.
close all
clear all
np = 4;
pfa = 1e-7;
pdm = 0.99945;
pda = 0.99812;
% calculate the improvement factor
\[ I_m = \text{improv\_fac}(\text{np}, \text{pfa}, \text{pdm}); \]
\[ I_a = \text{improv\_fac}(\text{np}, \text{pfa}, \text{pda}); \]
% calculate the integration loss
\[ L_m = 10 \times \log(10(\text{np})) - I_m; \]
\[ L_a = 10 \times \log(10(\text{np})) - I_a; \]
pt = 114.7e3; % peak power in Watts
freq = 3e+9; % radar operating frequency in Hz
g = 34.5139; % antenna gain in dB
sigmam = 0.5; % missile RCS m squared
sigmaa = 4; % aircraft RCS m squared
te = 290.0; % effective noise temperature in Kelvins
b = 1.0e+6; % radar operating bandwidth in Hz
nf = 6.0; % noise figure in dB
loss = 8.0; % radar losses in dB
losstm = loss + Lm; % total loss for missile
lossta = loss + La; % total loss for aircraft
range = linspace(20e3,120e3,1000); % range to target from 20 to 120 Km, 1000 points
% modify pt by np*pt to account for pulse integration
snrmnci = radar\_eq(np*pt, freq, g, sigmam, te, b, nf, losstm, range);
snrm = radar\_eq(pt, freq, g, sigmam, te, b, nf, loss, range);
snrancl = radar\_eq(np*pt, freq, g, sigmaa, te, b, nf, lossta, range);
snra = radar\_eq(pt, freq, g, sigmaa, te, b, nf, loss, range);
% plot SNR versus range
rangekm = range ./ 1000;
figure(1)
subplot(2,1,1)
plot(rangekm,snrmnci,'k',rangekm,snrm,'k - .')
grid
legend('With 4-pulse NCI','Single pulse')
ylabel ('SNR - dB');
title('Missile case')
subplot(2,1,2)
plot(rangekm,snranci,'k',rangekm,snra,'k - .')
grid
legend('With 4-pulse NCI','Single pulse')
ylabel ('SNR - dB');
title('Aircraft case')
xlabel('Detection range - Km')