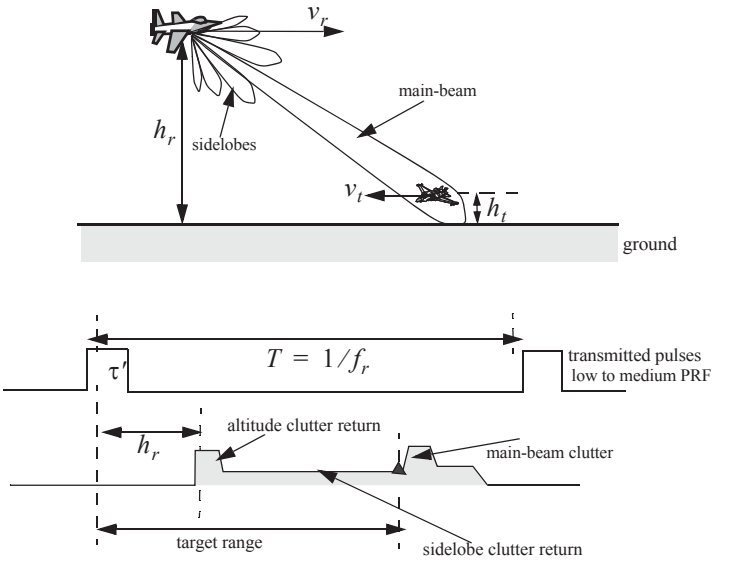


Figure 10.15. Pulse radar detection of ground targets with clutter interference.

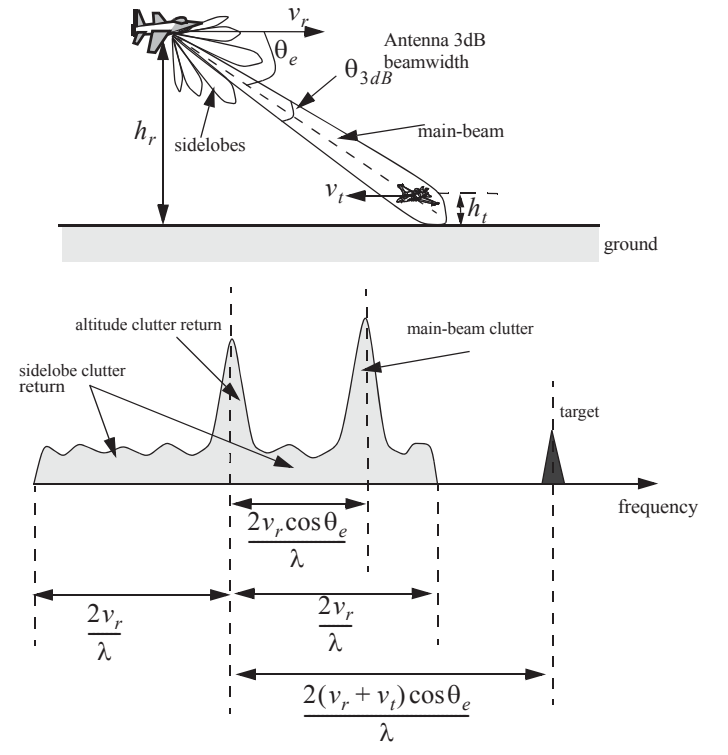


Figure 10.16. Illustration of frequency characteristics of pulse Doppler radar echoes.

10.7.1. Pulse Doppler Radar Signal Processing

The main idea behind pulse Doppler radar signal processing is to divide the footprint (the intersection of the antenna 3*dB* beamwidth with the ground) into resolution cells that constitute a range Doppler map, MAP. The sides of this map are range and Doppler, as illustrated in Fig. 10.17. Fine range resolution, ΔR, is accomplished in real time by utilizing range gating and pulse compression. Frequency (Doppler) resolution is obtained from the coherent processing interval.

To further illustrate this concept, consider the case where N_a is the number of azimuth (Doppler) cells, and N_r is the number of range bins. Hence, the MAP is of size $N_a \times N_r$, where the columns refer to range bins and the rows refer to azimuth cells. For each transmitted pulse within the dwell, the echoes from consecutive range bins are recorded sequentially in the first row of MAP. Once the first row is completely filled (i.e., returns from all range bins have been received), all data (in all rows) are shifted downward one row before the next pulse is transmitted. Thus, one row of MAP is generated for every transmitted pulse. Consequently, for the current observation interval, returns from the first transmitted pulse will be located in the bottom row of MAP, and returns from the last transmitted pulse will be in the top row of MAP.

Fine range resolution is achieved using the matched filter. Clutter rejection (filtering) is performed on each range bin (i.e., rows in the MAP). Then all samples from one dwell within each range bin are processed using an FFT to resolve targets in Doppler. It follows that a peak in a given resolution cell corresponds to a specific target detection at that range and Doppler frequency. Selection of the proper size FFT and its associated parameters were discussed in Chapter 3.

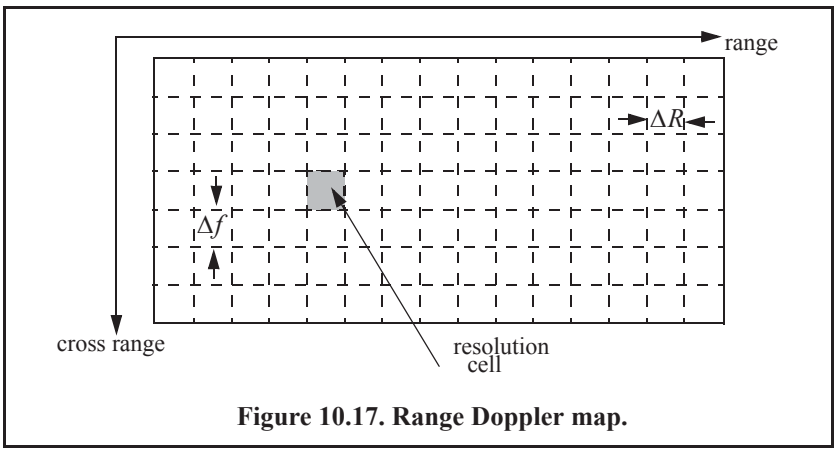


Figure 10.17. Range Doppler map.

10.7.2. Resolving Range Ambiguities

Pulse Doppler radars exhibit serve range ambiguities because they use high PRF pulse streams. In order to resolve these ambiguities, pulse Doppler radars utilize multiple high PRFs (PRF staggering) within each processing interval (dwell). For this purpose, consider a pulse Doppler radar that uses two PRFs, f_{r1} and f_{r2}, on transmit to resolve range ambiguity, as shown in Fig. 10.18. Denote R_{u1} and R_{u2} as the unambiguous ranges for the two PRFs,

respectively. Normally, these unambiguous ranges are relatively small and are short of the desired radar unambiguous range R_u (where $R_u \gg R_{u1}, R_{u2}$). Denote the radar desired PRF that corresponds to R_u as f_{rd}.

The choice of f_{r1} and f_{r2} is such that they are relatively prime with respect to one another. One choice is to select $f_{r1} = Nf_{rd}$ and $f_{r2} = (N+1)f_{rd}$ for some integer N. Within one period of the desired PRI ($T_d = 1/f_{rd}$), the two PRFs f_{r1} and f_{r2} coincide only at one location, which is the true unambiguous target position. The time delay T_d establishes the desired unambiguous range. The time delays t_1 and t_2 correspond to the time between the transmit of a pulse on each PRF and receipt of a target return due to the same pulse.

Let M_1 be the number of PRF1 intervals between transmit of a pulse and receipt of the true target return. The quantity M_2 is similar to M_1 except it is for PRF2. It follows that over the interval 0 to T_d, the only possible results are $M_1 = M_2 = M$ or $M_1 + 1 = M_2$. The radar needs only to measure t_1 and t_2. First, consider the case when $t_1 < t_2$. In this case,

$$t_1 + \frac{M}{f_{r1}} = t_2 + \frac{M}{f_{r2}} \qquad \textbf{Eq. (10.74)}$$

for which we get

$$M = \frac{t_2 - t_1}{T_1 - T_2} \qquad \textbf{Eq. 10.75)}$$

where $T_1 = 1/f_{r1}$ and $T_2 = 1/f_{r2}$. It follows that the round-trip time to the true target location is

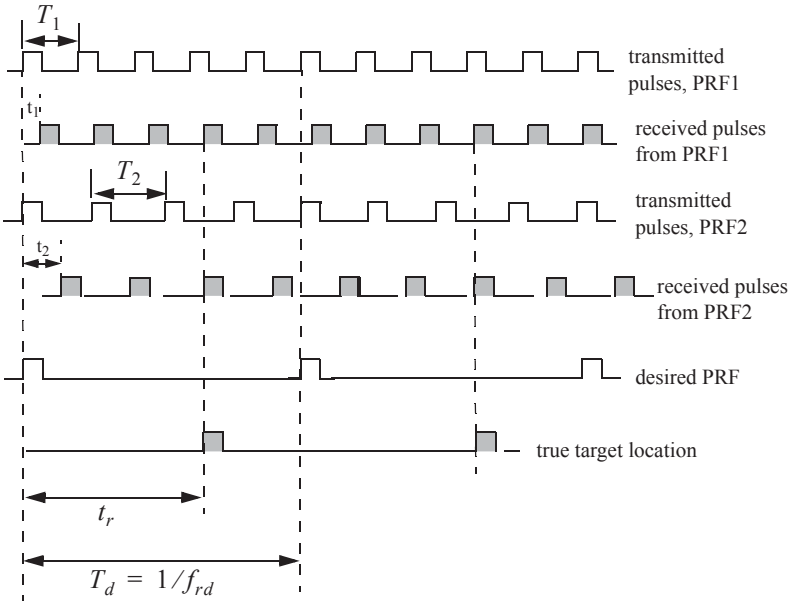


Figure 10.18. Resolving range ambiguity.

$$t_r = MT_1 + t_1$$
$$t_r = MT_2 + t_2 \qquad \textbf{Eq. (10.76)}$$

and the true target range is

$$R = ct_r/2\,. \qquad \textbf{Eq. (10.77)}$$

Now, if $t_1 > t_2$, then

$$t_1 + \frac{M}{f_{r1}} = t_2 + \frac{M+1}{f_{r2}}\,. \qquad \textbf{Eq. (10.78)}$$

Solving for M we get

$$M = \frac{(t_2 - t_1) + T_2}{T_1 - T_2} \qquad \textbf{Eq. (10.79)}$$

and the round-trip time to the true target location is

$$t_{r1} = MT_1 + t_1\,, \qquad \textbf{Eq. (10.80)}$$

and in this case, the true target range is

$$R = \frac{ct_{r1}}{2}\,. \qquad \textbf{Eq. (10.81)}$$

Finally, if $t_1 = t_2$, then the target is in the first ambiguity. It follows that

$$t_{r2} = t_1 = t_2 \qquad \textbf{Eq. (10.82)}$$

and

$$R = ct_{r2}/2 \qquad \textbf{Eq. (10.83)}$$

Since a pulse cannot be received while the following pulse is being transmitted, these times correspond to blind ranges. This problem can be resolved by using a third PRF. In this case, once an integer N is selected, then in order to guarantee that the three PRFs are relatively prime with respect to one another, we may choose $f_{r1} = N(N+1)f_{rd}$, $f_{r2} = N(N+2)f_{rd}$, and $f_{r3} = (N+1)(N+2)f_{rd}$.

10.7.3. Resolving Doppler Ambiguity

In the case where the pulse Doppler radar is utilizing medium PRFs, it will be ambiguous in both range and Doppler. Resolving range ambiguities was discussed in the previous section. In this section, Doppler ambiguity is addressed. Remember that the line spectrum of a train of pulses has $\sin x/x$ envelope, and the line spectra are separated by the PRF, f_r, as illustrated in Fig. 10.19. The Doppler filter bank is capable of resolving target Doppler as long as the anticipated Doppler shift is less than one half the bandwidth of the individual filters (i.e., one half the width of an FFT bin). Thus, pulsed radars are designed such that

$$f_r = 2f_{dmax} = (2v_{rmax})/\lambda \qquad \textbf{Eq. (10.84)}$$

where f_{dmax} is the maximum anticipated target Doppler frequency, v_{rmax} is the maximum anticipated target radial velocity, and λ is the radar wavelength.

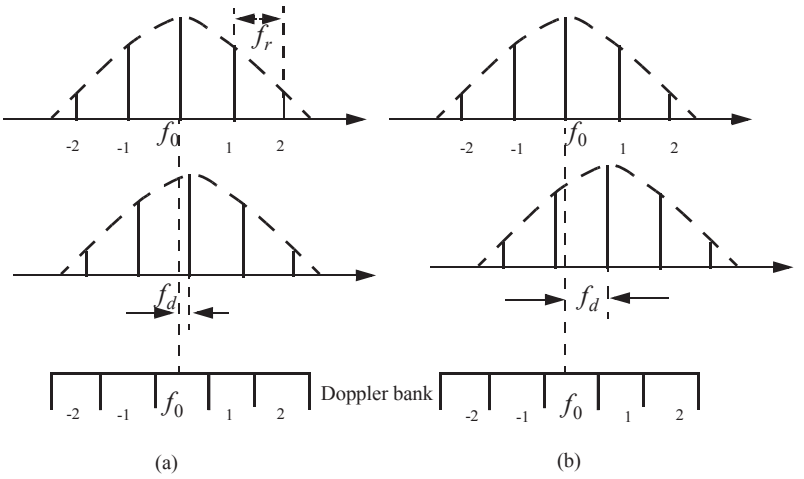


Figure 10.19. Spectra of transmitted and received waveforms, and Doppler bank. (a) Doppler is resolved. (b) Spectral lines have moved into the next Doppler filter. This results in an ambiguous Doppler measurement.

If the Doppler frequency of the target is high enough to make an adjacent spectral line move inside the Doppler band of interest, the radar can be Doppler ambiguous. Therefore, in order to avoid Doppler ambiguities, radar systems require high PRF rates when detecting high-speed targets. When a long-range radar is required to detect a high-speed target, it may not be possible to be both range and Doppler unambiguous. This problem can be resolved by using multiple PRFs. Multiple PRF schemes can be incorporated sequentially within each dwell interval (scan or integration frame) or the radar can use a single PRF in one scan and resolve ambiguity in the next. The latter technique, however, may have problems due to changing target dynamics from one scan to the next.

The Doppler ambiguity problem is analogous to that of range ambiguity. Therefore, the same methodology can be used to resolve Doppler ambiguity. In this case, we measure the Doppler frequencies f_{d1} and f_{d2} instead of t_1 and t_2. If $f_{d1} > f_{d2}$, then we have

$$M = \frac{(f_{d2} - f_{d1}) + f_{r2}}{f_{r1} - f_{r2}}. \qquad \textbf{Eq. (10.85)}$$

And if $f_{d1} < f_{d2}$,

$$M = \frac{f_{d2} - f_{d1}}{f_{r1} - f_{r2}} \qquad \textbf{Eq. (10.86)}$$

and the true Doppler is

$$f_d = Mf_{r1} + f_{d1} \quad ; f_d = Mf_{r2} + f_{d2}. \qquad \textbf{Eq. (10.87)}$$

Finally, if $f_{d1} = f_{d2}$, then

$$f_d = f_{d1} = f_{d2}. \qquad \textbf{Eq. (10.88)}$$

Again, blind Dopplers can occur, which can be resolved using a third PRF.

Example:

A certain radar uses two PRFs to resolve range ambiguities. The desired unambiguous range is $R_u = 100Km$. Choose $N = 59$. Compute f_{r1}, f_{r2}, R_{u1}, and R_{u2}.

Solution:

First let us compute the desired PRF, f_{rd}

$$f_{rd} = \frac{c}{2R_u} = \frac{3 \times 10^8}{200 \times 10^3} = 1.5KHz.$$

It follows that

$$f_{r1} = Nf_{rd} = (59)(1500) = 88.5KHz$$

$$f_{r2} = (N+1)f_{rd} = (59+1)(1500) = 90KHz$$

$$R_{u1} = \frac{c}{2f_{r1}} = \frac{3 \times 10^8}{2 \times 88.5 \times 10^3} = 1.695Km$$

$$R_{u2} = \frac{c}{2f_{r2}} = \frac{3 \times 10^8}{2 \times 90 \times 10^3} = 1.667Km.$$

Example:

Consider a radar with three PRFs; $f_{r1} = 15KHz$, $f_{r2} = 18KHz$, and $f_{r3} = 21KHz$. Assume $f_0 = 9GHz$. Calculate the frequency position of each PRF for a target whose velocity is $550m/s$. Calculate f_d (Doppler frequency) for another target appearing at $8KHz$, $2KHz$, and $17KHz$ for each PRF.

Solution:

The Doppler frequency is

$$f_d = 2\frac{vf_0}{c} = \frac{2 \times 550 \times 9 \times 10^9}{3 \times 10^8} = 33KHz.$$

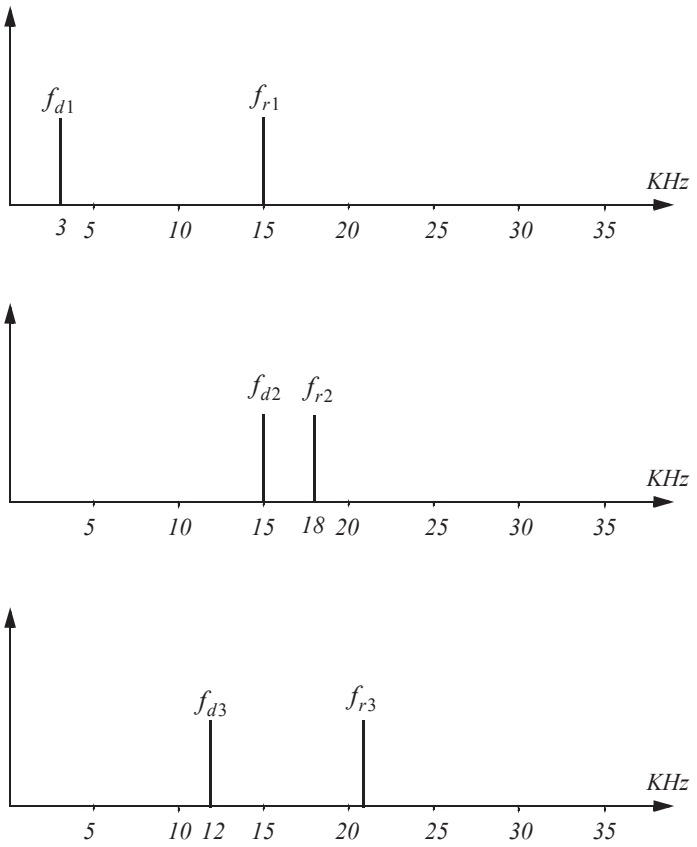
Then by using Eq. (10.87) $n_i f_{ri} + f_{di} = f_d$ where $i = 1, 2, 3$, we can write

$$n_1 f_{r1} + f_{d1} = 15n_1 + f_{d1} = 33$$

$$n_2 f_{r2} + f_{d2} = 18n_2 + f_{d2} = 33$$

$$n_3 f_{r3} + f_{d3} = 21n_3 + f_{d3} = 33.$$

We will show here how to compute n_1, and leave the computations of n_2 and n_3 to the reader. First, if we choose $n_1 = 0$, that means $f_{d1} = 33KHz$, which cannot be true since f_{d1} cannot be greater than f_{r1}. Choosing $n_1 = 1$ is also invalid since $f_{d1} = 18KHz$ cannot be true either. Finally, if we choose $n_1 = 2$, we get $f_{d1} = 3KHz$, which is an acceptable value. It follows that the minimum n_1, n_2, n_3 that may satisfy the above three relations are $n_1 = 2$, $n_2 = 1$, and $n_3 = 1$. Thus, the apparent Doppler frequencies are $f_{d1} = 3KHz$, $f_{d2} = 15KHz$, and $f_{d3} = 12KHz$, as seen below.



Now for the second part of the problem. Again, by using Eq. (10.87) we have

$$n_1 f_{r1} + f_{d1} = f_d = 15n_1 + 8$$

$$n_2 f_{r2} + f_{d2} = f_d = 18n_2 + 2$$

$$n_3 f_{r3} + f_{d3} = f_d = 21n_3 + 17\,.$$

We can now solve for the smallest integers n_1, n_2, n_3 that satisfy the above three relations. See the table below. Thus, $n_1 = 2 = n_2$, and $n_3 = 1$, and the true target Doppler is $f_d = 38KHz$. It follows that

$$v_r = 38000 \times \frac{0.0333}{2} = 632.7\frac{m}{\text{sec}}\,.$$

n	**0**	**1**	**2**	**3**	**4**
f_d from f_{r1}	8	23	<u>38</u>	53	68
f_d from f_{r2}	2	20	<u>38</u>	56	
f_d from f_{r3}	17	<u>38</u>	39		

10.8. Phase Noise

It was determined in earlier chapters that the radar performance is improved as the SNR becomes larger, and in Chapter 2, an expression of the SNR against thermal noise was derived and analyzed. It was also established that in the presence of clutter, the radar performance will degrade beyond the impact of thermal noise alone, and in this case, the SCR becomes more critical than the SNR. Another source of noise that greatly limits MTI and pulsed Doppler radar performance is known as phase noise. Phase noise, sometimes called flicker noise, is random in nature and is caused by instabilities within the radar's local oscillator.

Phase noise may limit, depending on its actual value, pulsed Doppler radars' ability to detect very slow moving targets whose RCS is relatively small. This is illustrated in Fig. 10.20. In this case, when the slow-moving target return signal is close to zero (or multiple integers of the PRF) it will likely be masked by the phase noise power caused by a noisy radar local oscillator. In addition to the masking problem illustrated in Fig. 10.20, the MTI improvement factor will also be reduced by some appreciable values corresponding to the amount of phase noise present in the radar receiver, as will be explained later in this section.

Simply put, phase noise is a term used to describe the random frequency perturbation (relatively small in nature) occurring around the signal carrier frequency, thus causing a new instantaneous signal frequency that is slightly different form the original value. For example, consider the signal defined by the simple sinusoid

$$x(t) = r(t)\sin(2\pi f_c t) \qquad \textbf{Eq. (10.89)}$$

where $r(t)$ is the amplitude modulation and f_c is the carrier frequency. The perturbed signal due to amplitude and phase instabilities of the local oscillator will take on the form

$$x(t) = r(t + a(t))\sin(2\pi f_c t + \phi(t)) \qquad \textbf{Eq. (10.90)}$$

where $a(t)$ is the amplitude perturbation and $\phi(t)$ is the phase perturbation or fluctuation. Recalling that the instantaneous frequency is the derivative of phase with respect to time divided by 2π, then the phase perturbation will change the center frequency from f_c to $f_c + \delta f$ where δf is a fractional frequency deviation away from the carrier. More specifically,

$$\delta f = \frac{1}{2\pi}\frac{d}{dt}\phi(t)\,. \qquad \textbf{Eq. (10.91)}$$

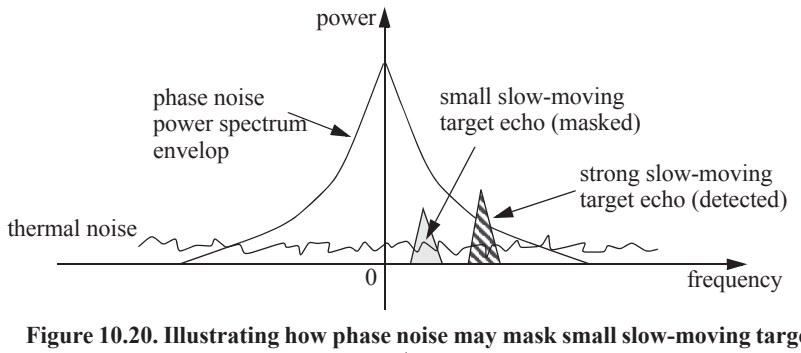


Figure 10.20. Illustrating how phase noise may mask small slow-moving target returns.

The notation commonly used in the literature to describe the phase noise power spectrum density is

$$S_{\phi}(f) = 2\mathcal{L}(f). \quad \textbf{Eq. (10.92)}$$

The factor of 2 accounts for both sidebands of the spectrum (lower and upper sidebands around f_c) and $\mathcal{L}(f)$ is the ratio of noise power in a $1Hz$ bandwidth at an offset from the carrier signal power measured in dBc/Hz. In this notation, $\mathcal{L}(f)$ is often used to denote phase noise. For example, consider a frequency-modulated signal as described in Eq. (2.117b) in Chapter 2, which is repeated here for convenience as Eq. (10.93),

$$x(t) = A\left\{J_0(\beta)\cos 2\pi f_c t + \right. \quad \textbf{Eq. (10.93)}$$

$$\left[\sum_{n=2(even)}^{\infty} J_n(\beta)[\cos(2\pi f_c + 2n\pi f_m)t + \cos(2\pi f_c - 2n\pi f_m)t]\right] +$$

$$\left.\left[\sum_{q=1(odd)}^{\infty} J_q(\beta)[\cos(2\pi f_c + 2q\pi f_m)t - \cos(2\pi f_c - 2q\pi f_m)t]\right]\right\}$$

where β is the modulation index and A is the amplitude. In this case, phase noise is defined as the ratio of the sideband power to the carrier power at a certain modulation frequency offset from the carrier. More specifically,

$$\mathcal{L}(f)\big|_{dBc/Hz} = 10 \times \log\left\{\frac{|J_1(\beta)|^2}{|J_0(\beta)|^2}\right\}. \quad \textbf{Eq. (10.94)}$$

In general, the phase noise $\mathcal{L}(f)$ decreases with frequency as a function of $(1/f^3)$, $(1/f^2)$, and $(1/f)$. Figure 10.21 shows an illustration plot for $\mathcal{L}(f)$ versus the *log* of the frequency. Typically, the manufacturer of a given oscillator will measure and publish the phase noise characteristics (plots similar to Fig. 10.21) as part of their product documentation. Observation of Fig. 10.21 shows that this plot is a piece-wise linear function. It follows that the formula for a given segment of this plot is given by

$$\mathcal{L}(f; f_{i+1} - f_i)\big|_{dBc/Hz} = m_i \log(f) - m_i \log(f_i) + \mathcal{L}(f_i) \quad \textbf{Eq. (10.95)}$$

where m_i is the slope of the ith segment defined by

$$m_i = \frac{\mathcal{L}(f_{i+1}) - \mathcal{L}(f_i)}{\log(f_{i+1}) - \log(f_i)}. \quad \textbf{Eq. (10.96)}$$

Eq. (10.95) can be written in a more compact form as

$$\mathcal{L}(f) = \mathcal{L}(f_i) \times 10^{\left[\frac{m_i}{10}\log\left(\frac{f}{f_i}\right)\right]}. \quad \textbf{Eq. (10.97)}$$

Figure 10.22 shows an actual plot for $\mathcal{L}(f)$ using the following values:

$$\{\mathcal{L}(f_1), \mathcal{L}(f_2), \mathcal{L}(f_3), \mathcal{L}(f_4), \mathcal{L}(f_5)\} = \{-55, -85, -105, -115, -115\}.$$

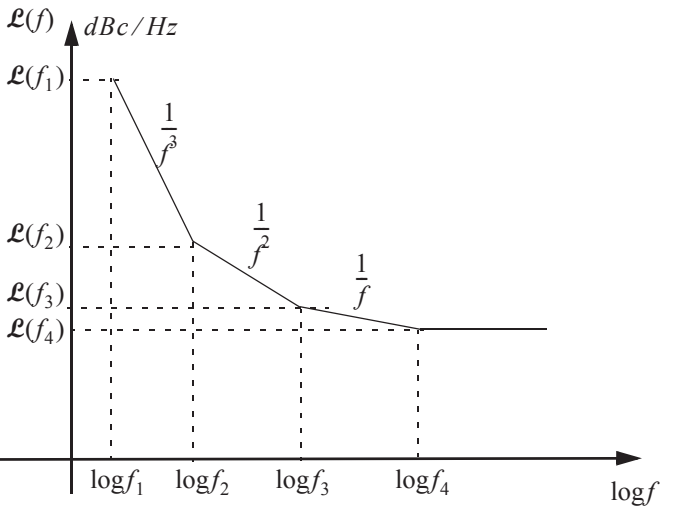


Figure 10.21. Typical graph of phase noise PSD versus frequency.

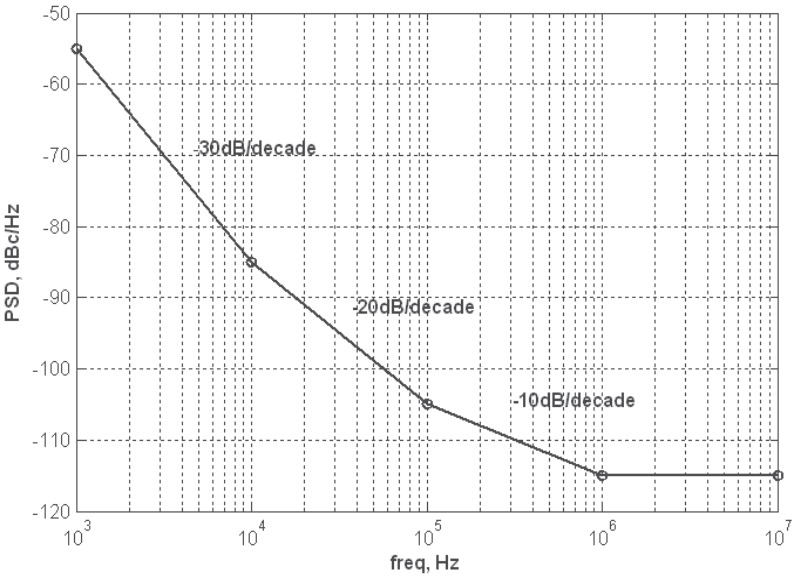


Figure 10.22. Example of $\mathcal{L}(f)$ PSD versus frequency.

The literature is flooded with sources on phase noise. Different users use slightly different formulas to express phase noise in their particular application. In this book, the following formula for phase noise is adopted,

$$\mathcal{L}(f) = \frac{1}{\pi} \frac{c_p \pi f_c^2}{(f - c_p \pi f_c)^2 + (c_p \pi f_c^2)^2} \qquad \textbf{Eq. (10.98)}$$

where f_c is the carrier frequency and c_p is a constant that describes phase noise of the oscillator. Figure 10.23 shows some typical plots of the ratio of noise power to the carrier power (as defined in Eq. (10.98)). This figure can be reproduced using MATLAB Program *"Fig10_23.m,"* listed in Appendix 10-A. Note that when the constant c_p becomes larger, the noise ratio spectrum becomes wider with lower amplitude of the main beam, and hence less phase noise in the system.

The normalized phase noise power spectrum density can be computed using Eqs. (10.92) and (10.97), and can be approximated as

$$\mathcal{L}(f) = \begin{cases} \dfrac{1}{\pi f_b} & ;f \text{ approaches } f_c \\ \dfrac{1}{2\pi f_b} & ;f = f_c + f_b \\ \dfrac{f_b}{\pi f^2} & ;f \gg f_c + f_b \end{cases} \qquad \textbf{Eq. (10.99)}$$

where $f_b = c_p \pi f_c^2$. Figure 10.24 shows the corresponding normalized phase noise power spectrum density versus frequency. This figure can be reproduced using MATLAB program *"Fig10_24.m,"* listed in Appendix 10-A.

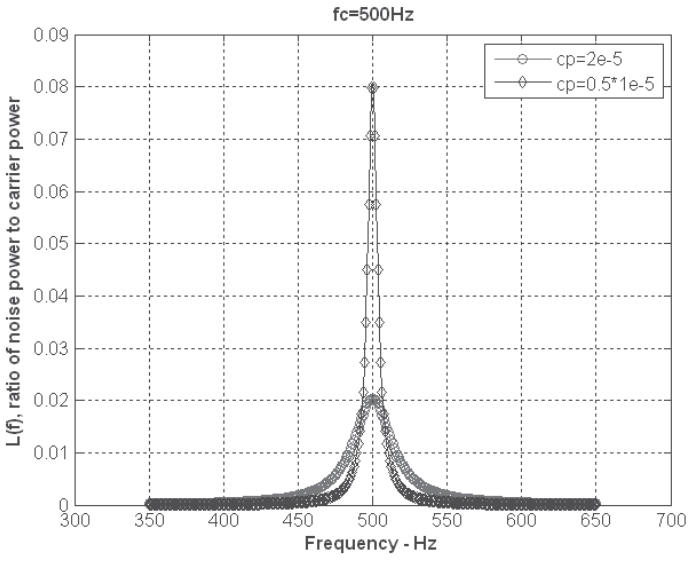


Figure 10.23. $\mathcal{L}(f)$ ratio of noise power to carrier power versus frequency.

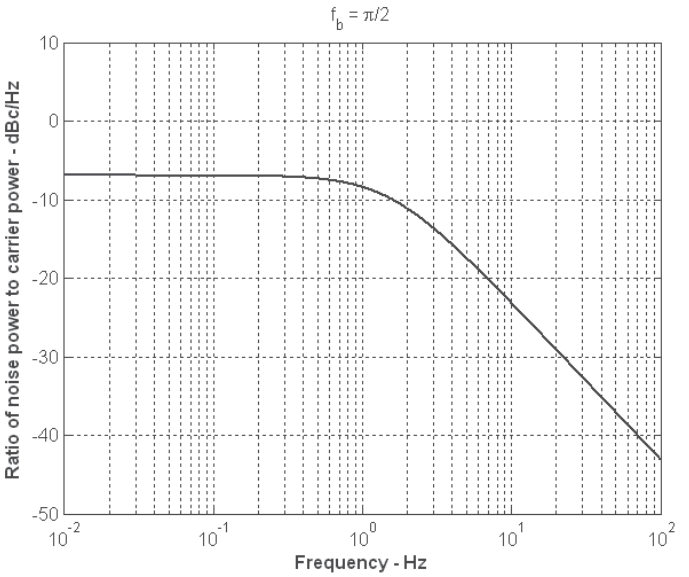


Figure 10.24. Normalized phase noise power spectrum density.

As indicated by Fig. 10.24, quiet oscillators will almost always have phase noise of less than $0 dBc/Hz$ at frequency offsets of more than $1 Hz$ away from the carrier. However, at some small frequency bandwidth of less than $1 Hz$, phase noise maybe greater than $0 dBc/Hz$.

The power spectral density function of phase noise at the output of the radar's matched filter can be expressed as

$$S_\phi(f) = \mathcal{L}_0 P_c \left(\frac{\sin(\pi f \tau)}{\pi f \tau} \right)^2 \qquad \textbf{Eq. (10.100)}$$

where $\mathcal{L}_0$ is the phase noise ratio relative to the carrier, P_c was defined in Eq. (10.2) as the clutter power, and τ is the radar pulsewidth. $\mathcal{L}_0$ can be computed from the analysis presented earlier; however, an acceptable range for $\mathcal{L}_0$ varies between 10^{-9} to 10^{-15} *dBc/Hz*. The clutter attenuation was defined in Eq. (10.41) as

$$CA = C_i / C_o \qquad \textbf{Eq. (10.101)}$$

where C_i is equal to P_c (see the derivation of Eq. (10.47)).

Ignoring the phase noise, the clutter power spectrum was given in Eq. (10.44), but when phase noise is taken into consideration, Eq. (10.44) is modified to

$$S_t(f) = S(f) + S_\phi(f) = \frac{P_c}{\sqrt{2\pi}\ \sigma_f} \exp(-f^2 / 2\sigma_f^2) + \mathcal{L}_0 P_c \left(\frac{\sin(\pi f \tau)}{\pi f \tau} \right)^2 \qquad \textbf{Eq. (10.102)}$$

and the clutter output of the MTI filter is computed in a manner similar to that described in Section 10.4, except in this case $S(f)$ is replaced by $S_t(f)$ in Eq. (10.102). Performing the integration and collecting terms (assuming a 2-pulse MTI filter), yields

$$C_o = P_c \frac{1}{2}\left(\frac{2\pi\sigma_f}{f_r}\right)^2 + P_c \frac{\mathcal{L}_0}{\tau} \qquad \text{Eq. (10.103)}$$

where f_r is the PRF. It follows that the clutter attenuation in the presence of phase noise is given by

$$CA = \frac{C_i}{C_o} = \frac{1}{\frac{1}{2}\left(\frac{2\pi\sigma_f}{f_r}\right)^2 + \frac{\mathcal{L}_0}{\tau}}. \qquad \text{Eq. (10.104)}$$

Figure 10.25 shows the clutter attenuation for two values of σ_f, with $f_r = 2.5KHz$ and $\tau = 1\mu s$ using Eq. (10.104). Observation of Fig. 10.26 leads to the following conclusions: Larger values of σ_f will result in less clutter attenuation, so if more clutter attenuation is desired then a 3-pulse or higher order MTI filter ought to be used. Next, phase noise does not start to affect the performance of the MTI filter until it becomes higher than -100*dBc/Hz*. Clearly, using higher-order MTI filters will increase the amount of clutter attenuation; however, the question that remain, is how does phase noise affect the MTI performance when higher-order filters are used? This analysis is left as an exercise (see Problem 10.20). Figure 10.25 can be reproduced using MATLAB program *"Fig10_25.m,"* listed in Appendix 10-A.

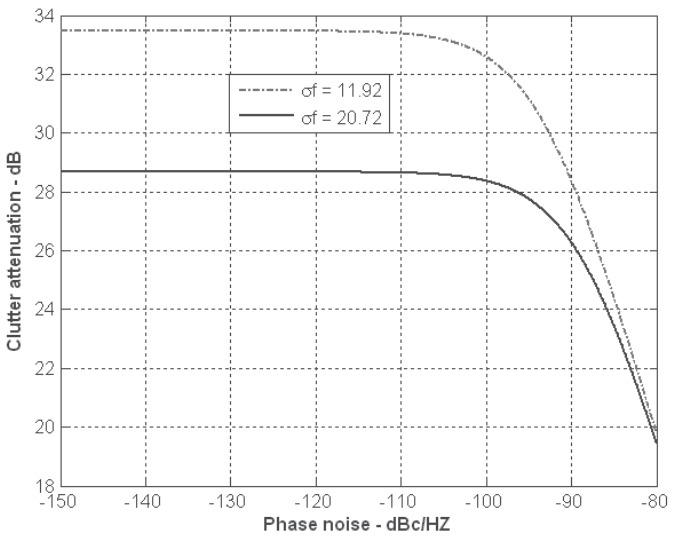
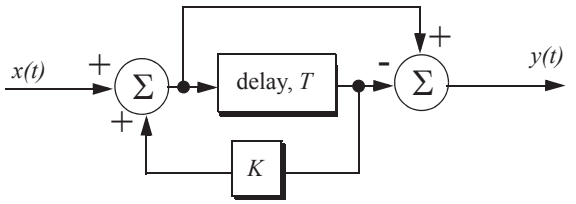


Figure 10.25. Clutter attenuation in the presence of phase noise; 2-pulse MTI case.

Problems

10.1. (a) Derive an expression for the impulse response of a single delay line canceler. (b) Repeat for a double delay line canceler.

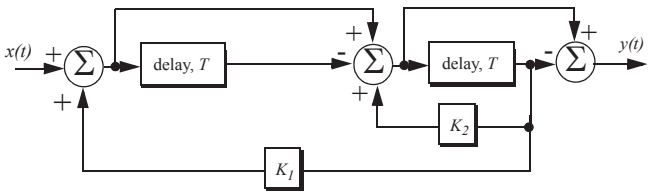
10.2. One implementation of a single delay line canceler with feedback is shown below.



(a) What is the transfer function, $H(z)$? (b) If the clutter power spectrum is $W(f) = w_0 \exp(-f^2/2\sigma_c^2)$, find an exact expression for the filter power gain. (c) Repeat part (b) for small values of frequency, f. (d) Compute the clutter attenuation and the improvement factor in terms of K and σ_c.

10.3. Plot the frequency response for the filter described in the previous problem for $K = -0.5, 0, and\ 0.5$.

10.4. An implementation of a double delay line canceler with feedback is shown below.



(a) What is the transfer function, $H(z)$? (b) Plot the frequency response for $K_1 = 0 = K_2$, and $K_1 = 0.2, K_2 = 0.5$.

10.5. Consider a single delay line canceler. Calculate the clutter attenuation and the improvement factor. Assume that $\sigma_c = 4Hz$ and PRF $f_r = 450Hz$.

10.6. Develop an expression for the improvement factor of a double delay line canceler.

10.7. Repeat Problem 9.10 for a double delay line canceler.

10.8. An experimental expression for the clutter power spectrum density is

$$W(f) = w_0 \exp(-f^2/2\sigma_c^2)$$

where w_0 is a constant. Show that using this expression leads to the same result obtained for the improvement factor as developed in Section 10.4.

10.9. A certain radar uses two PRFs with a stagger ratio 63/64. If the first PRF is $f_{r1} = 500Hz$. Compute the blind speeds for both PRFs and for the resultant composite PRF. Assume $\lambda = 3cm$.

10.10. Using PRI ratios 25:30:27:31, generate the MTI response for a 3-pulse MTI.

10.11. A certain filter used for clutter rejection has an impulse response $h(n) = \delta(n) - 3\delta(n-1) + 3\delta(n-2) - \delta(n-3)$. (a) Show an implementation of this filter using delay lines and adders. (b) What is the transfer function? (c) Plot the frequency response of this filter. (d) Calculate the output when the input is the unit step sequence.

10.12. The quadrature components of the clutter power spectrum are given in Problem 9.3. Let $\sigma_c = 10Hz$ and $f_r = 500Hz$. Compute the improvement of the signal-to-clutter ratio when a double delay line canceler is utilized.

10.13. The quadrature components of the clutter power spectrum are

$$\overline{S_I}(f) = \delta(f) + \frac{C}{\sqrt{2\pi}\sigma_c}\exp(-f^2/2\sigma_c^2)$$

$$\overline{S_Q}(f) = \frac{C}{\sqrt{2\pi}\sigma_c}\exp(-f^2/2\sigma_c^2).$$

Let $\sigma_c = 10Hz$ and $f_r = 500Hz$. Compute the improvement of the signal-to-clutter ratio when a double delay line canceler is utilized.

10.14. Develop an expression for the clutter improvement factor for single and double line cancelers using the clutter autocorrelation function.

10.15. Consider a medium PRF radar on board an aircraft moving at a speed of 350 m/s with PRFs $f_{r1} = 10KHz$, $f_{r2} = 15KHz$, and $f_{r3} = 20KHz$; the radar operating frequency is $9.5GHz$. Calculate the frequency position of a nose-on target with a speed of 300 m/s. Also calculate the closing rate of a target appearing at 6, 5, and $18KHz$ away from the center line of PRF 10, 15, and $20KHz$, respectively.

10.16. Repeat Problem 10.13 when the target is $15°$ off the radar line of sight.

10.17. A certain radar operates at two PRFs, f_{r1} and f_{r2}, where $T_{r1} = (1/f_{r1}) = T/5$ and $T_{r2} = (1/f_{r2}) = T/6$. Show that this multiple PRF scheme will give the same range ambiguity as that of a single PRF with PRI T.

10.18. Consider an X-band radar with wavelength $\lambda = 3cm$ and bandwidth $B = 10MHz$. The radar uses two PRFs, $f_{r1} = 50KHz$ and $f_{r2} = 55.55KHz$. A target is detected at range bin 46 for f_{r1} and at bin 12 for f_{r2}. Determine the actual target range.

10.19. A certain radar uses two PRFs to resolve range ambiguities. The desired unambiguous range is $R_u = 150Km$. Select a reasonable value for N. Compute the corresponding f_{r1}, f_{r2}, R_{u1}, and R_{u2}.

10.20. A certain radar uses three PRFs to resolve range ambiguities. The desired unambiguous range is $R_u = 250Km$. Select $N = 43$. Compute the corresponding f_{r1}, f_{r2}, f_{r3}, R_{u1}, R_{u2}, and R_{u3}.

10.21. Starting with Eq. (10.94), derive a closed form expression for the phase noise of an FM modulated waveform.

10.22. Reproduce Fig. 10.26 for a 3-pulse and a 4-pulse MTI system, and briefly discuss your output in terms how much phase noise affects the performance of each system,

Appendix 10-A: Chapter 10 MATLAB Code Listings

The MATLAB code provided in this chapter was designed as an academic standalone tool and is not adequate for other purposes. The code was written in a way to assist the reader in gaining a better understanding of the theory. The code was not developed, nor is it intended to be used as part of an open-loop or a closed-loop simulation of any kind. The MATLAB code found in this textbook can be downloaded from this book's web page on the CRC Press website. Simply use your favorite web browser, go to www.crcpress.com, and search for keyword *"Mahafza"* to locate this book's web page.

MATLAB Function "single_canceler.m" Listing

```
function [resp] = single_canceler (fofr1)
% single delay canceller
eps = 0.00001;
fofr = 0:0.01:fofr1;
arg1 = pi .* fofr;
resp = 4.0 .*((sin(arg1)).^2);
max1 = max(resp);
resp = resp ./ max1;
subplot(2,1,1)
plot(fofr,resp,'k')
xlabel ('Normalized frequency in f/fr')
ylabel( 'Amplitude response in Volts')
grid
subplot(2,1,2)
resp=10.*log10(resp+eps);
plot(fofr,resp,'k');
axis tight
grid
xlabel ('Normalized frequency in f/fr')
ylabel( 'Amplitude response in dB')
end
```

MATLAB Function "double_canceler.m" Listing

```
function [resp] = double_canceler(fofr1)
eps = 0.00001;
fofr = 0:0.01:fofr1;
arg1 = pi .* fofr;
resp = 4.0 .* ((sin(arg1)).^2);
max1 = max(resp);
resp = resp ./ max1;
resp2 = resp .* resp;
subplot(2,1,1);
plot(fofr,resp,'k--',fofr, resp2,'k');
ylabel ('Amplitude response - Volts')
resp2 = 20. .* log10(resp2+eps);
resp1 = 20. .* log10(resp+eps);
subplot(2,1,2)
plot(fofr,resp1,'k--',fofr,resp2,'k');
legend ('single canceler','double canceler')
xlabel ('Normalized frequency f/fr')
ylabel ('Amplitude response in dB')
```

```
end
```

MATLAB Program "Fig10_8.m" Listing

```
% generates Fig. 10.8 of text
clear all;
fofr = 0:0.001:1;
arg = 2.*pi.*fofr;
nume = 2.*(1.-cos(arg));
den11 = (1. + 0.25 * 0.25);
den12 = (2. * 0.25) .* cos(arg);
den1 = den11 - den12;
den21 = 1.0 + 0.7 * 0.7;
den22 = (2. * 0.7) .* cos(arg);
den2 = den21 - den22;
den31 = (1.0 + 0.9 * 0.9);
den32 = ((2. * 0.9) .* cos(arg));
den3 = den31 - den32;
resp1 = nume ./ den1;
resp2 = nume ./ den2;
resp3 = nume ./ den3;
plot(fofr,resp1,'k',fofr,resp2,'k-.',fofr,resp3,'k--');
xlabel('Normalized frequency')
ylabel('Amplitude response')
legend('K=0.25','K=0.7','K=0.9')
grid
axis tight
```

MATLAB Program "Fig10_9.m" Listing

```
% generates Fig 9.10 of text
clc
close all
clear all
fofr=0:0.001:1;
f1=4.*fofr;
f2=5.*fofr;
arg1=pi.*f1;
arg2=pi.*f2;
resp1=abs(sin(arg1));
resp2=abs(sin(arg2));
resp=resp1+resp2;
max1=max(resp);
resp=resp./max1;
subplot(3,1,1)
plot(fofr,resp1);
ylabel('\bfFilter response')
grid on
subplot(3,1,2)
plot(fofr,resp2);
ylabel('\bfFilter response')
grid on
subplot(3,1,3)
plot(fofr,resp);
```

```
ylabel('\bfFilter response')
xlabel('\bff/fr')
grid on
```

MATLAB Program "Fig10_10.m" Listing

```
% generates Fig 10.10 of text
k = .00035/25; a = 25*k; b = 30*k; c = 27*k; d = 31*k;
v2 = linspace(0,1345,10000);
f2 = (2.*v2)/.0375;
% H1(f)
T1 = exp(-j*2*pi.*f2*a); X1 = 1/2.*(1 - T1).*conj(1 - T1); H1 = 10*log10(abs(X1));
% H2(f)
T2 = exp(-j*2*pi.*f2*b); X2 = 1/2.*(1 - T2).*conj(1 - T2); H2 = 10*log10(abs(X2));
% H3(f)
T3 = exp(-j*2*pi.*f2*c); X3 = 1/2.*(1 - T3).*conj(1 - T3); H3 = 10*log10(abs(X3));
% H4(f)
T4 = exp(-j*2*pi.*f2*d); X4 = 1/2.*(1 - T4).*conj(1 - T4); H4 = 10*log10(abs(X4));
% Plot of the four components of H(f)
figure(1)
subplot(2,1,1)
% H(f) Average
ave2 = abs((X1 + X2 + X3 + X4)./4);
Have2 = 10*log10(abs((X1 + X2 + X3 + X4)./4));
plot(v2,Have2);
axis([0 1345 -25 5]);
 title('Two pulse MTI stagger ratio 25:30:27:31');
xlabel('Radial Velocity (m/s)');
 ylabel('MTI Gain (dB)'); grid on
% %Mean value of H(f)
v4 = v2; f4 = (2.*v4)/.0375;
% H1(f)
T1 = exp(-j*2*pi.*f4*a);
 T2 = exp(-j*2*pi.*f4*(a + b));
T3 = exp(-j*2*pi.*f4*(a + b + c));
X1 = 1/20.*(1 - 3.*T1 + 3.*T2 - T3).*conj(1 - 3.*T1 + 3.*T2 - T3);
H1 = 10*log10(abs(X1));
% H2(f)
T3 = exp(-j*2*pi.*f4*b);
T4 = exp(-j*2*pi.*f4*(b + c));
T5 = exp(-j*2*pi.*f4*(b + c + d));
X2 = 1/20.*(1 - 3.*T3 + 3.*T4 - T5).*conj(1 - 3.*T3 + 3.*T4 - T5);
H2 = 10*log10(abs(X2));
% H3(f)
T6 = exp(-j*2*pi.*f4*c);
T7 = exp(-j*2*pi.*f4*(c + d));
T8 = exp(-j*2*pi.*f4*(c + d + a));
X3 = 1/20.*(1 - 3.*T6 + 3.*T7 - T8).*conj(1 - 3.*T6 + 3.*T7 - T8);
H3 = 10*log10(abs(X3));
% H4(f)
T9 = exp(-j*2*pi.*f4*d); T10 = exp(-j*2*pi.*f4*(d + a));
T11 = exp(-j*2*pi.*f4*(d + a + b));
X4 = 1/20.*(1 - 3.*T9 + 3.*T10 - T11).*conj(1 - 3.*T9 + 3.*T10 - T11);
H4 = 10*log10(abs(X4));
```

```
% H(f) Average
ave4 = abs((X1 + X2 + X3 + X4)./4);
Have4 = 10*log10(abs((X1 + X2 + X3 + X4)./4));
% Plot of H(f) Average
subplot(2,1,2)
plot(v4,Have4);
axis([0 1345 -25 5]);
title('Four pulse MTI stagger ratio 25:30:27:31');
xlabel('Radial Velocity (m/s)');
ylabel('MTI Gain (dB)');
grid on
```

MATLAB Program "Fig10_23.m" Listing

```
% generates Fig. 10.23 of text
clc
close all;
clear all;
fc = 500;
f  = linspace(350,650, 300);
c1  = 2*1e-5;
fc1 = c1*pi*fc^2;
L1f = 1/pi*fc1./(fc1^2 + (f-fc).^2);
c2  = 0.5*1e-5;
fc2 = c2*pi*fc^2;
L2f = 1/pi*fc2./(fc2^2 + (f-fc).^2);
plot(f,L1f,'ro-','linewidth',1.);
hold on;
plot(f,L2f,'bd-','linewidth',1.);
xlabel('\bfFrequency - Hz');
ylabel('\bfL(f), ratio of noise power to carrier power');
axis([300 700 0 0.09]);
title('\bf fc=500Hz')
grid on;
legend('cp=2e-5','cp=0.5*1e-5');
```

MATLAB Program "Fig10_24.m" Listing

```
% generates Fig. 10.24 of text
clc; close all
clear all;
fc = 500;
f  = [0.01:.01:100];
fb = pi/2;
Lf = 1/pi*fb./(fb^2 + (f-fc+fc).^2);
semilogx((f),10*log10(Lf),'b','linewidth',1.5);
xlabel('\bfFrequency - Hz');
ylabel('\bfRatio of noise power to carrier power - dBc/Hz');
title('\bf fb= \pi /2')
axis([0.01 100 -50 10]); grid on;
```

MATLAB Program "Fig10_25.m" Listing

```
% generates Fig. 10.25 of text
```

```
clc
clear all
close all
format long
PRF = 1/400e-6;
taup = 1e-6;
sigma1 = 11.93;
sigma2 = 20.72;
phi0L = 10^-15.00;
phi0U = 10^-8;
phi0 = linspace(phi0L,phi0U,150000);
phi_ratio = phi0 ./ taup;
 % two-pulse MTI
%%%% sigma1 %%%%
gns = 1/2 * (2*pi*sigma1/PRF)^2
den = gns + phi_ratio;
CA_NS = 10*log10(1.0 ./ den);
%%%% sigma2 %%%%%
gs = 1/2 * (2*pi*sigma2/PRF)^2;
den = gs + phi_ratio;
CA_S = 10*log10(1.0 ./ den);
x_axis = 10*log10(phi0);
figure(1)
plot(x_axis,CA_NS,'r-.',x_axis,CA_S,'b','linewidth',1.5)
grid
% axis([-135 -90 20 35])
xlabel('\bfPhase noise - dBc/HZ')
ylabel('\bfClutter attenuation - dB')
legend('\sigmaf = 11.92', '\sigmaf = 20.72')
```

Part IV

Radar Detection

Chapter 11

Random Variables and Random Processes

The material in Part IV of this book, requires a strong background in random variables and random processes. Users of this book are advised to use this chapter as means for a quick top-level review of random variables and random processes. Instructors using this book as a text may assign Chapter 11 as a reading assignment to their students. This chapter is written in such a way that it only highlights the major points of the subject.

11.1. Random Variables

Consider an experiment with outcomes defined by a certain sample space. The rule or functional relationship that maps each point in this sample space into a real number is called a random variable. Random variables are designated by capital letters (e.g., X, Y, …), and a particular value of a random variable is denoted by a lowercase letter (e.g., x, y, …).

The Cumulative Distribution Function (*cdf*) associated with the random variable X is denoted as $F_X(x)$ and is interpreted as the total probability that the random variable X is less than or equal to the value x. More precisely,

$$F_X(x) = Pr\{X \le x\} . \qquad \textbf{Eq. (11.1)}$$

The probability that the random variable X is in the interval (x_1, x_2) is then given by

$$F_X(x_2) - F_X(x_1) = Pr\{x_1 \le X \le x_2\} . \qquad \textbf{Eq. (11.2)}$$

It is often practical to describe a random variable by the derivative of its *cdf*, which is called the Probability Density Function *(pdf)*. The *pdf* of the random variable X is

$$f_X(x) = \frac{\mathrm{d}}{\mathrm{d}x} F_X(x) \qquad \textbf{Eq. (11.3)}$$

or, equivalently,

$$F_X(x) = Pr\{X \le x\} = \int_{-\infty}^{x} f_X(\lambda) d\lambda . \qquad \textbf{Eq. (11.4)}$$

It follows that Eq. (11.2) can be written in the following equivalent form

$$F_X(x_2) - F_X(x_1) = Pr\{x_1 \le X \le x_2\} = \int_{x_1}^{x_2} f_X(x)dx . \qquad \textbf{Eq. (11.5)}$$

The *cdf* has the following properties:

$$\begin{aligned} 0 \le F_X(x) \le 1 \\ F_X(-\infty) = 0 \\ F_X(\infty) = 1 \\ F_X(x_1) \le F_X(x_2) \Leftrightarrow x_1 \le x_2 \end{aligned} \qquad \textbf{Eq. (11.6)}$$

Define the nth moment for the random variable X as

$$E[X^n] = \overline{X^n} = \int_{-\infty}^{\infty} x^n f_X(x)dx . \qquad \textbf{Eq. (11.7)}$$

The first moment, $E[X]$, is called the mean value, while the second moment, $E[X^2]$, is called the mean squared value. When the random variable X represents an electrical signal across a 1Ω resistor, then $E[X]$ is the DC component, and $E[X^2]$ is the total average power.

The nth central moment is defined as

$$E[(X-\bar{X})^n] = \overline{(X-\bar{X})^n} = \int_{-\infty}^{\infty} (x-\bar{x})^n f_X(x)dx . \qquad \textbf{Eq. (11.8)}$$

and thus the first central moment is zero. The second central moment is called the variance and is denoted by the symbol σ_X^2,

$$\sigma_X^2 = E[(X-\bar{X})^2] = \overline{(X-\bar{X})^2} . \qquad \textbf{Eq. (11.9)}$$

In practice, the random nature of an electrical signal may need to be described by more than one random variable. In this case, the joint *cdf* and *pdf* functions need to be considered. The joint *cdf* and *pdf* for the two random variables X and Y are, respectively, defined by

$$F_{XY}(x,y) = Pr\{X \le x; Y \le y\} \qquad \textbf{Eq. (11.10)}$$

$$f_{XY}(x,y) = \frac{\partial^2}{\partial x \partial y} F_{XY}(x,y) . \qquad \textbf{Eq. (11.11)}$$

The marginal *cdf*s are obtained as follows:

$$\begin{aligned} F_X(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{x} f_{UV}(u,v)dudv = F_{XY}(x,\infty) \\ F_Y(y) = \int_{-\infty}^{\infty}\int_{-\infty}^{y} f_{UV}(u,v)dvdu = F_{XY}(\infty,y) \end{aligned} \qquad \textbf{Eq. (11.12)}$$

If the two random variables are statistically independent, then the joint *cdf*s and *pdf*s are, respectively, given by

$$F_{XY}(x, y) = F_X(x)F_Y(y) \qquad \textbf{Eq. (11.13)}$$

$$f_{XY}(x, y) = f_X(x)f_Y(y)\,. \qquad \textbf{Eq. (11.14)}$$

Consider a case when the two random variables X and Y are mapped into two new variables U and V through some transformations T_1 and T_2 defined by

$$U = T_1(X, Y) \quad ; \; V = T_2(X, Y)\,. \qquad \textbf{Eq. (11.15)}$$

The joint *pdf,* $f_{UV}(u, v)$, may be computed based on the invariance of probability under the transformation. For this purpose, one must first compute the matrix of derivatives; then the new joint *pdf* is computed as

$$f_{UV}(u, v) = f_{XY}(x, y)|\mathbf{J}| \qquad \textbf{Eq. (11.16)}$$

$$|\mathbf{J}| = \begin{vmatrix} \dfrac{\partial x}{\partial u} & \dfrac{\partial x}{\partial v} \\ \dfrac{\partial y}{\partial u} & \dfrac{\partial y}{\partial v} \end{vmatrix} \qquad \textbf{Eq. (11.17)}$$

where the determinant of the matrix of derivatives $|\mathbf{J}|$ is called the Jacobian. The characteristic function for the random variable X is defined as

$$C_X(\omega) = E[e^{j\omega X}] = \int_{-\infty}^{\infty} f_X(x)e^{j\omega x}dx\,. \qquad \textbf{Eq. (11.18)}$$

The characteristic function can be used to compute the *pdf* for a sum of independent random variables. More precisely, let the random variable Y be

$$Y = X_1 + X_2 + \ldots + X_M \qquad \textbf{Eq. (11.19)}$$

where $\{X_m \; ; \; i = 1, \ldots, M\}$ is a set of independent random variables. It can be shown that

$$C_Y(\omega) = C_{X_1}(\omega)C_{X_2}(\omega)\ldots C_{X_M}(\omega) \qquad \textbf{Eq. (11.20)}$$

and the *pdf* $f_Y(y)$ is computed as the inverse Fourier transform of $C_Y(\omega)$ is

$$f_Y(y) = \frac{1}{2\pi}\int_{-\infty}^{\infty} C_Y(\omega)e^{-j\omega y}d\omega\,. \qquad \textbf{Eq. (11.21)}$$

The characteristic function may also be used to compute the nth moment for the random variable X as

$$E[X^n] = (-j)^n \frac{\mathrm{d}^n}{\mathrm{d}\omega^n} C_X(\omega)\bigg|_{\omega = 0}\,. \qquad \textbf{Eq. (11.22)}$$

11.2. Multivariate Gaussian Random Vector

Consider a joint probability for M random variables, $X_1, X_2, \ldots, X_M$. These variables can be represented as components of an $M \times 1$ random column vector, $\mathbf{x}$. More precisely,

$$\mathbf{x} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_M \end{bmatrix}. \qquad \text{Eq. (11.23)}$$

The joint *pdf* for the vector $\mathbf{x}$ is

$$f_X(\mathbf{x}) = f_{X_1, X_2, \ldots, X_M}(x_1, x_2, \ldots, x_M). \qquad \text{Eq. (11.24)}$$

The mean vector is defined as

$$\mu_{\mathbf{x}} = \begin{bmatrix} E[X_1] \\ E[X_2] \\ \vdots \\ E[X_M] \end{bmatrix} \qquad \text{Eq. (11.25)}$$

and the covariance is an $M \times M$ matrix given by

$$\mathbf{C}_X = E[\mathbf{x}\ \mathbf{x}^t] - \mu_{\mathbf{x}}\ \mu_{\mathbf{x}}^t \qquad \text{Eq. (11.26)}$$

where the superscript t indicates the transpose operation. Note that if the elements of the vector $\mathbf{x}$ are independent, then the covariance matrix is a diagonal matrix.

A random vector $\mathbf{x}$ is multivariate Gaussian if its *pdf* is of the form

$$f_X(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^M |\mathbf{C}_X|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_{\mathbf{x}})^t \mathbf{C}_X^{-1} (\mathbf{x} - \mu_{\mathbf{x}})\right) \qquad \text{Eq. (11.27)}$$

where $\mu_{\mathbf{x}}$ is the mean vector, $\mathbf{C}_X$ is the covariance matrix, $\mathbf{C}_X^{-1}$ is inverse of the covariance matrix, $|\mathbf{C}_X|$ is its determinant, and $\mathbf{x}$ is of dimension $M \times 1$. If $\mathbf{A}$ is a $K \times M$ matrix of rank K, then the random vector $\mathbf{y} = \mathbf{A}\mathbf{x}$ is a K-variate Gaussian vector with

$$\mu_{\mathbf{y}} = \mathbf{A}\mu_{\mathbf{x}} \qquad \text{Eq. (11.28)}$$

$$\mathbf{C}_Y = \mathbf{A}\ \mathbf{C}_X\ \mathbf{A}^t. \qquad \text{Eq. (11.29)}$$

The characteristic function for a multivariate Gaussian *pdf* is defined by

$$C_X = E[\exp\{j(\omega_1 X_1 + \omega_2 X_2 + \ldots + \omega_M X_M)\}] = \exp\left\{j\mu_{\mathbf{x}}^t \omega - \frac{1}{2}\omega^t \mathbf{C}_X \omega\right\}. \qquad \text{Eq. (11.30)}$$

Then the moments for the joint distribution can be obtained by partial differentiation. For example,

$$E[X_1X_2X_3] = \frac{\partial^3}{\partial\omega_1\partial\omega_2\partial\omega_3}C_X(\omega_1, \omega_2, \omega_3) \qquad at \quad \omega = \mathbf{0}. \qquad \textbf{Eq. (11.31)}$$

A special case of Eq. (11.29) is when the matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{bmatrix} a_1 a_2 \ \ldots \ a_M \end{bmatrix}. \qquad \textbf{Eq. (11.32)}$$

It follows that $\mathbf{y} = \mathbf{Ax}$ is a sum of random variables X_m, that is

$$Y = \sum_{m=1}^{M} a_m \ X_m. \qquad \textbf{Eq. (11.33)}$$

The finding in Eq. (11.33) leads to the conclusion that the linear sum of Gaussian variables is also a Gaussian variable with mean and variance given by

$$\overline{Y} = a_1\overline{X}_1 + a_2\overline{X}_2 + \ldots + a_M\overline{X}_M \qquad \textbf{Eq. (11.34)}$$

$$\sigma_Y^2 = E[(X-\overline{X})^2] = E[a_1^2(X_1-\overline{X}_1)^2] + E[a_2^2(X_2-\overline{X}_2)^2] + \ldots + E[a_M^2(X_M-\overline{X}_M)^2], \qquad \textbf{Eq. (11.35)}$$

and if the variables X_i are independent then Eq. (11.35) reduces to

$$\sigma_Y^2 = a_1^2\sigma_{X_1}^2 + a_2^2\sigma_{X_2}^2 + \ldots + a_M^2\sigma_{X_M}^2. \qquad \textbf{Eq. (11.36)}$$

Finally, in this case, the probability density function $f_Y(y)$ is given by (which can also be derived from Eq. (11.20))

$$f_Y(y) = f_{X_1}(x_1) \otimes f_{X_2}(x_2) \otimes \ldots \otimes f_{X_M}(x_M) \qquad \textbf{Eq. (11.37)}$$

where $\otimes$ indicates convolution.

Example:

Define

$$\mathbf{x}_1 = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \qquad \mathbf{x}_2 = \begin{bmatrix} X_3 \\ X_4 \end{bmatrix}$$

and the vector $\mathbf{x}$ *as a 4-variate Gaussian with*

$$\mu_{\mathbf{x}} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix} \text{ and } \mathbf{C}_X = \begin{bmatrix} 6 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 3 \end{bmatrix}.$$

Find the distribution of $\mathbf{x}_1$ *and the distribution of*

$$\mathbf{y} = \begin{bmatrix} 2\mathbf{x}_1 \\ \mathbf{x}_1 + 2\mathbf{x}_2 \\ \mathbf{x}_3 + \mathbf{x}_4 \end{bmatrix}.$$

Solution:

$\mathbf{x}_1$ *has a bivariate Gaussian distribution with*

$$\mu_{\mathbf{x}_1} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \qquad \mathbf{C}_{X_1} = \begin{bmatrix} 6 & 3 \\ 3 & 4 \end{bmatrix}.$$

The vector $\mathbf{y}$ *can be expressed as*

$$\mathbf{y} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} = \mathbf{A}\mathbf{x}.$$

It follows that

$$\mu_{\mathbf{y}} = \mathbf{A}\mu_{\mathbf{x}} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 1 \end{bmatrix}$$

$$\mathbf{C}_Y = \mathbf{A}\mathbf{C}_X \mathbf{A}^t = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 6 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 3 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 24 & 24 & 6 \\ 24 & 34 & 13 \\ 6 & 13 & 13 \end{bmatrix}.$$

11.2.1. Complex Multivariate Gaussian Random Vector

Consider the complex envelope for the $M \times 1$ vector random variable $\tilde{X}$ is,

$$\tilde{\mathbf{x}} = \mathbf{x}_I + j\mathbf{x}_Q \qquad \textbf{Eq. (11.38)}$$

where $\mathbf{x}_I$ and $\mathbf{x}_Q$ are real random multivariate Gaussian random vectors. The joint *pdf* for the complex random vector $\tilde{\mathbf{x}}$ is computed from the joint *pdf* of the two real vectors. The mean for the vector $\tilde{\mathbf{x}}$ is

$$E[\tilde{\mathbf{x}}] = E[\mathbf{x}_I] + jE[\mathbf{x}_Q]. \qquad \textbf{Eq. (11.39)}$$

The covariance matrix is also defined by

$$\mathbf{C}_{\tilde{X}} = E[(\tilde{\mathbf{x}} - E[\tilde{\mathbf{x}}])(\tilde{\mathbf{x}} - E[\tilde{\mathbf{x}}])^{\dagger}] \qquad \textbf{Eq. (11.40)}$$

where the operator † indicates complex conjugate transpose.

The *pdf* for the vector $\tilde{\mathbf{x}}$ is

$$f_{\tilde{X}}(\tilde{\mathbf{x}}) = \frac{\exp[-(\tilde{\mathbf{x}} - E[\tilde{\mathbf{x}}])^{\dagger}\mathbf{C}_{\tilde{X}}^{-1}(\tilde{\mathbf{x}} - E[\tilde{\mathbf{x}}])]}{\pi^{M}|\mathbf{C}_{\tilde{X}}|} \quad \text{Eq. (11.41)}$$

with the following three conditions holding true

$$E[(\mathbf{x}_{I_i} - E[\mathbf{x}_{I_i}])(\mathbf{x}_{Q_i} - E[\mathbf{x}_{Q_i}])^{\dagger}] = \mathbf{0} \quad \text{Eq. (11.42)}$$

$$E[(\mathbf{x}_{I_i} - E[\mathbf{x}_{I_i}])(\mathbf{x}_{I_k} - E[\mathbf{x}_{I_k}])^{\dagger}] = E[(\mathbf{x}_{Q_i} - E[\mathbf{x}_{Q_i}])(\mathbf{x}_{Q_k} - E[\mathbf{x}_{Q_k}])^{\dagger}] \;; all \;\; i,k \quad \text{Eq. (11.43)}$$

$$E[(\mathbf{x}_{I_i} - E[\mathbf{x}_{I_i}])(\mathbf{x}_{Q_k} - E[\mathbf{x}_{Q_k}])^{\dagger}] = -E[(\mathbf{x}_{Q_i} - E[\mathbf{x}_{Q_i}])(\mathbf{x}_{I_k} - E[\mathbf{x}_{I_k}])^{\dagger}] \;\; ; \;\; i \neq k \,. \quad \text{Eq. (11.44)}$$

11.3. Rayleigh Random Variables

Let X_I and X_Q be zero mean independent Gaussian random variables with zero mean and variance σ^2. Define two new random variables R and Φ as

$$\begin{aligned} X_I &= R\cos\Phi \\ X_Q &= R\sin\Phi \end{aligned} \quad \text{Eq. (11.45)}$$

The joint *pdf* of the two random variables $X_I;X_Q$ is

$$f_{X_I X_Q}(x_I, x_Q) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x_I^2 + x_Q^2}{2\sigma^2}\right) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(r\cos\varphi)^2 + (r\sin\varphi)^2}{2\sigma^2}\right). \quad \text{Eq. (11.46)}$$

The joint *pdf* for the two random variables $R;\Phi$ is given by

$$f_{R\Phi}(r, \varphi) = f_{X_I X_Q}(x_I, x_Q)|\mathbf{J}| \quad \text{Eq. (11.47)}$$

where $[\mathbf{J}]$ is a matrix of derivatives defined by

$$[\mathbf{J}] = \begin{bmatrix} \dfrac{\partial x_I}{\partial r} & \dfrac{\partial x_I}{\partial \varphi} \\ \dfrac{\partial x_Q}{\partial r} & \dfrac{\partial x_Q}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -r\sin\varphi \\ \sin\varphi & r\cos\varphi \end{bmatrix}. \quad \text{Eq. (11.48)}$$

The determinant of the matrix of derivatives is called the Jacobian, and in this case it is equal to

$$|\mathbf{J}| = r \quad \text{Eq. (11.49)}$$

Substituting Eqs. (11.46) and (11.49) into Eq. (11.47) and collecting terms yields

$$f_{R\Phi}(r, \varphi) = \frac{r}{2\pi\sigma^2}\exp\left(-\frac{(r\cos\varphi)^2 + (r\sin\varphi)^2}{2\sigma^2}\right) = \frac{r}{2\pi\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right). \quad \text{Eq. (11.50)}$$

The *pdf* for R alone is obtained by integrating Eq. (11.50) over φ

$$f_R(r) = \int_0^{2\pi} f_{R\Phi}(r, \varphi) d\varphi = \frac{r}{\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right) \frac{1}{2\pi} \int_0^{2\pi} d\varphi \qquad \text{Eq. (11.51)}$$

where the integral inside Eq. (11.51) is equal to 2π; thus,

$$f_R(r) = \frac{r}{\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad ; r \geq 0 \ . \qquad \text{Eq. (11.52)}$$

The *pdf* described in Eq. (11.52) is referred to as a Rayleigh probability density function. The density function for the random variable Φ is obtained from

$$f_\Phi(\varphi) = \int_0^r f(r, \varphi) \ dr \, . \qquad \text{Eq. (11.53)}$$

substituting Eq. (11.50) into Eq. (11.53) and performing integration by parts yields

$$f_\Phi(\varphi) = \frac{1}{2\pi} \quad ; \ 0 < \varphi < 2\pi \, , \qquad \text{Eq. (11.54)}$$

which is a uniform probability density function.

11.4. The Chi-Square Random Variables

11.4.1. Central Chi-Square Random Variable with N Degrees of Freedom

Let the random variables $\{X_1, X_2, \ldots, X_M\}$ be zero mean, statistically independent Gaussian random variable with unity variance. The variable

$$\chi_N^2 = \sum_{m=1}^{M} X_m^2 \qquad \text{Eq. (11.55)}$$

is the central chi-square random variable with M degrees of freedom. The chi-square *pdf* is

$$f_{\chi_M^2}(x) = \begin{cases} \dfrac{x^{(M-2)/2} \ e^{(-x/2)}}{2^{M/2} \ \Gamma(M/2)} & x \geq 0 \\ 0 & x < 0 \end{cases} \qquad \text{Eq. (11.56)}$$

where the Gamma function is defined as

$$\Gamma(m) = \int_0^{\infty} \lambda^{m-1} e^{-\lambda} \ d\lambda \ ; \ m > 0 \qquad \text{Eq. (11.57)}$$

with the following recursion

$$\Gamma(m+1) = m\Gamma(m) \qquad \text{Eq. (11.58)}$$

and

$$\Gamma(m+1) = m! \qquad ; \; m = 0, 1, 2, \ldots, and \;\; 0! = 1 . \qquad \textbf{Eq. (11.59)}$$

The mean and variance for the central chi-square are, respectively, given by

$$E[\chi_M^2] = M \qquad \textbf{Eq. (11.60)}$$

$$\sigma_{\chi_M^2} = 2M . \qquad \textbf{Eq. (11.61)}$$

Hence, the degrees of freedom M is the ratio of twice the squared mean to the variance

$$M = (2E^2[\chi_M^2]) / \sigma_{\chi_M^2} . \qquad \textbf{Eq. (11.62)}$$

11.4.2. Non-Central Chi-Square Random Variable with N Degrees of Freedom

In the general, the chi-square random variable requires that the Gaussian random variables $\{X_1, X_2, \ldots, X_M\}$ do not have zero means. Define a multivariate random variable $\mathbf{y}$ such that

$$Y_m = X_m + \mu_{X_m} \;; m = 1, 2, \ldots, M \qquad \textbf{Eq. (11.63a)}$$

$$\mathbf{y} = \mathbf{x} + \mu_{\mathbf{x}} . \qquad \textbf{Eq. (11.63b)}$$

Consider the random variable

$$\chi'^2_M = \sum_{m=1}^{M} Y_m^2 = \sum_{m=1}^{M} (X_m + \mu_{X_m})^2 . \qquad \textbf{Eq. (11.64)}$$

the variable χ'^2_M is called the non-central chi-square random variable with M degrees of freedom and with a non-central parameter λ, where

$$\lambda = \sum_{m=1}^{M} \mu_{X_m}^2 = \sum_{m=1}^{M} E^2[Y_m] . \qquad \textbf{Eq. (11.65)}$$

The non-central chi-square *pdf* is

$$f_{\chi'^2_M}(x) = \begin{cases} \left(\frac{1}{2}\right)\left(\frac{x}{\lambda}\right)^{(M-2)/4} e^{[-(x+\lambda)/2]} I_{(M-2)/2}(\sqrt{\lambda x}) & x \geq 0 \\ 0 & x < 0 \end{cases} \qquad \textbf{Eq. (11.66)}$$

where I is the modified Bessel function (or occasionally called the hyperbolic Bessel function) of the first kind; and the subscripts are referred to as its order.

11.5. Random Processes

A random variable X is by definition a mapping of all possible outcomes of a random experiment to numbers. When the random variable becomes a function of both the outcome of the experiment and time, it is called a random process and is denoted by $X(t)$. Thus, one can view

a random process as an ensemble of time-domain functions that are the outcome of a certain random experiment, as compared with single real numbers in the case of a random variable.

Since the *cdf* and *pdf* of a random process are time dependent, we will denote them as $F_X(x;t)$ and $f_X(x;t)$, respectively. The nth moment for the random process $X(t)$ is

$$E[X^n(t)] = \int_{-\infty}^{\infty} x^n f_X(x;t)dx. \quad \textbf{Eq. (11.67)}$$

A random process $X(t)$ is referred to as stationary to order one if all its statistical properties do not change with time. Consequently, $E[X(t)] = \overline{X}$, where $\overline{X}$ is a constant. A random process $X(t)$ is called stationary to order two (or wide-sense stationary) if

$$f_X(x_1, x_2;t_1, t_2) = f_X(x_1, x_2;t_1 + \Delta t, t_2 + \Delta t) \quad \textbf{Eq. (11.68)}$$

for all t_1, t_2 and Δt.

Define the statistical autocorrelation function for the random process $X(t)$ as

$$\Re_X(t_1, t_2) = E[X(t_1)X(t_2)]. \quad \textbf{Eq. (11.69)}$$

The correlation $E[X(t_1)X(t_2)]$ is, in general, a function of (t_1, t_2). As a consequence of the wide-sense stationary definition, the autocorrelation function depends on the time difference $\tau = t_2 - t_1$, rather than on absolute time; and thus, for a wide-sense stationary process we have

$$\begin{aligned} E[X(t)] &= \overline{X} \\ \Re_X(\tau) &= E[X(t)X(t+\tau)] \end{aligned}. \quad \textbf{Eq. (11.70)}$$

If the time average and time correlation functions are equal to the statistical average and statistical correlation functions, the random process is referred to as an ergodic random process. The following is true for an ergodic process:

$$\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t)dt = E[X(t)] = \overline{X} \quad \textbf{Eq. (11.71)}$$

$$\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x^*(t)x(t+\tau)dt = \Re_X(\tau). \quad \textbf{Eq. (11.72)}$$

The covariance of two random processes $X(t)$ and $Y(t)$ is defined by

$$C_{XY}(t, t+\tau) = E[\{X(t) - E[X(t)]\}\{Y(t+\tau) - E[Y(t+\tau)]\}], \quad \textbf{Eq. (11.73)}$$

which can also be written as

$$C_{XY}(t, t+\tau) = \Re_{XY}(\tau) - \overline{X}\,\overline{Y}. \quad \textbf{Eq. (11.74)}$$

11.6. The Gaussian Random Process

Let $X(t)$ be a random process defined over the interval $\{0, T\}$, then $X(t)$ is said to be a Gaussian random process if every possible outcome of this process over this interval is a Gaussian process, provided that the mean square value of this process is finite. More precisely, $Y(t)$ will be a random process over the same interval $\{0, T\}$ defined by

$$Y(t) = \int_0^T x(t)z(t)\ dt \qquad \text{Eq. (11.75)}$$

where $z(t)$ is any function that yields $E[|Y|^2] < \infty$.

Gaussian random processes have a few unique properties that distinguish them from other types of random processes. (1) If the input to an LTI system is said to be a Gaussian random process, then its output is also a Gaussian random process. (2) If $X(t)$ is a Gaussian random process for any set of time occurrences $\{t_1, t_2, \ldots, t_M\}$, then the random variables $\{X(t_1), X(t_2), \ldots, X(t_M)\}$ are jointly Gaussian random variables. Finally, (3) any linear combination of a Gaussian process yields another jointly Gaussian random variable.

11.6.1. Lowpass Gaussian Random Processes

Let $X(t)$ be a real-valued Gaussian random process. If this process is an essentially band-limited process (recall the definition of an essentially band-limited signals in Chapter3) over the frequency interval $\{-B, B\}$, then the minimal number of samples required to represent this process is $M = 2TB$ real samples. Therefore, over the interval $\{0, T\}$, there are M random variables represented by the vector $\mathbf{x}$ made of M random variables, that is

$$\mathbf{x} = \begin{bmatrix} X(1/2B) \\ X(1/B) \\ \vdots \\ X(M/2B) \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_M \end{bmatrix}. \qquad \text{Eq. (11.76)}$$

If the random process $X(t)$ is a complex lowpass Gaussian process, represented by its complex envelop $\tilde{X}(t)$, then in this case, the minimal number of samples required to represent this process is $M = TB$ complex samples. The resulting jointly Gaussian complex random vector comprising $M = TB$ complex random variables is

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{X}(1/B) \\ \tilde{X}(2/B) \\ \vdots \\ \tilde{X}(M/B) \end{bmatrix} = \begin{bmatrix} \tilde{X}_1 \\ \tilde{X}_2 \\ \vdots \\ \tilde{X}_M \end{bmatrix}. \qquad \text{Eq. (11.77)}$$

If the power spectral destiny of a real Gaussian random process $X(t)$ is defined by

$$S_X(f) = \begin{cases} S_o & ;|f| < B \\ 0 & ;otherwise \end{cases}, \qquad \text{Eq. (11.78)}$$

then the probability density function of the vector $\mathbf{x}$ is given by

$$f_X(x(t)) = \frac{1}{(4\pi S_o B)^{M/2}} \exp\left[\left(-\frac{1}{4S_o B}\right)\sum_{m=1}^{M} X_m\right] = \frac{1}{(4\pi S_o B)^{M/2}} \exp\left(-\frac{\mathbf{x}^t\mathbf{x}}{4S_o B}\right). \quad \text{Eq. (11.79)}$$

The mean of the random process defined in Eq. (11.76) is

$$\mu_{\mathbf{x}} = \begin{bmatrix} E[X_1] \\ E[X_2] \\ \vdots \\ E[X_M] \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_M \end{bmatrix}. \quad \text{Eq. (11.80)}$$

When the power spectral density of the process is non-white over the bandwidth, then in this case the random variables defined in Eq. (11.76) are no longer independent. Therefore, the *pdf* given in Eq. (11.79) is modified to

$$f_X(x(t)) = \frac{1}{\sqrt{(2\pi)^M \ \mathbf{C}_X}} \exp\left[\left(-\frac{1}{2}\right)(\mathbf{x}-\mu_{\mathbf{x}})^t \mathbf{C}_X^{-1}(\mathbf{x}-\mu_{\mathbf{x}})\right] \quad \text{Eq. (11.81)}$$

where the covariance matrix is

$$\mathbf{C}_X = E[(\mathbf{x}-\mu_{\mathbf{x}})(\mathbf{x}-\mu_{\mathbf{x}})^t]. \quad \text{Eq. (11.82)}$$

11.6.2. Bandpass Gaussian Random Processes

It is customary to define the bandpass Gaussian random process through its complex envelope as

$$\tilde{X}(t) = X_I(t) + jX_Q(t) \quad \text{Eq. (11.83)}$$

where both $X_I(t)$ and $X_Q(t)$ are jointly lowpass statistically independent and stationary Gaussian random processes with zero mean and equal variance σ^2. The *pdf* for a sample $\tilde{X}(t_0)$ of the complex envelope is the joint *pdf* for $X_I(t)$ and $X_Q(t)$. That is,

$$f_X(\tilde{x}(t_0)) = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{x_I^2(t_0)+x_Q^2(t_0)}{2\sigma^2}\right] = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{|\tilde{x}(t_0)|^2}{2\sigma^2}\right]. \quad \text{Eq. (11.84)}$$

Now, if both lowpass processes do not have zero mean and instead have a mean defined by

$$\mu(t) = \mu_I(t)\cos(2\pi f_0 t) + j\mu_Q(t)\sin(2\pi f_0 t), \quad \text{Eq. (11.85)}$$

the mean complex envelope is

$$\tilde{\mu}(t) = \mu_I(t) + j\mu_Q(t). \quad \text{Eq. (11.86)}$$

It follows that Eq. (11.84) can be rewritten as

$$f_X(\tilde{x}(t_0)) = \frac{1}{2\pi\sigma^2}\exp\left[-\frac{[x_I(t_0)-\mu_I(t_0)]^2+[x_Q(t_0)-\mu_Q(t_0)]^2}{2\sigma^2}\right] = \frac{1}{2\pi\sigma^2}\exp\left[\frac{\left|\tilde{x}(t_0)-\tilde{\mu}(t_0)\right|^2}{2\sigma^2}\right] \quad \text{Eq. (11.87)}$$

Consider a duration of the process that spans the interval $\{0, T\}$. Then this segment of the complex envelope of the random process can be represented using a complex random variable vector of at least $M = BT$ elements where B is the bandwidth of the process. Define

$$\tilde{X}_i = \tilde{X}\left(\frac{m}{B}\right) \quad ;m = 1, 2, \ldots, BT = M \quad \text{Eq. (11.88)}$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{X}_1 \\ \tilde{X}_2 \\ \vdots \\ \tilde{X}_M \end{bmatrix}. \quad \text{Eq. (11.89)}$$

By definition, the covariance matrix $\mathbf{C}$ is

$$\tilde{\mathbf{C}}_X = E[(\tilde{\mathbf{x}}-\tilde{\mu}_{\mathbf{x}})(\tilde{\mathbf{x}}-\tilde{\mu}_{\mathbf{x}})^{\dagger}] = 2(\tilde{\mathbf{C}}_{X_I}+j\tilde{\mathbf{C}}_{X_{IQ}}) \quad \text{Eq. (11.90)}$$

where

$$\tilde{\mathbf{C}}_{X_I} = E[(\tilde{\mathbf{x}}_I-\tilde{\mu}_{\mathbf{x}_I})(\tilde{\mathbf{x}}_I-\tilde{\mu}_{\mathbf{x}_I})^{\dagger}] \quad \text{Eq. (11.91)}$$

$$\tilde{\mathbf{C}}_{X_{IQ}} = E[(\tilde{\mathbf{x}}_I-\tilde{\mu}_{\mathbf{x}_I})(\tilde{\mathbf{x}}_Q-\tilde{\mu}_{\mathbf{x}_Q})^{\dagger}]. \quad \text{Eq. (11.92)}$$

Therefore, the *pdf* for the segment $\{\tilde{X}(t) \;;0 < t < T\}$ is

$$f_X(\tilde{\mathbf{x}}) = \frac{\exp[-(\tilde{\mathbf{x}}-\tilde{\mu}_{\mathbf{x}})^{\dagger}\tilde{\mathbf{C}}_X^{-1}(\tilde{\mathbf{x}}-\tilde{\mu}_{\mathbf{x}})]}{\pi^N\left|\tilde{\mathbf{C}}_X\right|}. \quad \text{Eq. (11.93)}$$

11.6.3. The Envelope of a Bandpass Gaussian Process

Consider the *pdf* of a segment of the envelope of a bandpass Gaussian random process. This process can be expressed as

$$X(t) = X_I(t)\cos(2\pi f_0 t) - X_Q(t)\sin(2\pi f_0 t) \quad \text{Eq. (11.94)}$$

where $X_I(t)$ and $X_Q(t)$ are zero mean independent lowpass Gaussian processes. The envelope and phase are respectively denoted by $R(t)$ and $\Phi(t)$, where

$$R(t) = \sqrt{X_I(t)^2 + X_Q(t)^2} \quad \text{Eq. (11.95)}$$

and

$$\Phi(t) = \left[\tan\left(\frac{X_Q(t)}{X_I(t)}\right)\right]^{-1} \qquad \text{Eq. (11.96)}$$

where

$$\begin{aligned} X_I(t) &= R(t)\cos(\Phi(t)) \\ X_Q(t) &= R(t)\sin(\Phi(t)) \end{aligned} \qquad \text{Eq. (11.97)}$$

The two processes $R(t)$ and $\Phi(t)$ are also independent, and their respective *pdfs* were derived in Section 11.3 and were given in Eqs. (11.52) and (11.54), respectively.

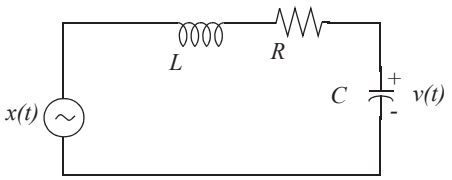
Problems

11.1. Suppose you want to determine an unknown DC voltage v_{dc} in the presence of additive white Gaussian noise $n(t)$ of zero mean and variance σ_n^2. The measured signal is $x(t) = v_{dc} + n(t)$. An estimate of v_{dc} is computed by making three independent measurements of $x(t)$ and computing the arithmetic mean, $\hat{v}_{dc} \approx (x_1 + x_2 + x_3)/3$. (a) Find the mean and variance of the random variable $\hat{v}_{dc}$. (b) Does the estimate of v_{dc} get better by using ten measurements instead of three? Why?

11.2. Assume the X and Y miss distances of darts thrown at a bulls-eye dart board are Gaussian with zero mean and variance σ^2. (a) Determine the probability that a dart will fall between 0.8σ and 1.2σ. (b) Determine the radius of a circle about the bull's-eye that contains 80% of the darts thrown. (c) Consider a square with side s in the first quadrant of the board. Determine s so that the probability that a dart will fall within the square is 0.07.

11.3. (a) A random voltage $v(t)$ has an exponential distribution function $f_V(v) = a\exp(-av)$, where $(a > 0);(0 \le v < \infty)$. The expected value $E[V] = 0.5$. Determine $Pr\{V > 0.5\}$.

11.4. Consider the network shown in the figure below, where $x(t)$ is a random voltage with zero mean and autocorrelation function $\Re_x(\tau) = 1 + \exp(-a|t|)$. Find the power spectrum $S_x(\omega)$. What is the transfer function? Find the power spectrum $S_v(\omega)$.



11.5. Let $\bar{S}_X(\omega)$ be the PSD function for the stationary random process $X(t)$. Compute an expression for the PSD function of

$$Y(t) = X(t) - 2X(t - T).$$

11.6. Let X be a random variable with

$$f_X(x) = \begin{Bmatrix} \frac{1}{\sigma} t^3 e^{-t} & t \geq 0 \\ 0 & elsewhere \end{Bmatrix}.$$

(a) Determine the characteristic function $C_X(\omega)$. (b) Using $C_X(\omega)$, validate that $f_X(x)$ is a proper *pdf*. (c) Use $C_X(\omega)$ to determine the first two moments of X. (d) Calculate the variance of X.

11.7. Let the random variable Z be written in terms of two other random variables X and Y as follows: $Z = X + 3Y$. Find the mean and variance for the new random variable in terms of the other two.

11.8. Suppose you have the following sequences of statistically independent Gaussian random variables with zero means and variances σ^2. if

$$X_1, X_2, \ldots, X_N \; ; \; X_i = A_i \cos\Theta_i \text{ and } Y_1, Y_2, \ldots, Y_N \; ; \; Y_i = A_i \sin\Theta_i .$$

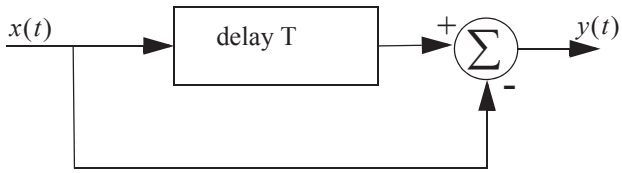
Define $Z = \sum_{i=1}^{N} A_i^2$. Find an expression where Z exceeds a threshold value v_T.

11.9. Repeat the previous problem when two single delay line cancelers are cascaded to produce a double delay line canceler. Let $X(t)$ be a stationary random process, $E[X(t)] = 1$ and the autocorrelation $\Re_x(\tau) = 3 + \exp(-|\tau|)$. Define a new random variola Y as

$$Y = \int_0^2 x(t)dt . \qquad \textbf{Eq. (11.98)}$$

Compute $E[Y(t)]$ and σ_Y^2.

11.10. Consider the single delay line canceler in the figure below. The input $x(t)$ is a wide-sense stationary random process with variance σ_x^2 and mean μ_x and a covariance matrix Λ. Find the mean and variance and the autocorrelation function of the output $y(t)$.



Chapter 12

Single Pulse Detection

12.1. Single Pulse with Known Parameters

In its simplest form, a radar signal can be represented by a single pulse comprising a sinusoid of known amplitude and phase. Consequently, a retuned signal will also comprise a sinusoid. Under the assumption of completely known signal parameters, a returned pulse from a target has known amplitude and known phase with no random components; and the radar signal processor will attempt to maximize the probability of detection for a given probability of false alarm. In this case, detection is referred to as coherent detection or coherent demodulation. A radar system will declare detection with a certain probability of detection if the received voltage signal envelope exceeds a pre-set threshold value. For this purpose, the radar receiver is said to employ an envelope detector.

Figure 12.1 shows a simplified block diagram of a radar matched filter receiver followed by a threshold decision logic. The signal at the input of the matched filter $s(t)$ is composed of the target echo signal $x(t)$ and additive zero mean Gaussian noise (white noise is assumed in the analysis presented in this chapter) random process $n(t)$, with variance σ^2. The input noise is assumed to be spatially incoherent and uncorrelated with the signal. The matched filter impulse response is $h(t)$, and its output is denoted by the signal $v(t)$, which was derived in Chapter 4; it is given by

$$v(t) = \int_{-\infty}^{\infty} s(t)h(t-u)\ du\,. \qquad \text{Eq. (12.1)}$$

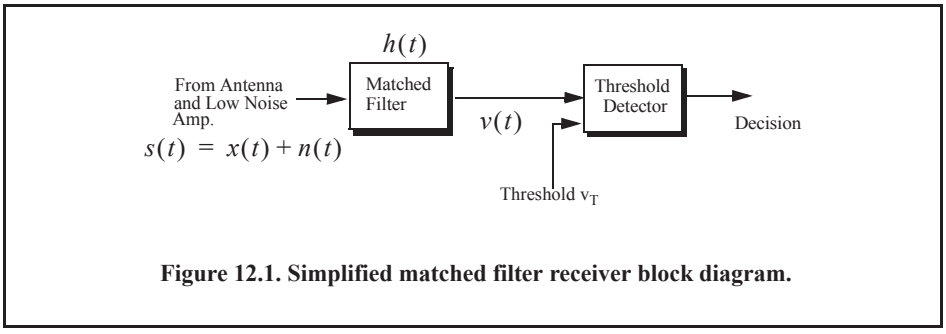


Figure 12.1. Simplified matched filter receiver block diagram.

Recall that if $n(t)$ is a Gaussian random process, then so is $s(t)$, since $x(t)$ is a deterministic signal; its only effect is a shift of the mean of the random process. Following the definition of the Gaussian process developed in Section 11.6 of Chapter 11, one concludes that the signal $v(t)$ is also a Gaussian random process, and over a coherent processing interval $\{0, T\}$, two hypotheses are considered, they are:

H_0 when the signal $s(t)$ is made of noise only, and

H_1 when the signal $s(t)$ is made of signal plus noise.

More specifically, by following the analysis in Section 11.6.1 of Chapter 11, one gets

$$H_0 \Leftrightarrow \mathbf{s} = \mathbf{n} \qquad ;0 < t < T \qquad \textbf{Eq. (12.2)}$$

$$H_1 \Leftrightarrow \mathbf{s} = \mathbf{n} + \mathbf{x} \qquad ;0 < t < T \qquad \textbf{Eq. (12.3)}$$

where all vectors are of size $M \times 1$ ($M = 2TB$), and B is the operating bandwidth of the receiver. It follows that,

$$\begin{aligned} E[\mathbf{n}] &= \mathbf{0} \\ E[\mathbf{n}\mathbf{n}^{\dagger}] &= \mathbf{C}_n \\ E[\mathbf{s}/H_1] &= \mathbf{x} \end{aligned} \qquad \textbf{Eq. (12.4)}$$

where $\mathbf{C}_n$ is the noise covariance matrix.

When the noise $n(t)$ is white, or it is band-limited white over the frequency band $\{-B, B\}$, then its power spectrum density is given by

$$\bar{S}_n(f) = \frac{\eta_o}{2} \qquad ;-B < f < B \qquad \textbf{Eq. (12.5)}$$

where η_o is a constant. Analysis of the non-white noise case is left as an exercise. The conditional probability for the H_0 case was derived in Eq. (11.79) of Chapter 11; it is

$$f(\mathbf{s}/H_0) = \frac{1}{(2\pi\eta_o B)^{M/2}} \exp\left(-\frac{\mathbf{s}^{\dagger}\mathbf{s}}{2\eta_o B}\right). \qquad \textbf{Eq. (12.6)}$$

Alternatively, the conditional probability for H_1 is identical to Eq. (12.6) except in this case, one must replace $\mathbf{s}$ by $\mathbf{s} - \mathbf{x}$. It follows that

$$f(\mathbf{s}/H_1) = \frac{1}{(2\pi\eta_o B)^{M/2}} \exp\left(-\frac{(\mathbf{s}-\mathbf{x})^{\dagger}(\mathbf{s}-\mathbf{x})}{2\eta_o B}\right). \qquad \textbf{Eq. (12.7)}$$

As determined earlier, the statistics associated with the random process $v(t)$ over the interval $\{0, T\}$ is Gaussian. In general, a Gaussian *pdf* function is given by

$$f_V(v) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(v-\bar{V})^2}{2\sigma^2}\right) \qquad \textbf{Eq. (12.8)}$$

where σ^2 is the variance and $\bar{V}$ is the mean value. It follows (the proof is left as an exercise (see Problem 12.1) that

$$E[V/H_0] = 0 \qquad \textbf{Eq. (12.9)}$$

$$Var[V/H_0] = \frac{E_x \eta_o}{2} = \sigma^2 \qquad \textbf{Eq. (12.10)}$$

$$E[V/H_1] = \int_0^T x^*(t)x(t)\ dt = E_x \qquad \textbf{Eq. (12.11)}$$

$$Var[V/H_1] = \frac{E_x \eta_o}{2} = \sigma^2 \qquad \textbf{Eq. (12.12)}$$

where E_x is the signal's energy.

Assuming the H_0 hypothesis, then the probability of a false P_{fa} alarm is computed from Eq. (12.8) when the signal $v(t)$ exceeds a set threshold value V_T. More specifically,

$$P_{fa} = Pr\{v(t) > V_T/H_0\} = \int_{V_T}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{v^2}{2\sigma^2}\right) dv\,. \qquad \textbf{Eq. (12.13)}$$

Substituting the variance as computed in Eq. (12.10) into Eq. (12.13) yields,

$$P_{fa} = \int_{V_T}^{\infty} \frac{1}{\sqrt{\pi E_x \eta_o}} \exp\left(-\frac{v^2}{E_x \eta_o}\right) dv\,. \qquad \textbf{Eq. (12.14)}$$

Making the change of variable $\zeta = v/(\sqrt{E_x \eta_o})$ yields

$$P_{fa} = \int_{\frac{V_T}{\sqrt{E_x \eta_o}}}^{\infty} \frac{1}{\sqrt{\pi}} e^{-\zeta^2} d\zeta\,. \qquad \textbf{Eq. (12.15)}$$

Multiplying and dividing Eq. (12.15) by *2* yields

$$P_{fa} = \frac{2}{2}\frac{1}{\sqrt{\pi}} \int_{\frac{V_T}{\sqrt{E_x \eta_o}}}^{\infty} e^{-\zeta^2} d\zeta = \frac{1}{2} erfc\left(\frac{V_T}{\sqrt{E_x \eta_o}}\right) \qquad \textbf{Eq. (12.16)}$$

where *erfc* is the complimentary error function defined by

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{-\zeta^2} d\zeta\,. \qquad \textbf{Eq. (12.17)}$$

The error function *erf* is related to the complimentary error function using the relation

$$erfc(z) = 1 - erf(z) = 1 - \frac{2}{\sqrt{\pi}} \int_0^z e^{-\zeta^2} d\zeta\,. \qquad \textbf{Eq. (12.18)}$$

Both *erf* and *erfc* are intrinsic MATLAB functions found within the Signal Processing Toolbox. Using similar analysis, one can derive the probability of detection as

$$P_D = Pr\{v(t) > V_T / H_1\} = \frac{1}{2} erfc\left(\frac{V_T - Ex}{\sqrt{E_x \eta_o}}\right). \qquad \text{Eq. (12.19)}$$

The derivation of Eq. (12.19) is left as an exercise (see Problem 12.2).

Table 12.1 gives samples of the single pulse SNR corresponding to few values of P_D and P_{fa}, using Eq. (12.19). For example, if $P_D = 0.99$ and $P_{fa} = 10^{-10}$, then the minimum single pulse SNR required to accomplish this combination of P_D and P_{fa} is $SNR = 16.12 dB$.

TABLE 12.1. Single Pulse SNR (dB)

	P_{fa}									
P_D	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
.1	4.00	6.19	7.85	8.95	9.94	10.44	11.12	11.62	12.16	12.65
.2	5.57	7.35	8.75	9.81	10.50	11.19	11.87	12.31	12.85	13.25
.3	6.75	8.25	9.50	10.44	11.10	11.75	12.37	12.81	13.25	13.65
.4	7.87	8.85	10.18	10.87	11.56	12.18	12.75	13.25	13.65	14.00
.5	8.44	9.45	10.62	11.25	11.95	12.60	13.11	13.52	14.00	14.35
.6	8.75	9.95	11.00	11.75	12.37	12.88	13.50	13.87	14.25	14.62
.7	9.56	10.50	11.50	12.31	12.75	13.31	13.87	14.20	14.59	14.95
.8	10.18	11.12	12.05	12.62	13.25	13.75	14.25	14.55	14.87	15.25
.9	10.95	11.85	12.65	13.31	13.85	14.25	14.62	15.00	15.45	15.75
.95	11.50	12.40	13.12	13.65	14.25	14.64	15.10	15.45	15.75	16.12
.98	12.18	13.00	13.62	14.25	14.62	15.12	15.47	15.85	16.25	16.50
.99	12.62	13.37	14.05	14.50	15.00	15.38	15.75	16.12	16.47	16.75
.995	12.85	13.65	14.31	14.75	15.25	15.71	16.06	16.37	16.65	17.00
.998	13.31	14.05	14.62	15.06	15.53	16.05	16.37	16.7	16.89	17.25
.999	13.62	14.25	14.88	15.25	15.85	16.13	16.50	16.85	17.12	17.44
.9995	13.84	14.50	15.06	15.55	15.99	16.35	16.70	16.98	17.35	17.55
.9999	14.38	14.94	15.44	16.12	16.50	16.87	17.12	17.35	17.62	17.87

12.2. Single Pulse with Known Amplitude and Unknown Phase

In this case, the retuned radar signal comprises a sinusoid of a deterministic amplitude and random phase whose *pdf* is uniform over the interval $\{0, 2\pi\}$. The output of the matched filter receiver that employs an envelope detector is denoted by $v(t)$ (see Fig. 12.2), and it can be written as a bandpass random process as

$$\begin{aligned} v(t) &= v_I(t)\cos\omega_0 t + v_Q(t)\sin\omega_0 t = r(t)\cos(\omega_0 t - \Phi(t)) \\ v_I(t) &= r(t)\cos\Phi(t) \\ v_Q(t) &= r(t)\sin\Phi(t) \end{aligned} \qquad \text{Eq. (12.20)}$$

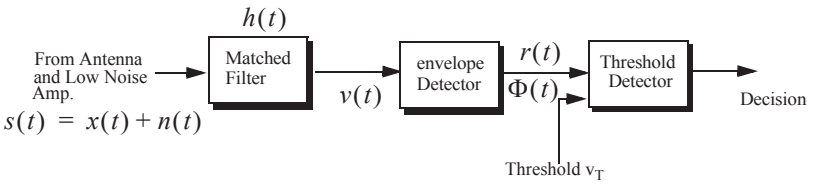


Figure 12.2. Simplified matched filter receiver employing an envelope detector.

$$r(t) = \sqrt{[v_I(t)]^2 + [v_Q(t)]^2}$$
$$\Phi(t) = \left[\tan\left(\frac{v_Q(t)}{v_I(t)}\right)\right]^{-1} \qquad \textbf{Eq. (12.21)}$$

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, and 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

$$\begin{aligned} H_1 &\Leftrightarrow s(t) = x(t) + n(t) \qquad and \qquad r(t) > v_T \Rightarrow Detection \\ H_0 &\Leftrightarrow s(t) = n(t) \qquad\quad and \qquad r(t) > v_T \Rightarrow False\ \ alarm \end{aligned} \qquad \textbf{Eq. (12.22)}$$

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. The matched filter output is a complex random variable that comprises either noise alone or noise plus target returns (i.e., sine wave of amplitude A and random phase). The quadrature components corresponding to the case of noise alone are

$$v_I(t) = n_I(t)$$
$$v_Q(t) = n_Q(t) \qquad \textbf{Eq. (12.23)}$$

where the noise quadrature components $n_I(t)$ and $n_Q(t)$ are uncorrelated zero mean lowpass Gaussian noise with equal variances, σ^2. In the second case the quadrature components are

$$\begin{aligned} v_I(t) = A + n_I(t) = r(t)\cos\Phi(t) \Rightarrow n_I(t) &= r(t)\cos\Phi(t) - A \\ v_Q(t) = n_Q(t) &= r(t)\sin\Phi(t) \end{aligned} \qquad \textbf{Eq. (12.24)}$$

The joint probability density function (*pdf*) of the two random variables $n_I; n_Q$ is

$$f_{n_I n_Q}(n_I, n_Q) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{n_I^2 + n_Q^2}{2\sigma^2}\right) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(r\cos\varphi - A)^2 + (r\sin\varphi)^2}{2\sigma^2}\right), \qquad \textbf{Eq. (12.25)}$$

which can be written as

$$f_{n_I n_Q}(n_I, n_Q) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(r\cos\varphi - A)^2 + (r\sin\varphi)^2}{2\sigma^2}\right). \qquad \textbf{Eq. (12.26)}$$

The *pdfs* of the random variables $r(t)$ and $\Phi(t)$, respectively, represent the modulus and phase of $v(t)$. The joint *pdf* for the two random variables $r(t);\Phi(t)$ are derived using a similar approach to that developed in Chapter 11. More precisely,

$$f_{R\Phi}(r, \varphi) = f_{n_I n_Q}(n_I, n_Q)|\mathbf{J}| \qquad \text{Eq. (12.27)}$$

where $|\mathbf{J}|$ is determinant of the matrix of derivatives $\mathbf{J}$ and referred to as the Jacobian. The matrix of derivatives is given by

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial n_I}{\partial r} & \dfrac{\partial n_I}{\partial \varphi} \\ \dfrac{\partial n_Q}{\partial r} & \dfrac{\partial n_Q}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -r\sin\varphi \\ \sin\varphi & r\cos\varphi \end{bmatrix}. \qquad \text{Eq. (12.28)}$$

It follows that the Jacobian is

$$|\mathbf{J}| = r(t). \qquad \text{Eq. (12.29)}$$

Substituting Eq. (12.25) and Eq. (12.29) into Eq. (12.27) and collecting terms yields

$$f_{R\Phi}(r, \varphi) = \frac{r}{2\pi\sigma^2} \exp\left(-\frac{r^2 + A^2}{2\sigma^2}\right) \exp\left(\frac{rA\cos\varphi}{\sigma^2}\right). \qquad \text{Eq. (12.30)}$$

The *pdf* for $r(t)$ alone is obtained by integrating Eq. (12.30) over φ. That is,

$$f_R(r) = \int_0^{2\pi} f_{R\Phi}(r, \varphi)d\varphi = \frac{r}{\sigma^2}\exp\left(-\frac{r^2 + A^2}{2\sigma^2}\right) \frac{1}{2\pi}\int_0^{2\pi} \exp\left(\frac{rA\cos\varphi}{\sigma^2}\right)d\varphi \qquad \text{Eq. (12.31)}$$

where the integral inside Eq. (12.31) is known as the modified Bessel function of zero order,

$$I_0(\beta) = \frac{1}{2\pi}\int_0^{2\pi} e^{\beta\cos\theta}\, d\theta. \qquad \text{Eq. (12.32)}$$

Thus,

$$f_R(r) = \frac{r}{\sigma^2} I_0\left(\frac{rA}{\sigma^2}\right) \exp\left(-\frac{r^2 + A^2}{2\sigma^2}\right), \qquad \text{Eq. (12.33)}$$

which is the Rician probability density function. The case when $A/\sigma^2 = 0$ (noise alone) was analyzed in Chapter 11 and the resulting *pdf* is a Rayleigh probability density function

$$f_R(r) = \frac{r}{\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right). \qquad \text{Eq. (12.34)}$$

When (A/σ^2) is very large, Eq. (12.33) becomes a Gaussian probability density function of mean A and variance σ^2:

$$f_R(r) \approx \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(r-A)^2}{2\sigma^2}\right). \qquad \text{Eq. (12.35)}$$

Figure 12.3 shows plots for the Rayleigh and Gaussian densities. The density function for the random variable Φ is obtained from

$$f_{\Phi}(\varphi) = \int_0^r f_{R\Phi}(r, \varphi)\ dr\,. \qquad \textbf{Eq. (12.36)}$$

While the detailed derivation is left as an exercise, the result is

$$f_{\Phi}(\varphi) = \frac{1}{2\pi}\ \exp\left(\frac{-A^2}{2\sigma^2}\right) + \frac{A\cos\varphi}{\sqrt{2\pi\sigma^2}}\ \exp\left(\frac{-(A\sin\varphi)^2}{2\sigma^2}\right)\ F\left(\frac{A\cos\varphi}{\sigma}\right) \qquad \textbf{Eq. (12.37)}$$

where

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}}\ e^{-\xi^2/2}\ d\xi\,. \qquad \textbf{Eq. (12.38)}$$

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. (12.37) collapses to a uniform *pdf* over the interval $\{0, 2\pi\}$. One excellent approximation for the function $F(x)$ is

$$F(x) = 1 - \left(\frac{1}{0.661x + 0.339\sqrt{x^2 + 5.51}}\right)\frac{1}{\sqrt{2\pi}}e^{-x^2/2} \qquad x \geq 0\,. \qquad \textbf{Eq. (12.39)}$$

and for negative values of x

$$F(-x) = 1 - F(x)\,. \qquad \textbf{Eq. (12.40)}$$

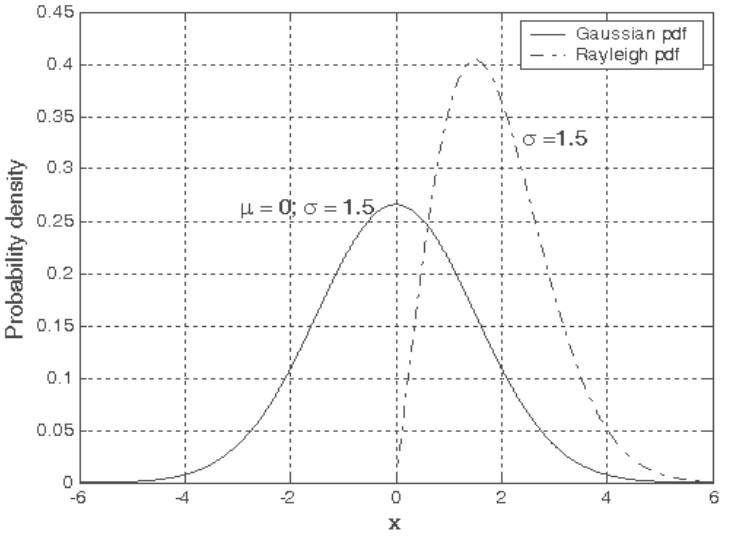


Figure 12.3. Gaussian and Rayleigh probability densities.

MATALAB Function "que_func.m"

The MATLAB function *"que_function.m"* calculates Eq.(12.38) using the approximation in Eqs. (12.39) and (12.40). Its syntax is as follows:

```
fofx = que_func(x).
```

12.2.1. 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:

$$P_{fa} = \int_{v_T}^{\infty} \frac{r}{\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right) dr = \exp\left(\frac{-v_T^2}{2\sigma^2}\right) \quad \textbf{Eq. (12.41)}$$

$$v_T = \sqrt{2\sigma^2 \ln\left(\frac{1}{P_{fa}}\right)} \,. \quad \textbf{Eq. (12.42)}$$

Figure 12.4 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. The false alarm time T_{fa} is related to the probability of false alarm by

$$T_{fa} = \frac{t_{int}}{P_{fa}} \quad \textbf{Eq. (12.43)}$$

where t_{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_{int}, by using the right-hand side of Eq. (12.41) and Eq. (12.42), one can rewrite T_{fa} as

$$T_{fa} = \frac{1}{B} \exp\left(\frac{v_T^2}{2\sigma^2}\right) \quad \textbf{Eq. (12.44)}$$

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

The false alarm number is defined as

$$n_{fa} = \frac{-\ln(2)}{\ln(1 - P_{fa})} \approx \frac{\ln(2)}{P_{fa}} \,. \quad \textbf{Eq. (12.45)}$$

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. (12.45), 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. (12.45) is always assumed. Hence, a clear distinction is made between Marcum's definition of the false alarm number and the definition in Eq. (12.45).

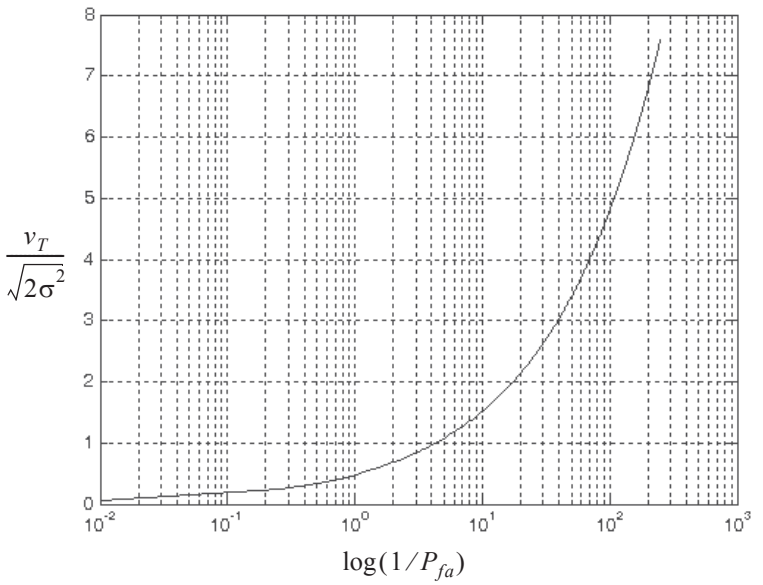


Figure 12.4. Normalized detection threshold versus probability of false alarm.

12.2.2. 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,

$$P_D = \int_{v_T}^{\infty} \frac{r}{\sigma^2} I_0\left(\frac{rA}{\sigma^2}\right) \exp\left(-\frac{r^2 + A^2}{2\sigma^2}\right) dr . \qquad \textbf{Eq. (12.46)}$$

Assuming that the radar signal is a sinusoid of amplitude A (completely known), then its power is $A^2/2$. Now, by using $SNR = A^2/2\sigma^2$ (single-pulse SNR) and $(v_T^2/2\sigma^2) = \ln(1/P_{fa})$, then Eq. (12.46) can be rewritten as

$$P_D = \int_{\sqrt{2\sigma^2 \ln(1/p_{fa})}}^{\infty} \frac{r}{\sigma^2} I_0\left(\frac{rA}{\sigma^2}\right) \exp\left(-\frac{r^2 + A^2}{2\sigma^2}\right) dr = Q\left[\sqrt{\frac{A^2}{\sigma^2}}, \sqrt{2\ln\left(\frac{1}{P_{fa}}\right)}\right] \qquad \textbf{Eq. (12.47)}$$

where

$$Q[a, b] = \int_{b}^{\infty} \zeta I_0(a\zeta) e^{-(\zeta^2 + a^2)/2} \, d\zeta . \qquad \textbf{Eq. (12.48)}$$

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. (12.47) can be approximated by

$$P_D \approx F\left(\frac{A}{\sigma} - \sqrt{2\ln\left(\frac{1}{P_{fa}}\right)}\right). \quad \textbf{Eq. (12.49)}$$

$F(x)$ is given by Eq. (12.38). Many approximations for Eq. (12.47) can be found throughout the literature. One very accurate approximation presented by North (1963) is given by

$$P_D \approx 0.5 \times erfc(\sqrt{-\ln P_{fa}} - \sqrt{SNR + 0.5}) \quad \textbf{Eq. (12.50)}$$

where the complementary error function was defined in Eq. (12.17).

The integral given in Eq. (12.47) is complicated and can be computed using numerical integration techniques. Parl[1] developed an excellent algorithm to numerically compute this integral. It is summarized as follows:

$$Q[a, b] = \begin{Bmatrix} \frac{\alpha_n}{2\beta_n}\exp\left(\frac{(a-b)^2}{2}\right) & a < b \\ 1 - \left(\frac{\alpha_n}{2\beta_n}\exp\left(\frac{(a-b)^2}{2}\right)\right) & a \geq b \end{Bmatrix} \quad \textbf{Eq. (12.51)}$$

$$\alpha_n = d_n + \frac{2n}{ab}\alpha_{n-1} + \alpha_{n-2} \quad \textbf{Eq. (12.52)}$$

$$\beta_n = 1 + \frac{2n}{ab}\beta_{n-1} + \beta_{n-2} \quad \textbf{Eq. (12.53)}$$

$$d_{n+1} = d_n d_1 \quad \textbf{Eq. (12.54)}$$

$$\alpha_0 = \begin{Bmatrix} 1 & a < b \\ 0 & a \geq b \end{Bmatrix} \quad \textbf{Eq. (12.55)}$$

$$d_1 = \begin{Bmatrix} a/b & a < b \\ b/a & a \geq b \end{Bmatrix}. \quad \textbf{Eq. (12.56)}$$

$\alpha_{-1} = 0.0$, $\beta_0 = 0.5$, and $\beta_{-1} = 0$. The recursive Eq. (12.51) through Eq. (12.56) 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.

MATLAB Function "marcumsq.m"

The MATLAB function *"marcumsq.m"* implements Parl's algorithm to calculate the probability of detection defined in Eq. (12.47). The syntax is as follows:

Pd = marcumsq (a, b)

1. Parl, S., A New Method of Calculating the Generalized Q Function, *IEEE Trans. Information Theory,* Vol. IT-26, January 1980, pp. 121-124.

where

Symbol	Description	Units	Status
a	A/σ	*dB*	*input*
b	$\sqrt{-2\ln(P_{fa})}$	*none*	*input*
Pd	*signal pulse probability of detection*	*none*	*output*

Figure 12.5 shows plots of the probability of detection, P_D, versus the single pulse SNR, with the P_{fa} as a parameter using this MATLAB function. This figure can be reproduced using MATLAB program *"Fig12_5.m,"* listed in Appendix 12-A.

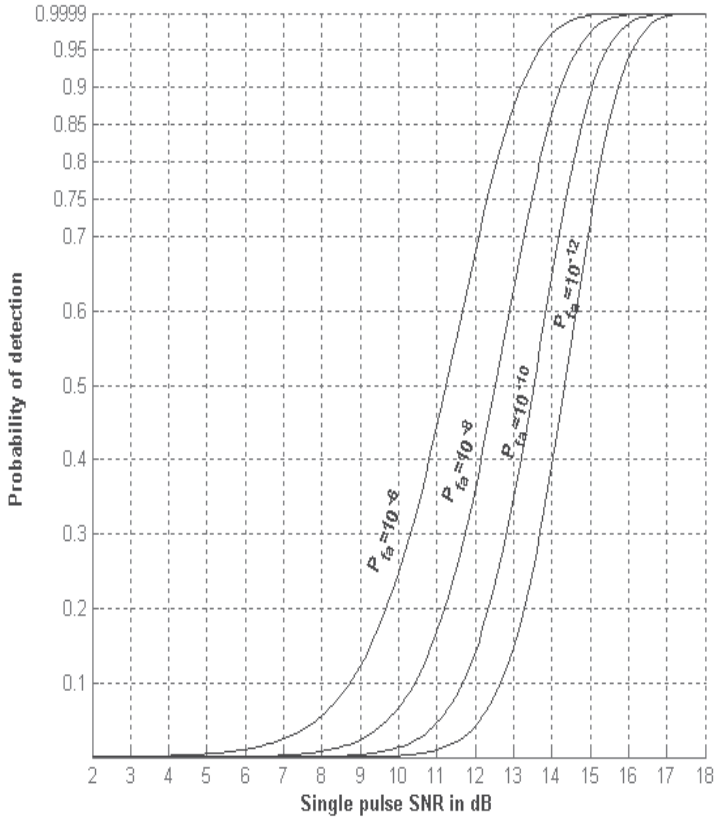


Figure 12.5. Probability of detection versus single pulse SNR, for several values of P_{fa}.

Problems

12.1. Prove the results given in Eqs. (12.9) through (12.12).

12.2. Derive Eq. (12.19).

12.3. Consider the matched filter receiver shown in Fig. 12.1. Develop expressions for the single pulse of known parameters probability of detection P_D and probability of false alarm P_{fa}

12.4. In the case of noise alone, the quadrature components of a radar return are independent Gaussian random variables with zero mean and variance σ^2. Assume that the radar processing consists of envelope detection followed by threshold decision. (a) Write an expression for the *pdf* of the envelope; (b) determine the threshold V_T as a function of σ that ensures a probability of false alarm $P_{fa} \leq 10^{-8}$.

12.5. A pulsed radar has the following specifications: time of false alarm $T_{fa} = 10min$, probability of detection $P_D = 0.95$, operating bandwidth $B = 1MHz$. (a) What is the probability of false alarm P_{fa}? (b) What is the single pulse SNR?

12.6. Show that when computing the probability of detection at the output of an envelope detector, it is possible to use Gaussian probability approximation when the SNR is very large.

12.7. A radar system uses a threshold detection criterion. The probability of false alarm is $P_{fa} = 10^{-10}$. (a) What must be the average SNR at the input of a linear detector so that the probability of miss is $P_m = 0.15$? Assume a large SNR approximation. (b) Write an expression for the *pdf* at the output of the envelope detector.

12.8. An X-band radar has the following specifications: received peak power $10^{-10}W$, probability of detection $P_D = 0.95$, time of false alarm $T_{fa} = 8min$, pulse width $\tau = 2\mu s$, operating bandwidth $B = 2MHz$, operating frequency $f_0 = 10GHz$, and detection range $R = 100Km$. Assume single pulse processing. (a) Compute the probability of false alarm P_{fa}. (b) Determine the SNR at the output of the matched filter. (c) At what SNR would the probability of detection drop to 0.9 (P_{fa} does not change)? (d) What is the increase in range that corresponds to this drop in the probability of detection?

12.9. Using the equation

$$P_D = 1 - e^{-SNR} \int_{P_{fa}}^{1} I_0(\sqrt{-4SNR \ln u}) du .$$

calculate P_D when $SNR = 10dB$ and $P_{fa} = 0.01$. Perform the integration numerically.

Appendix 12-A: Chapter 12 MATLAB Code Listings

The MATLAB code provided in this chapter was designed as an academic standalone tool and is not adequate for other purposes. The code was written in a way to assist the reader in gaining a better understanding of the theory. The code was not developed, nor is it intended to be used as part of an open-loop or a closed-loop simulation of any kind. The MATLAB code found in this textbook can be downloaded from this book's web page on the CRC Press website. Simply use your favorite web browser, go to www.crcpress.com, and search for keyword *"Mahafza"* to locate this book's web page.

MATLAB Function "que_func.m" Listing

```
function fofx = que_func(x)
% This function computes the value of the Q-function
% It uses the approximation in Eqs. (12.39) and (12.40)
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
```

MATLAB Function "marcumsq.m" Listing

```
function PD = marcumsq (a,b)
% This function uses Parl's method to compute PD
% Inputs
   % a    == sqrt(2.0 * 10^(.1*snr))
   % b    == sqrt(-2.0 * log(10^(-nfa)));
%%Output
   % PD  == single pulse probability of detection
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_1;
  betan = 1.0 + ratio .* n .* betan0 + betan_1;
  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
```

MATLAB Program "Fig12_5.m" Listing

```
% This program is used to produce Fig. 12.5
close all
clear all
for nfa = 6:2:12
  b = sqrt(-2.0 * log(10^(-nfa)));
  index = 0;
  hold on
  for snr = 2:.1:18
    index = index +1;
    a = sqrt(2.0 * 10^(.1*snr));
    pro(index) = marcumsq(a,b);
  end
  x = 2:.1:18;
  set(gca,'ytick',[.1 .2 .3 .4 .5 .6  .7 .75 .8 .85 .9 .95 .9999])
  set(gca,'xtick',[2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18])
   plot(x, pro,'k');
end
hold off
xlabel ('\bfSingle pulse SNR in dB'); ylabel ('\bfProbability of detection')
grid on
gtext('\bfP_f_a=10^-^6','rotation',65)
gtext('\bfP_f_a=10^-^8','rotation', 68)
gtext('\bfP_f_a=10^-^1^0','rotation', 70)
gtext('\bfP_f_a=10^-^1^2','rotation', 72)
```

Chapter 13

Detection of Fluctuating Targets

13.1. Introduction

In the previous chapter target detection was introduced in the context of single pulse detection with completely known (i.e., deterministic) amplitude and phase in one case, and known amplitude with random phase in another. The underlying assumption was that radar targets were made of non-varying (non-fluctuating) scatterers. However, in practice that it is rarely the case. First, one would expect the radar to receive multiple returns (pulses) from any given target in its field of view. Furthermore, real-world targets will fluctuate over the duration of a single pulse or from pulse to pulse. This chapter extends the analysis of Chapter 12 to account for target fluctuation as well as for target detection where multiple returned pulses are taken into consideration. Multiple returned pulses can be integrated (combined) coherently or non-coherently. The process of combining radar returns from many pulses is called radar pulse integration. Pulse integration can be performed on the quadrature components prior to the envelope detector. This is called coherent integration or predetection integration. Coherent integration preserves the phase relationship between the received pulses. Thus a buildup in the signal amplitude is expected. Alternatively, pulse integration performed after the envelope detector (where the phase relation is lost) is called noncoherent or post-detection integration, and a buildup in the signal amplitude is guaranteed.

13.2. Pulse Integration

Combining the returns from all pulses returned by a given target during a single scan is very likely to increase the radar sensitivity (i.e., SNR). The number of returned pulses from a given target depends on the antenna scan rate, the antenna beamwidth, and the radar PRF. More precisely, the number of pulses returned from a given target is given by

$$n_P = \frac{\theta_a T_{sc} f_r}{2\pi} \qquad \text{Eq. (13.1)}$$

where θ_a is the azimuth antenna beamwidth, T_{sc} is the scan time, and f_r is the radar PRF. The number of reflected pulses may also be expressed as

$$n_P = \frac{\theta_a f_r}{\dot{\theta}_{scan}} \qquad \text{Eq. (13.2)}$$

where $\dot{\theta}_{scan}$ is the antenna scan rate in degrees per second. Note that when using Eq. (13.1), θ_a is expressed in radians, while when using Eq. (13.2), it is expressed in degrees. As an example, consider a radar with an azimuth antenna beamwidth $\theta_a = 3°$, antenna scan rate $\dot{\theta}_{scan} = 45°/\text{sec}$ (antenna scan time, $T_{sc} = 8\,\text{sec}$), and a PRF $f_r = 300Hz$. Using either Eq. (13.1) or Eq. (13.2) yields $n_P = 20$ pulses.

As it was described in Chapter 2, pulse integration will very likely improve the receiver SNR. Nonetheless, caution should be exercised when attempting to account for how much SNR is attained through pulse integration. This is true for the following reasons: First, during a antenna scan, a given target will not always be located at the center of the radar beam (i.e., have maximum gain). In fact, during a scan, a target will first enter the antenna beam at the $3dB$ point, reach maximum gain, and finally leave the beam at the $3dB$ point again. Thus, the returns do not have the same amplitude even though the target RCS may be constant and all other factors that may introduce signal loss remain the same.

Other factors that may introduce further variation to the amplitude of the returned pulses include target RCS and propagation path fluctuations. Additionally, when the radar employs a very fast scan rate, an additional loss term is introduced due to the motion of the beam between transmission and reception. This is referred to as scan loss. A distinction should be made between scan loss due to a rotating antenna (which is described here) and the term scan loss that is normally associated with phased array antennas (which takes on a different meaning in that context).

Finally, since coherent integration utilizes the phase information from all integrated pulses, it is critical that any phase variation between all integrated pulses be known with a great level of confidence. Consequently, target dynamics (such as target range, range rate, tumble rate, RCS fluctuation) must be estimated or computed accurately so that coherent integration can be meaningful. In fact, if a radar coherently integrates pulses from targets without proper knowledge of the target dynamics, it suffers a loss in SNR rather than the expected SNR buildup. Knowledge of target dynamics is not as critical when employing noncoherent integration; nonetheless, target range rate must be estimated so that only the returns from a given target within a specific range bin are integrated. In other words, one must avoid range walk (i.e., having a target cross between adjacent range bins during a single scan).

A comprehensive analysis of pulse integration should also take into account issues such as the probability of detection P_D, probability of false alarm P_{fa}, the target statistical fluctuation model, and the noise or interference of statistical models. This is the subject of the rest of this chapter.

13.2.1. Coherent Integration

In coherent integration, and when a perfect integrator is used (100% efficiency) to integrate n_P pulses, the SNR is improved by the same factor. Otherwise, integration loss occurs, which is always the case for noncoherent integration. Coherent integration loss occurs when the integration process is not optimum. This could be due to target fluctuation, instability in the radar local oscillator, or propagation path changes.

Denote the single pulse SNR required to produce a given probability of detection as $(SNR)_1$. The SNR resulting from coherently integrating n_P pulses is then given by

$$(SNR)_{CI} = n_P(SNR)_1 . \qquad \textbf{Eq. (13.3)}$$

Coherent integration cannot be applied over a long interval of time, particularly if the target RCS is varying rapidly. If the target radial velocity is known and no acceleration is assumed, the maximum coherent integration time is limited to

$$t_{CI} = \sqrt{\frac{\lambda}{2a_r}} \qquad \textbf{Eq. (13.4)}$$

where λ is the radar wavelength and a_r is the target radial acceleration. Coherent integration time can be extended if the target radial acceleration can be compensated for by the radar.

In order to demonstrate the improvement in the SNR using coherent integration, consider the case where the radar return signal contains both signal plus additive noise. The mth pulse is

$$y_m(t) = s(t) + n_m(t) \qquad \textbf{Eq. (13.5)}$$

where $s(t)$ is the radar signal return of interest and $n_m(t)$ is white uncorrelated additive noise signal with variance σ^2. Coherent integration of n_P pulses yields

$$z(t) = \frac{1}{n_P}\sum_{m=1}^{n_P} y_m(t) = \sum_{m=1}^{n_P}\frac{1}{n_P}[s(t)+n_m(t)] = s(t) + \sum_{m=1}^{n_P}\frac{1}{n_P}n_m(t) \qquad \textbf{Eq. (13.6)}$$

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

$$\sigma^2_{n_P} = E\left[\left(\sum_{m=1}^{n_P}\frac{1}{n_P}n_m(t)\right)\left(\sum_{l=1}^{n_P}\frac{1}{n_P}n_l(t)\right)^*\right] \qquad \textbf{Eq. (13.7)}$$

where E is the expected value operator. It follows that

$$\sigma^2_{n_P} = \frac{1}{n_P^2}\sum_{m,l=1}^{n_P} E[n_m(t)n_l{}^*(t)] = \frac{1}{n_P^2}\sum_{m,l=1}^{n_P}\sigma^2_{ny}\delta_{ml} = \frac{1}{n_P}\sigma^2_{ny} \qquad \textbf{Eq. (13.8)}$$

where σ^2_{ny} is the single pulse noise power and δ_{ml} is equal to zero for $m \neq l$ and unity for $m = l$. Observation of Eqs. (13.6) and (13.8) indicates 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.

13.2.2. Noncoherent Integration

When the phase of the integrated pulses is not known, so that coherent integration is no longer possible, another form of pulse integration is done. In this case, pulse integration is performed by adding (integrating) the individual pulses' envelopes or the square of their envelopes. Thus, the term noncoherent integration is adopted. A block diagram of a radar receiver utilizing noncoherent integration is illustrated in Fig. 13.1.

The performance difference (measured in SNR) between the linear envelope detector and the quadratic (square law) detector is practically negligible. Robertson (1967) showed that this dif-

ference is typically less than $0.2dB$; he showed that the performance difference is higher than $0.2dB$ only for cases where $n_P > 100$ and $P_D < 0.01$. Both of these conditions are of no practical significance in radar applications. It is much easier to analyze and implement the square law detector in real hardware than is the case for the envelope detector. Therefore, most authors make no distinction between the type of detector used when referring to noncoherent integration, and the square law detector is almost always assumed. The analysis presented in this book will always assume, unless indicated otherwise, noncoherent integration using the square law detector.

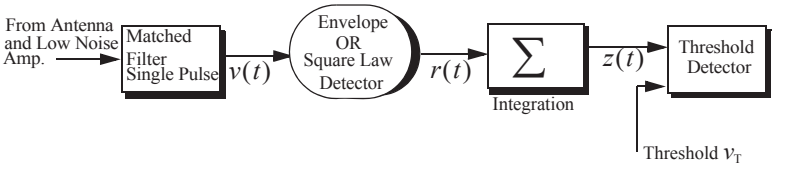


Figure 13.1. Simplified block diagram of a radar detector when noncoherent integration is used.

13.2.3. Improvement Factor and Integration Loss

Noncoherent integration is less efficient than coherent integration. Actually, the noncoherent integration gain is always smaller than the number of noncoherently integrated pulses. This loss in integration is referred to as post-detection or square-law detector loss.

Define $(SNR)_{NCI}$ as the SNR required to achieve a specific P_D given a particular P_{fa} when n_P pulses are integrated noncoherently. Also denote the single pulse SNR as $(SNR)_1$. It follows that

$$(SNR)_{NCI} = (SNR)_1 \times I(n_P) \qquad \textbf{Eq. (13.9)}$$

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 (1998) as

$$[I(n_P)]_{dB} = 6.79(1 + 0.253P_D)\left(1 + \frac{\log(1/P_{fa})}{46.6}\right)\log(n_P) \quad . \qquad \textbf{Eq. (13.10)}$$
$$(1 - 0.140\log(n_P) + 0.018310(\log n_P)^2)$$

The integration loss in *dB* is defined as

$$[L_{NCI}]_{dB} = 10\log n_P - [I(n_P)]_{dB}. \qquad \textbf{Eq. (13.11)}$$

MATLAB Function "impmrov_fact.m"

The function *"improv_fac.m"* calculates the improvement factor using Eq. (13.10). The syntax is as follows:

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

where

Symbol	Description	Units	Status
np	*number of integrated pulses*	*none*	*input*
pfa	*probability of false alarms*	*none*	*input*
pd	*probability of detection*	*none*	*input*
impr_of_np	*improvement factor*	*output*	*dB*

Figure 13.2 shows plots of the improvement factor versus the number of integrated pulses using different combinations of P_D and P_{fa}. The top part of Fig. 13.2 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. (13.10). While, the lower part of Fig. 13.2 shows plots of the corresponding integration loss versus n_P with P_D and P_{fa} as parameters. This figure can be reproduced using the MATLAB program *"Fig13_2.m,"* listed in Appendix 13-B.

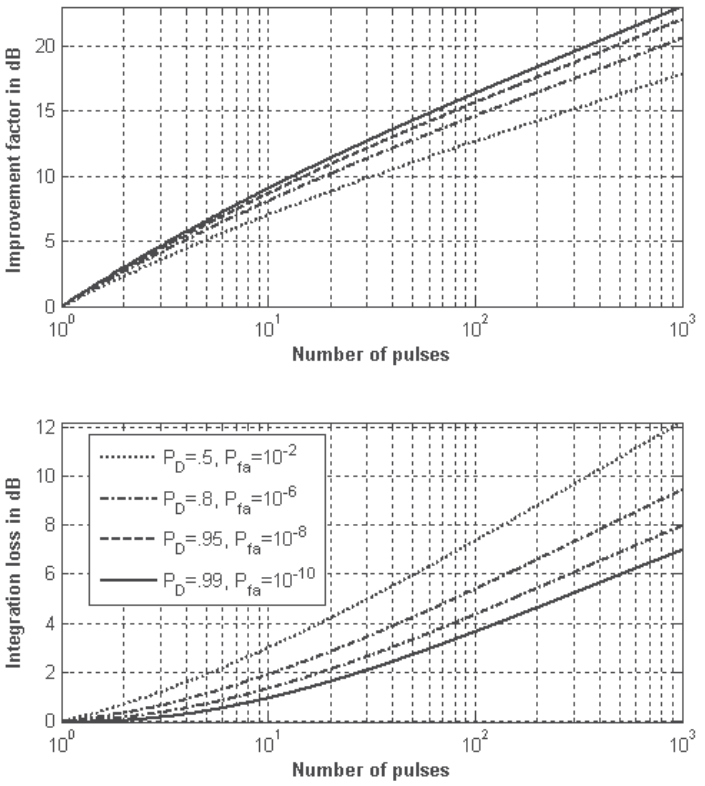


Figure 13.2. Typical plots for the improvement factor and integration loss versus number of noncoherently integrated pulses.

13.3. Target Fluctuation: The Chi-Square Family of Targets

Target detection utilizing the square law detector was first analyzed by Marcum[1], where he assumed a constant RCS (nonfluctuating target). This work was extended by Swerling[2] to four distinct cases of target RCS fluctuation. 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 as Swerling V. Target fluctuation introduces an additional loss factor in the SNR as compared to the case where fluctuation is not present, given the same P_D and P_{fa}.

Swerling V targets have constant amplitude over one antenna scan or observation interval; 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 from scan to scan 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. Noncoherent 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

$$f_X(x) = \frac{N}{(N-1)!\ \sqrt{\sigma_x^2}} \left(\frac{Nx}{\sigma_x}\right)^{N-1} \exp\left(-\frac{Nx}{\sigma_x}\right) \qquad \textbf{Eq. (13.12)}$$

where σ_x is the standard deviation for the 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,

$$f_X(x) = \frac{1}{\sigma_x} \exp\left(-\frac{x}{\sigma_x}\right) \qquad x \geq 0\ . \qquad \textbf{Eq. (13.13)}$$

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

$$f_X(x) = \frac{4x}{\sigma_x^2} \exp\left(-\frac{2x}{\sigma_x}\right) \qquad x \geq 0\ . \qquad \textbf{Eq. (13.14)}$$

1. Marcum, J. I., A Statistical Theory of Target Detection by Pulsed Radar, *IRE Transactions on Information Theory,* Vol IT-6, pp. 59-267, April 1960.
2. Swerling, P., Probability of Detection for Fluctuating Targets, *IRE Transactions on Information Theory*, Vol IT-6, pp. 269-308, April 1960.

13.4. Probability of False Alarm Formulation for a Square Law Detector

Computation of the general formula for the probability of false alarm P_{fa} and subsequently the rest of square law detection theory requires knowledge and a good understating of the incomplete Gamma function. Hence, those readers who are not familiar with this function are advised to read Appendix 13-A before proceeding with the rest of this chapter.

DiFranco and Rubin[1] derived a general form relating the threshold and P_{fa} for any number of pulses when noncoherent integration is used. The square law detector under consideration is shown in Fig. 13.3. There are $n_P \geq 2$ pulses integrated noncoherently and the noise power (variance) is σ^2.

The complex envelope in terms of the quadrature components is given by

$$\tilde{r}(t) = r_I(t) + jr_Q(t), \qquad \textbf{Eq. (13.15)}$$

thus the square of the complex envelope is

$$|\tilde{r}(t)|^2 = r_I^2(t) + r_Q^2(t). \qquad \textbf{Eq. (13.16)}$$

The samples $|\tilde{r}_k|^2$ are computed from the samples of $\tilde{r}(t)$ evaluated at $t = t_k;\ k = 1, 2, \ldots, n_P$. It follows that

$$Z = \frac{1}{2\sigma^2}\sum_{k=1}^{n_P}[r_I^2(t_k) + r_Q^2(t_k)]. \qquad \textbf{Eq. (13.17)}$$

The random variable Z is the sum of $2n_P$ squares of random variables, each of which is a Gaussian random variable with variance σ^2. Thus, using the analysis developed in Chapter 3, the *pdf* for the random variable Z is given by

$$f_Z(z) = \left\{ \begin{array}{ll} \dfrac{z^{n_P-1}e^{-z}}{\Gamma(n_P)} & z \geq 0 \\ 0 & z < 0 \end{array} \right\}. \qquad \textbf{Eq. (13.18)}$$

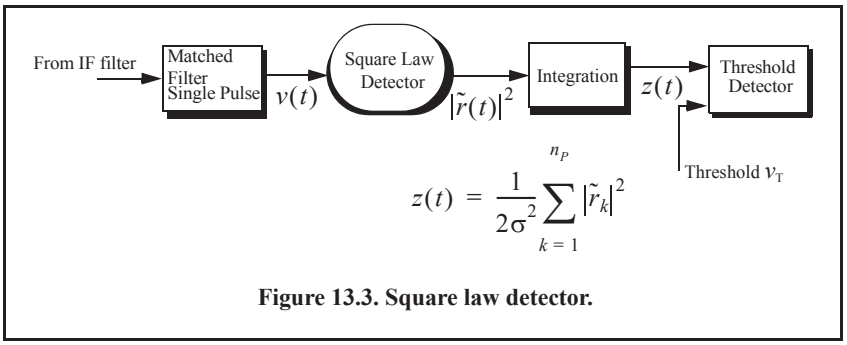


Figure 13.3. Square law detector.

1. DiFranco, J. V. and Rubin, W. L., *Radar Detection*, Artech House, Norwood, MA 1980.

Consequently, the probability of false alarm given a threshold value v_T is

$$P_{fa} = Prob\{Z \geq v_T\} = \int_{v_T}^{\infty} \frac{z^{n_P - 1} e^{-z}}{\Gamma(n_P)} \, dz \,. \qquad \text{Eq. (13.19)}$$

and using analysis provided in Appendix 13-A yields

$$P_{fa} = 1 - \Gamma_I\left(\frac{v_T}{\sqrt{n_P}}, n_P - 1\right). \qquad \text{Eq. (13.20)}$$

Using the algebraic expression for the incomplete Gamma function, Eq. (13.20) can be written as

$$P_{fa} = e^{-v_T} \sum_{k=0}^{n_P - 1} \frac{v_T^k}{k!} = 1 - e^{-v_T} \sum_{k=n_P}^{\infty} \frac{v_T^k}{k!} \,. \qquad \text{Eq. (13.21)}$$

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

$$v_{T,m} = v_{T,m-1} - \frac{G(v_{T,m-1})}{G'(v_{T,m-1})} \qquad ; \; m = 1, 2, 3, \ldots \qquad \text{Eq. (13.22)}$$

The iteration is terminated when $\left|v_{T,m} - v_{T,m-1}\right| < v_{T,m-1} / 10000.0$. The functions G and G' are

$$G(v_{T,m}) = (0.5)^{n_P / n_{fa}} - \Gamma_I(v_T, n_P) \qquad \text{Eq. (13.23)}$$

$$G'(v_{T,m}) = -\frac{e^{-v_T} v_T^{n_P - 1}}{(n_P - 1)!} \,. \qquad \text{Eq. (13.24)}$$

The initial value for the recursion is

$$v_{T,0} = n_P - \sqrt{n_P} + 2.3 \sqrt{-\log P_{fa}} \; (\sqrt{-\log P_{fa}} + \sqrt{n_P} - 1) \,. \qquad \text{Eq. (13.25)}$$

MATLAB Function "threshold.m"

The function *"threshold.m"* calculates the threshold value given the algorithm described in this section. The syntax is as follows:

[pfa, vt] = threshold (nfa, np)

where

Symbol	Description	Units	Status
nfa	*number of false alarm*	*none*	*input*
np	*number of pulses*	*none*	*input*
pfa	*probability of alarm*	*none*	*output*
vt	*threshold value*	*none*	*output*

Figure 13.4 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 the following MATLAB program, *"Fig13_4.m,"* listed in Appendix 13-B.

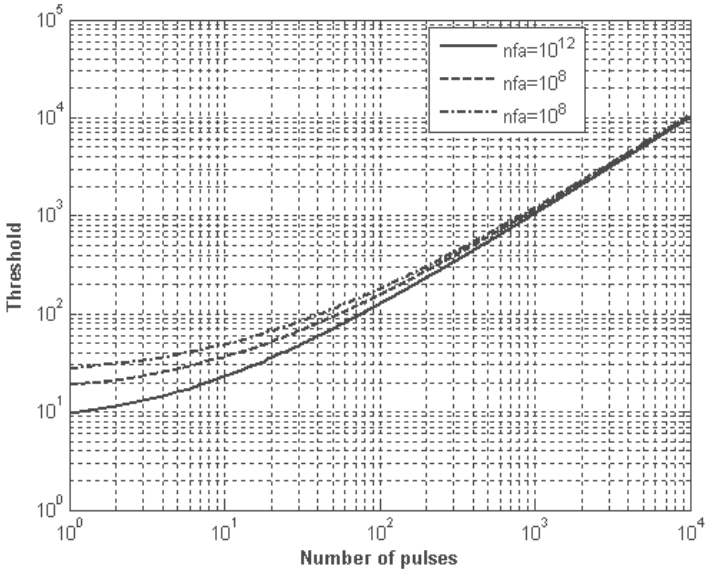


Figure 13.4. Threshold v_T versus n_p for several values of n_{fa}.

13.4.1. Square Law Detection

The *pdf* for the linear envelope $r(t)$ was derived in Chapter 12. Define a new dimensionless variable y as

$$y_n = r_n/\sigma \qquad \textbf{Eq. (13.26)}$$

where the subscript *n* denotes the nth pulse. Also define

$$\Re_p = A^2/\sigma^2 = 2SNR. \qquad \textbf{Eq. (13.27)}$$

σ^2 is the noise variance. It follows that the *pdf* for the new variable is

$$f_{Y_n}(y_n) = f_{R_n}(r_n)\left|\frac{dr_n}{dy_n}\right| = y_n\ I_0(y_n\sqrt{\Re_p})\ \exp\left(\frac{-(y_n^2+\Re_p)}{2}\right). \qquad \textbf{Eq. (13.28)}$$

The output of a square law detector for the nth pulse is proportional to the square of its input. Thus, it is convenient to define a new change variable,

$$z_n = \frac{1}{2}y_n^2. \qquad \textbf{Eq. (13.29)}$$

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

$$f_{Z_n}(x_n) = f(y_n)\left|\frac{dy_n}{dz_n}\right| = \exp\left(-\left(z_n + \frac{\Re_p}{2}\right)\right) I_0(\sqrt{2z_n\Re_p}) . \quad \textbf{Eq. (13.30)}$$

Noncoherent integration of n_p pulses is implemented as

$$z = \sum_{n=1}^{n_P} \frac{1}{2} y_n^2 . \quad \textbf{Eq. (13.31)}$$

Again, $n_P \geq 2$. Since the random variables y_n are independent, the *pdf* for the variable z is

$$f(z) = f((y_1) \otimes f(y_2) \otimes \ldots \otimes f(y_{n_p})) . \quad \textbf{Eq. (13.32)}$$

The operator $\otimes$ symbolically indicates convolution. The characteristic functions for the individual *pdf*s can then be used to compute the joint *pdf* for Eq. (13.32). The result is

$$f_Z(z) = \left(\frac{2z}{n_P\Re_p}\right)^{(n_P-1)/2} \exp\left(-z - \frac{1}{2}n_P\Re_p\right) I_{n_P-1}(\sqrt{2n_Pz\Re_p}) . \quad \textbf{Eq. (13.33)}$$

I_{n_P-1} is the modified Bessel function of order $n_P - 1$. Substituting Eq. (13.27) into (13.33) yields

$$f_Z(z) = \left(\frac{z}{n_P SNR}\right)^{(n_P-1)/2} e^{(-z-n_P SNR)} I_{n_P-1}(2\sqrt{n_P z SNR}) . \quad \textbf{Eq. (13.34)}$$

When target fluctuation is not present (referred to as Swerling 0 or Swerling V target), the probability of detection is obtained by integrating $f_Z(z)$ from the threshold value to infinity. The probability of false alarm is obtained by letting $\Re_p$ be zero and integrating the *pdf* from the threshold value to infinity. More specifically,

$$P_D\big|_{SNR} = \int_{v_T}^{\infty} \left(\frac{z}{n_P SNR}\right)^{(n_P-1)/2} e^{(-z-n_P SNR)} I_{n_P-1}(2\sqrt{n_P z SNR})dz , \quad \textbf{Eq. (13.35)}$$

which can be rewritten as

$$P_D\big|_{SNR} = e^{-n_P SNR}\left(\sum_{k=0}^{\infty}\frac{(n_P SNR)^k}{k!}\right)\left(\sum_{j=0}^{n_P-1+k}\frac{e^{-v_T}\, v_T^j}{j!}\right) . \quad \textbf{Eq. (13.36)}$$

Alternatively, when target fluctuation is present, the *pdf* is calculated using the conditional probability density function of Eq. (13.35) with respect to the SNR value of the target fluctuation type. In general, given a fluctuating target with SNR^F, where the superscript indicates fluctuation, the expression for the probability of detection is

$$P_D\big|_{SNR^F} = \int_0^{\infty} P_D\big|_{SNR}\, f_Z(z^F/SNR^F)dz = \quad . \quad \textbf{Eq. (13.37)}$$

$$\int_0^{\infty} P_D\big|_{SNR}\left(\frac{z^F}{n_P SNR^F}\right)^{(n_P-1)/2} e^{(-z^F-n_P SNR^F)} I_{n_P-1}(2\sqrt{n_P z^F SNR^F})dz$$

Remember that target fluctuation introduces an additional loss term in the SNR. It follows that for the same P_D given the same P_{fa} and the same n_P, $SNR^F > SNR$. One way to calculate this additional SNR is to first compute the required SNR given no fluctuation, then add to it the amount of target fluctuation loss to get the required value for SNR^F. How to calculate this fluctuation loss will be addressed later on in this chapter. Meanwhile, hereinafter, the superscript $\{^F\}$ will be dropped and it will always be assumed.

13.5. Probability of Detection Calculation

Marcum defined the probability of false alarm for the case when $n_P > 1$ as

$$P_{fa} \approx \ln(2)\left(\frac{n_P}{n_{fa}}\right). \qquad \textbf{Eq. (13.38)}$$

The single pulse probability of detection for nonfluctuating targets was derived in Chapter 12. When $n_P > 1$, the probability of detection is computed using the Gram-Charlier series. In this case, the probability of detection is

$$P_D \cong \frac{erfc(V/\sqrt{2})}{2} - \frac{e^{-V^2/2}}{\sqrt{2\pi}}[C_3(V^2-1) + C_4V(3-V^2) - C_6V(V^4-10V^2+15)] \qquad \textbf{Eq. (13.39)}$$

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

$$V = \frac{v_T - n_P(1+SNR)}{\varpi}. \qquad \textbf{Eq. (13.40)}$$

In general, values for C_3, C_4, C_6, and ϖ vary depending on the target fluctuation type.

13.5.1. Detection of Swerling 0 (Swerling V) Targets

For Swerling 0 (Swerling V) target fluctuations, the probability of detection is calculated using Eq. (13.39). In this case, the Gram-Charlier series coefficients are

$$C_3 = -\frac{SNR + 1/3}{\sqrt{n_p}(2SNR+1)^{1.5}} \qquad \textbf{Eq. (13.41)}$$

$$C_4 = \frac{SNR + 1/4}{n_p(2SNR+1)^2} \qquad \textbf{Eq. (13.42)}$$

$$C_6 = C_3^2/2 \qquad \textbf{Eq. (13.43)}$$

$$\varpi = \sqrt{n_p(2SNR+1)}. \qquad \textbf{Eq. (13.44)}$$

MATLAB Function "pd_swerling5.m"

The function *"pd_swerling5.m"* calculates the probability of detection for Swerling 0 targets. The syntax is as follows:

[pd] = pd_swerling5 (input1, indicator, np, snr)

where

Symbol	Description	Units	Status
input1	P_{fa} or n_{fa}	*none*	*input*
indicator	*1 when input1* = P_{fa} *2 when input1* = n_{fa}	*none*	*input*
np	*number of integrated pulses*	*none*	*input*
snr	*SNR*	*dB*	*input*
pd	*probability of detection*	*none*	*output*

Figure 13.5 shows a plot for the probability of detection versus SNR for cases $n_p = 1, 10$. Note that it requires less SNR, with ten pulses integrated noncoherently, 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 $I(10) \approx 8.55dB$. Figure 13.5 shows that the ten pulse SNR is about $6.03dB$. Therefore, the single pulse SNR is about $14.5dB$, which can be read from the figure. This figure can be reproduced using MATLAB program *"Fig13_5.m,"* listed in Appendix 13-B.

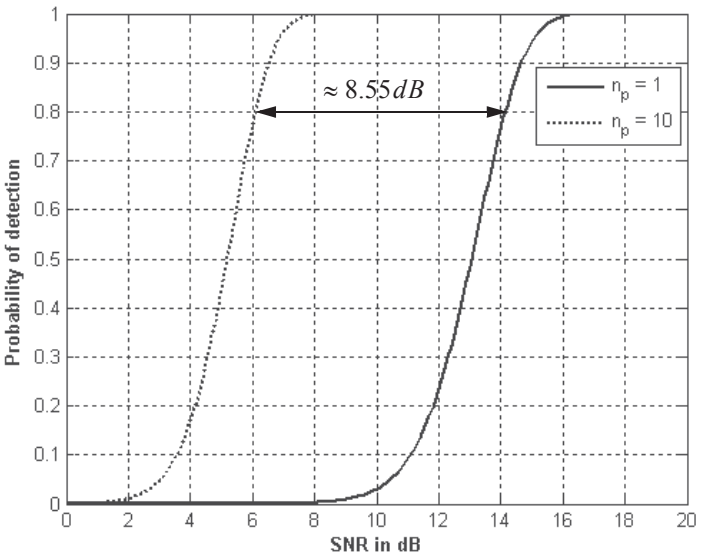


Figure 13.5. Probability of detection versus SNR, $P_{fa} = 10^{-9}$, and noncoherent integration; Swerling 0.

13.5.2. Detection of Swerling I Targets

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

$$P_D = e^{-(v_T)/(1+SNR)} \quad ; \; n_P = 1 \qquad \textbf{Eq. (13.45)}$$

$$P_D = 1 - \Gamma_I(v_T, n_P - 1) + \left(1 + \frac{1}{n_P SNR}\right)^{n_p - 1} \Gamma_I\left(\frac{v_T}{1 + \frac{1}{n_P SNR}}, n_P - 1\right) \qquad \textbf{Eq. (13.46)}$$

$$\times \; e^{-v_T/(1+n_P SNR)} \quad ; \; n_P > 1.$$

MATLAB Function "pd_swerling1.m"

The function *"pd_swerling1.m"* calculates the probability of detection for Swerling I type targets. The syntax is as follows:

[pd] = pd_swerling1 (nfa, np, snr)

where

Symbol	Description	Units	Status
nfa	*Marcum's false alarm number*	*none*	*input*
np	*number of integrated pulses*	*none*	*input*
snr	*SNR*	*dB*	*input*
pd	*probability of detection*	*none*	*output*

Figure 13.6 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 (Swerling 0) 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 the MATLAB program *"Fig13_6.m,"* listed in Appendix 13-B. Figure 13.7 is similar to Fig. 13.6, except in this case $P_{fa} = 10^{-6}$ and $n_P = 5$. This figure can be reproduced using the following MATLAB program, *"Fig13_7.m,"* listed in Appendix 13-B.

13.5.3. Detection of Swerling II Targets

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

$$P_D = 1 - \Gamma_I\left(\frac{v_T}{(1+SNR)}, n_p\right) \quad ; \; n_P \le 50. \qquad \textbf{Eq. (13.47)}$$

For the case when $n_P > 50$ the probability of detection is computed using the Gram-Charlier series. In this case,

$$C_3 = -\frac{1}{3\sqrt{n_p}} \quad , \; C_6 = \frac{C_3^2}{2} \qquad \textbf{Eq. (13.48)}$$

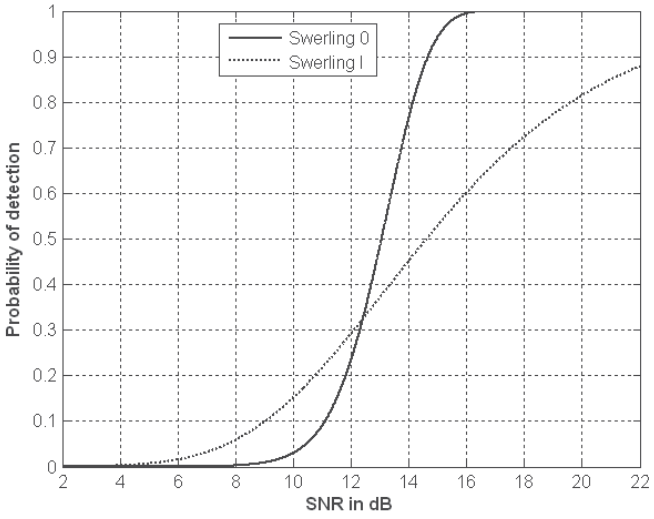


Figure 13.6. Probability of detection versus SNR, single pulse. $P_{fa} = 10^{-9}$.

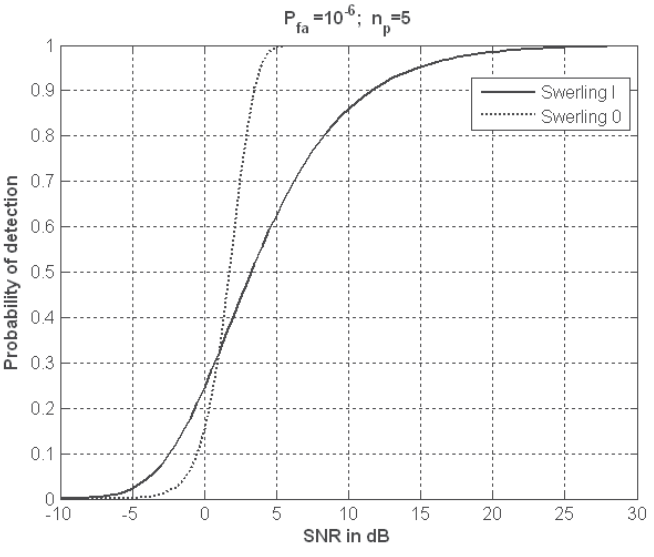


Figure 13.7. Probability of detection versus SNR. Swerling I and Swerling 0.

$$C_4 = \frac{1}{4n_P} \qquad \textbf{Eq. (13.49)}$$

$$\varpi = \sqrt{n_P}\ (1 + SNR)\,. \qquad \textbf{Eq. (13.50)}$$

MATLAB Function "pd_swerling2.m"

The function *"pd_swerling2.m"* calculates P_D for Swerling II type targets. The syntax is as follows:

[pd] = pd_swerling2 (nfa, np, snr)

where

Symbol	Description	Units	Status
nfa	*Marcum's false alarm number*	*none*	*input*
np	*number of integrated pulses*	*none*	*input*
snr	*SNR*	*dB*	*input*
pd	*probability of detection*	*none*	*output*

Figure 13.8 shows a plot of the probability of detection for Swerling 0, Swerling I, and Swerling II with $n_P = 5$, where $P_{fa} = 10^{-7}$. Figure 13.9 is similar to Fig. 13.8 except in this case $n_P = 2$. Both figures can be reproduced, respectively, using the MATLAB programs *"Fig13_8.m"* and *"Fig13_9.m,"* listed in Appendix 13-B.

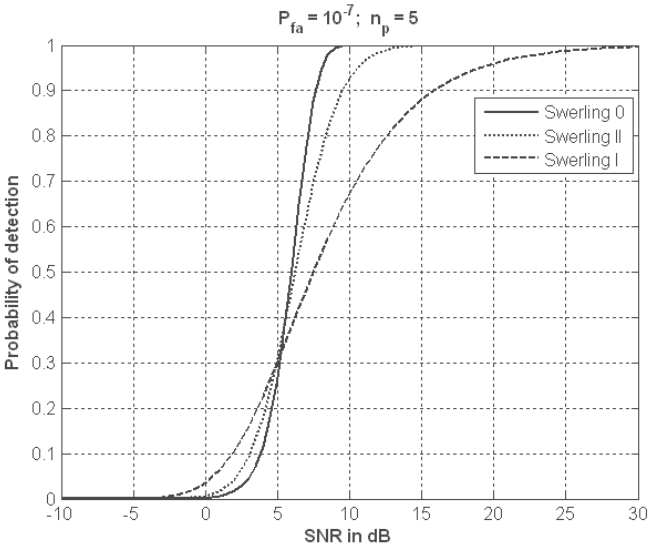


Figure 13.8. Probability of detection versus SNR. Swerling II, Swerling I, and Swerling 0.

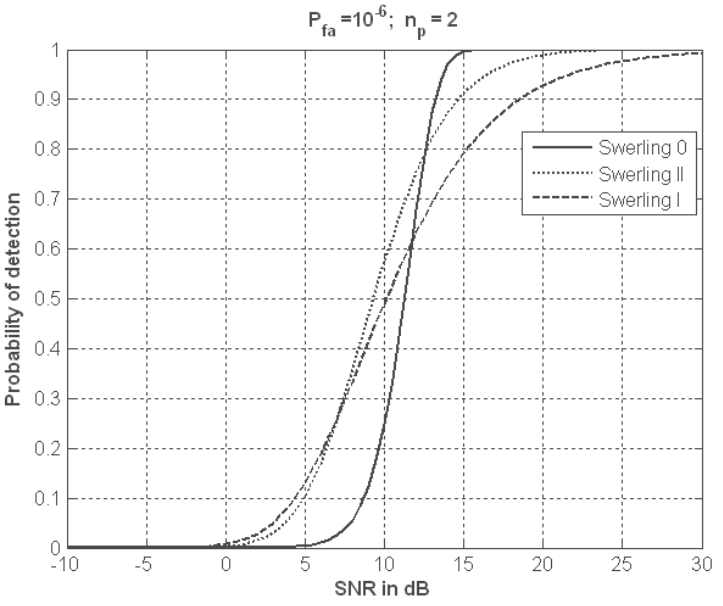


Figure 13.9. Probability of detection versus SNR. Swerling II, Swerling I, and Swerling 0.

13.5.4. Detection of Swerling III Targets

The exact formulas, developed by Marcum, for the probability of detection for Swerling III type targets when $n_P = 1, 2$

$$P_D = \exp\left(\frac{-v_T}{1 + n_P SNR/2}\right)\left(1 + \frac{2}{n_P SNR}\right)^{n_P - 2} \times K_0$$

$$K_0 = 1 + \frac{v_T}{1 + n_P SNR/2} - \frac{2}{n_P SNR}(n_P - 2) \qquad \textbf{Eq. (13.51)}$$

For $n_P > 2$ the expression is

$$P_D = \frac{v_T^{n_P - 1} e^{-V_T}}{(1 + n_P SNR/2)(n_P - 2)!} + 1 - \Gamma_I(v_T, n_P - 1) + K_0 \quad . \qquad \textbf{Eq. (13.52)}$$

$$\times \ \Gamma_I\left(\frac{v_T}{1 + 2/n_p SNR}, n_p - 1\right)$$

MATLAB Function "pd_swerling3.m"

The function *"pd_swerling3.m"* calculates P_D for Swerling III type targets. The syntax is as follows:

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

where

Symbol	Description	Units	Status
nfa	*Marcum's false alarm number*	*none*	*input*
np	*number of integrated pulses*	*none*	*input*
snr	*SNR*	*dB*	*input*
pd	*probability of detection*	*none*	*output*

Figure 13.10 shows a plot of the probability of detection as a function of SNR for $n_P = 1, 10, 50, 100$, where $P_{fa} = 10^{-9}$. Figure 13.11 shows a plot of the probability of detection for Swerling 0, Swerling I, Swerling II, and Swerling III with $n_P = 5$ and $P_{fa} = 10^{-7}$.

Notice that (see Fig. 13.11) as the target fluctuation becomes more rapid, as in the case of Swerling I type targets, it requires more SNR to achieve the same probability of detection when considering lesser fluctuating targets as in the case of Swerling 0, for example. Figures 13.10 and 13.11 can be reproduced, respectively, using the MATLAB programs *"Fig13_10.m"* and *"Fig13_11.m,"* listed in Appendix 13-B.

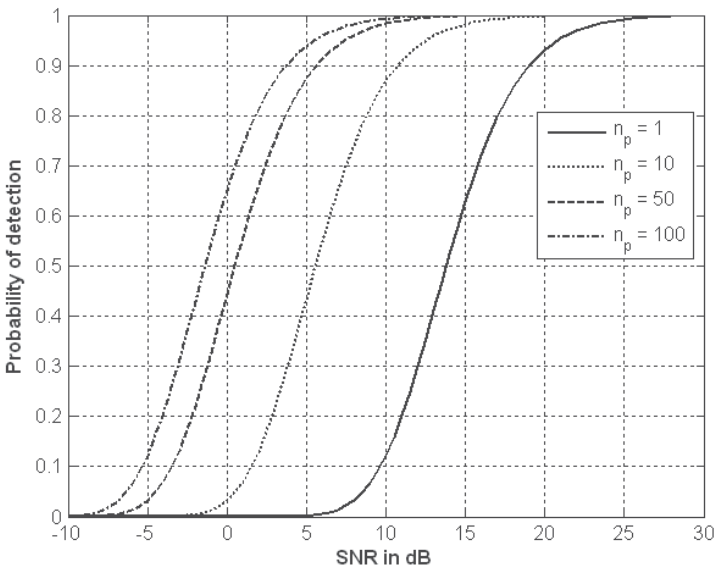


Figure 13.10. Probability of detection versus SNR. Swerling III. $P_{fa} = 10^{-9}$.

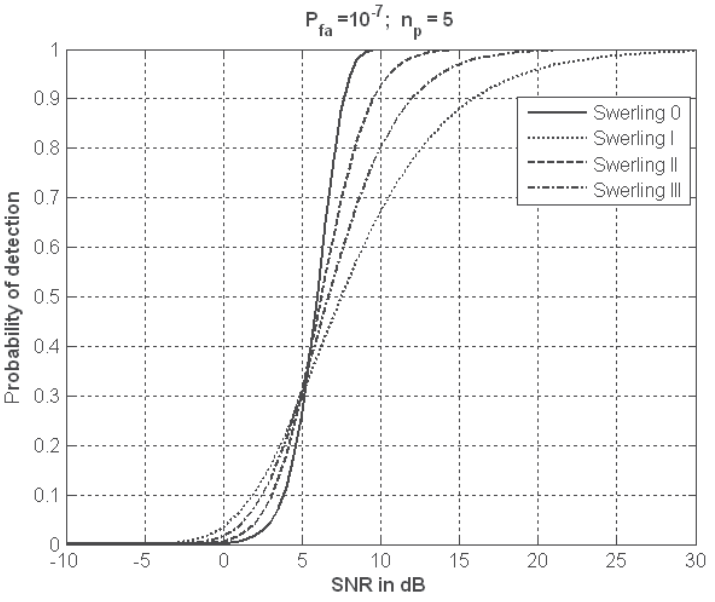


Figure 13.11. Probability of detection versus SNR. Swerling III, Swerling II, Swerling I, and Swerling 0.

13.5.5. Detection of Swerling IV Targets

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

$$P_D = 1 - \left[\gamma_0 + \left(\frac{SNR}{2}\right) n_P \gamma_1 + \left(\frac{SNR}{2}\right)^2 \frac{n_P(n_P-1)}{2!}\gamma_2 + \ldots + \left(\frac{SNR}{2}\right)^{n_P} \gamma_{n_P}\right] \Big/ \left(1 + \frac{SNR}{2}\right)^{-n_P} \quad \text{Eq. (13.53)}$$

$$\gamma_i = \Gamma_I\left(\frac{v_T}{1 + (SNR)/2}\ , n_P + i\right). \quad \text{Eq. (13.54)}$$

By using the recursive formula

$$\Gamma_I(x, i+1) = \Gamma_I(x, i) - \frac{x^i}{i! \exp(x)}, \quad \text{Eq. (13.55)}$$

only γ_0 needs to be calculated using Eq. (13.54), and the rest of γ_i are calculated from the following recursion:

$$\gamma_i = \gamma_{i-1} - A_i \qquad ;\ i > 0 \quad \text{Eq. (13.56)}$$

$$A_i = \frac{v_T/(1 + (SNR)/2)}{n_P + i - 1}\ A_{i-1} \qquad ;\ i > 1 \quad \text{Eq. (13.57)}$$

$$A_1 = \frac{(v_T/(1+(SNR)/2))^{n_P}}{n_P!\exp(v_T/(1+(SNR)/2))} \quad \textbf{Eq. (13.58)}$$

$$\gamma_0 = \Gamma_I\left(\frac{v_T}{(1+(SNR)/2)}\ , n_P\right). \quad \textbf{Eq. (13.59)}$$

For the case when $n_P \geq 50$, the Gram-Charlier series can be used to calculate the probability of detection. In this case,

$$C_3 = \frac{1}{3\sqrt{n_P}}\frac{2\beta^3-1}{(2\beta^2-1)^{1.5}} \quad ; \ C_6 = \frac{C_3^2}{2} \quad \textbf{Eq. (13.60)}$$

$$C_4 = \frac{1}{4n_P}\frac{2\beta^4-1}{(2\beta^2-1)^2} \quad \textbf{Eq. (13.61)}$$

$$\varpi = \sqrt{n_P(2\beta^2-1)} \quad \textbf{Eq. (13.62)}$$

$$\beta = 1+(SNR)/2\,. \quad \textbf{Eq. (13.63)}$$

MATLAB Function "pd_swerling4.m"

The function *"pd_swerling4.m"* calculates P_D for Swerling IV type targets. The syntax is as follows:

[pd] = pd_swerling4 (nfa, np, snr)

where

Symbol	Description	Units	Status
nfa	*Marcum's false alarm number*	*none*	*input*
np	*number of integrated pulses*	*none*	*input*
snr	*SNR*	*dB*	*input*
pd	*probability of detection*	*none*	*output*

Figure 13.12 shows plots of the probability of detection as a function of SNR for $n_P = 1, 10, 25, 75$, where $P_{fa} = 10^{-6}$. This figure can be reproduced using the MATLAB program *"Fig13_12.m,"* listed in Appendix 13-B.

13.6. Computation of the Fluctuation Loss

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. Kanter[1] developed an exact analysis for calculating the fluctuation loss. In this text, this author will take advantage of the computational power of MATLAB and the MATLAB functions developed in this text to numerically calculate the amount of fluctuation loss.

1. Kanter, I., Exact Detection Probability for Partially Correlated Rayleigh Targets, *IEEE Trans*, AES-22, pp. 184-196, March 1986.

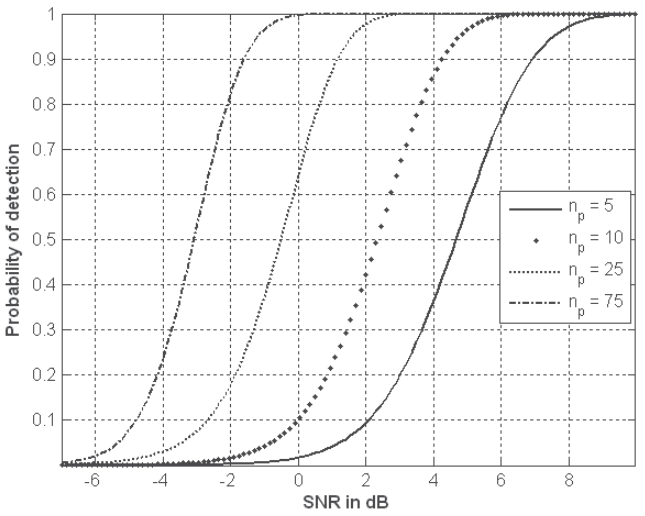


Figure 13.12. Probability of detection versus SNR. Swerling IV.

$P_{fa} = 10^{-6}$.

MATLAB Function "fluct.m"

To calculate the amount of fluctuation loss, the MATALB function "*fluct.m"* was developed. Its syntax is as follows:

[SNR] = fluct(pd, pfa, np, sw_case)

where

Symbol	Description	Units	Status
pd	*desired probability of detection*	*none*	*input*
nfa	*desired number of false alarms*	*none*	*input*
np	*number of pulses*	*none*	*input*
sw_case	*0, 1, 2, 3, or 4 depending on the desired Swerling case*	*none*	*input*
SNR	*Resulting SNR*	*dB*	*output*

For example, using the syntax

[SNR0] = fluct(0.8, 1e6, 5, 0)

will calculate the *SNR0* corresponding to a Swerling 0. If one would use this *SNR* in the function *"pd_swerling5.m"* with following syntax

[pd] = pd_swerling5 (1e6, 2, 5, SNR0),

the resulting P_D will be equal to 0.8 . Similarly, if the following syntax is used

[SNR1] = fluct(0.8, 1e-6, 5, 1),

then the value *SNR1* will be that of Swerling 1. Of course, if one would use this *SNR1* value in the function *"pd_swerling1.m"* with following syntax

[pd] = pd_swerling1(1e6, 5,0.8, SNR1),

the same P_D of 0.8 will be calculated. Therefore, the fluctuation loss for this case is equal to *SNR0 - SNR1*. Figure 13.13 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 *"Fig13_13.m,"* listed in Appendix 13-B.

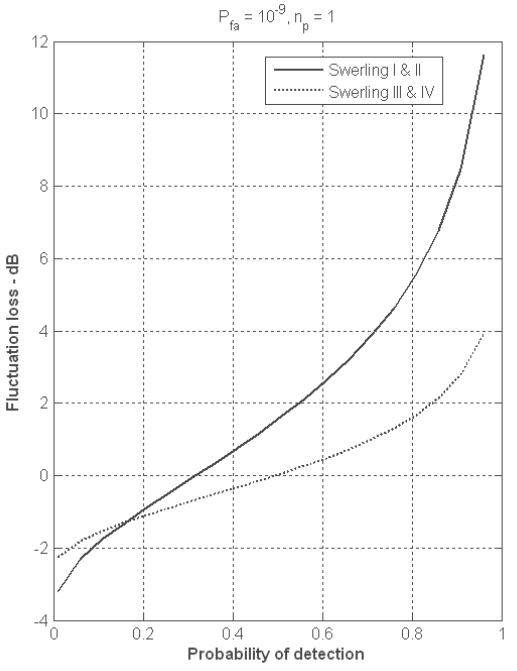


Figure 13.13. Fluctuation loss versus probability of detection.

13.7. 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

$$(SNR)_{R_0} = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 k T_0 B F L R_0^4} = 1 . \qquad \textbf{Eq. (13.64)}$$

The single pulse SNR at any range R is

$$SNR = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 k T_0 B F L R^4}. \qquad \text{Eq. (13.65)}$$

Dividing Eq. (13.165) by Eq. (13.64) yields

$$\frac{SNR}{(SNR)_{R_0}} = \left(\frac{R_0}{R}\right)^4. \qquad \text{Eq. (13.66)}$$

Therefore, if the range R_0 is known, then the SNR at any other range R is

$$(SNR)_{dB} = 40\log\left(\frac{R_0}{R}\right). \qquad \text{Eq. (13.67)}$$

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 nth frame is P_{D_n}; then, the cumulative probability of detecting the target at least once during the nth frame (see Fig. 13.14) is given by

$$P_{C_n} = 1 - \prod_{i=1}^{n} (1 - P_{D_i}). \qquad \text{Eq. (13.68)}$$

P_{D_1} is usually selected to be very small. Clearly, the probability of not detecting the target during the nth frame is $1 - P_{C_n}$. The probability of detection for the ith frame, P_{D_i}, is computed as discussed in the previous section.

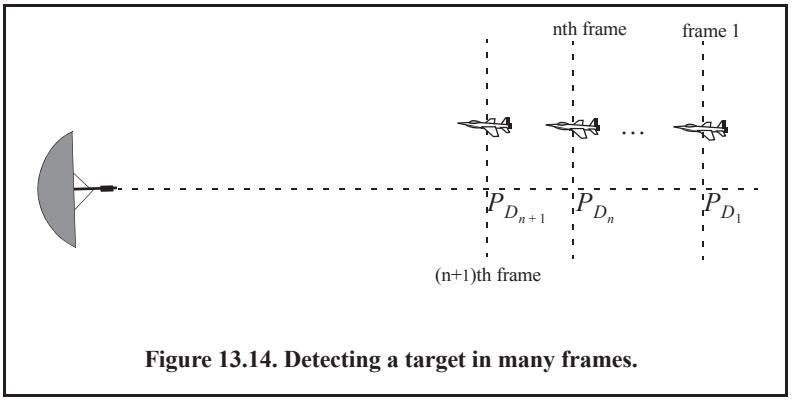


Figure 13.14. Detecting a target in many frames.

Example:

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 $R = 8Km$?

Solution:

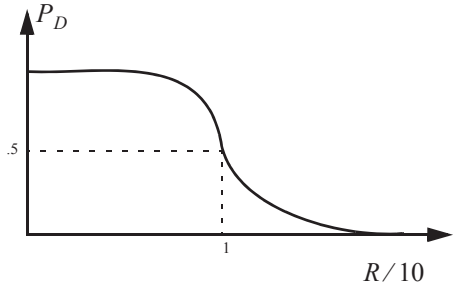
From the function "marcumsq.m," the SNR corresponding to $P_D = 0.5$ *and* $P_{fa} = 10^{-7}$ *is approximately* $12dB$. *By using a similar analysis to that which led to Eq. (13.67), we can express the SNR at any range* R *as*

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

By using the function "marcumsq.m," we can construct the following table:

R *Km*	(SNR) dB	P_D
2	39.09	0.999
4	27.9	0.999
6	20.9	0.999
8	15.9	0.999
9	13.8	0.9
10	12.0	0.5
11	10.3	0.25
12	8.8	0.07
14	6.1	0.01
16	3.8	ε
20	0.01	ε

where ε *is very small. A sketch of* P_D *versus normalized range is shown in the figure below.*



Cumulative probability of detection versus normalized range.

The cumulative probability of detection is given in Eq. (13.68), 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 $R = 16Km$. *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* $R = 8Km$.

frame	1	2	3	4	5	6	7	8	9
range in Km	16	15	14	13	12	11	10	9	8

The cumulative probability of detection at 8*Km is then*

$$P_{C_9} = 1-(1-0.999)(1-0.9)(1-0.5)(1-0.25)(1-0.07) \quad .$$
$$(1-0.01)(1-\varepsilon)^2 \approx 0.9998$$

13.8. Constant False Alarm Rate (CFAR)

The detection threshold is computed so that the radar receiver maintains a constant predetermined probability of false alarm. The relationship between the threshold value V_T and the probability of false alarm P_{fa} was derived in Chapter 12, and for convenience is repeated here as Eq. (13.69):

$$v_T = \sqrt{2\sigma^2 \ln\left(\frac{1}{P_{fa}}\right)} \,. \qquad \textbf{Eq. (13.69)}$$

If the noise power σ^2 is constant, then a fixed threshold can satisfy Eq. (13.69). 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 the 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 the analog Cell-Averaging CFAR (CA-CFAR) technique is examined. The analysis presented in this section closely follows Urkowitz[1].

13.8.1. Cell-Averaging CFAR (Single Pulse)

The CA-CFAR processor is shown in Fig. 13.15. 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

$$Y_1 \geq K_0 Z \,. \qquad \textbf{Eq. (13.70)}$$

CA-CFAR assumes that the target of interest is in the CUT and all reference cells contain zero-mean independent Gaussian noise of variance σ^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

1. Urkowitz, H., Decision and Detection Theory, unpublished lecture notes. Lockheed Martin Co., Moorestown, NJ.

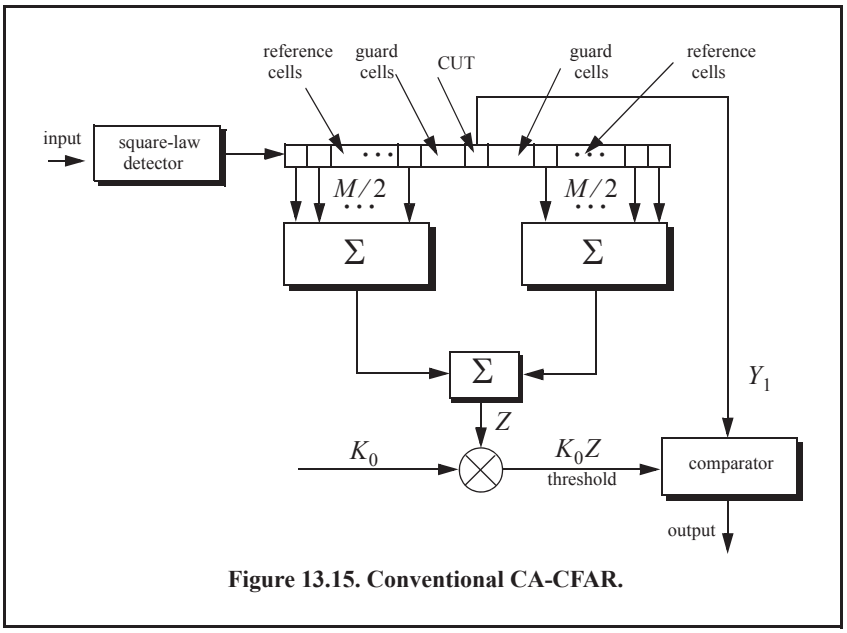


Figure 13.15. Conventional CA-CFAR.

$$f(z) = \frac{z^{(M/2)-1} e^{(-z/2\sigma^2)}}{2^{M/2}\ \sigma^M \Gamma(M/2)} \quad ;\ z > 0\,. \qquad \textbf{Eq. (13.71)}$$

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

$$P_{fa}(v_T = y) = e^{-y/2\sigma^2}\,. \qquad \textbf{Eq. (13.72)}$$

It follows that the unconditional probability of false alarm is

$$P_{fa} = \int_0^\infty P_{fa}(v_T = y) f(y)\, dy \qquad \textbf{Eq. (13.73)}$$

where $f(y)$ is the *pdf* of the threshold, which except for the constant K_0, is the same as that defined in Eq. (13.71). Therefore,

$$f(y) = \frac{y^{M-1} e^{(-y/2K_0\sigma^2)}}{(2K_0\sigma^2)^M \Gamma(M)} \quad ;\ y \geq 0\,. \qquad \textbf{Eq. (13.74)}$$

Performing the integration in Eq. (13.73) yields

$$P_{fa} = 1/(1 + K_0)^M\,. \qquad \textbf{Eq. (13.75)}$$

Observation of Eq. (13.75) shows that the probability of false alarm is now independent of the noise power, which is the objective of CFAR processing.

13.8.2. Cell-Averaging CFAR with Noncoherent Integration

In practice, CFAR averaging is often implemented after noncoherent integration, as illustrated in Fig. 13.16. 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,

$$P_{fa} = Prob\{Y_1/Z > K_1\}. \qquad \textbf{Eq. (13.76)}$$

Equation (12.76) 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

$$P_{fa} = \int_0^\infty P_{fa}(v_T = y) f(y)\, dy \qquad \textbf{Eq. (13.77)}$$

where $f(y)$ is the *pdf* of the threshold. In view of this, the probability density function describing the random variable $K_1 Z$ is given by

$$f(y) = \frac{(y/K_1)^{Mn_P - 1} e^{(-y/2K_1\sigma^2)}}{(2\sigma^2)^{Mn_P} K_1 \ \Gamma(Mn_P)} \quad ; \ y \geq 0. \qquad \textbf{Eq. (13.78)}$$

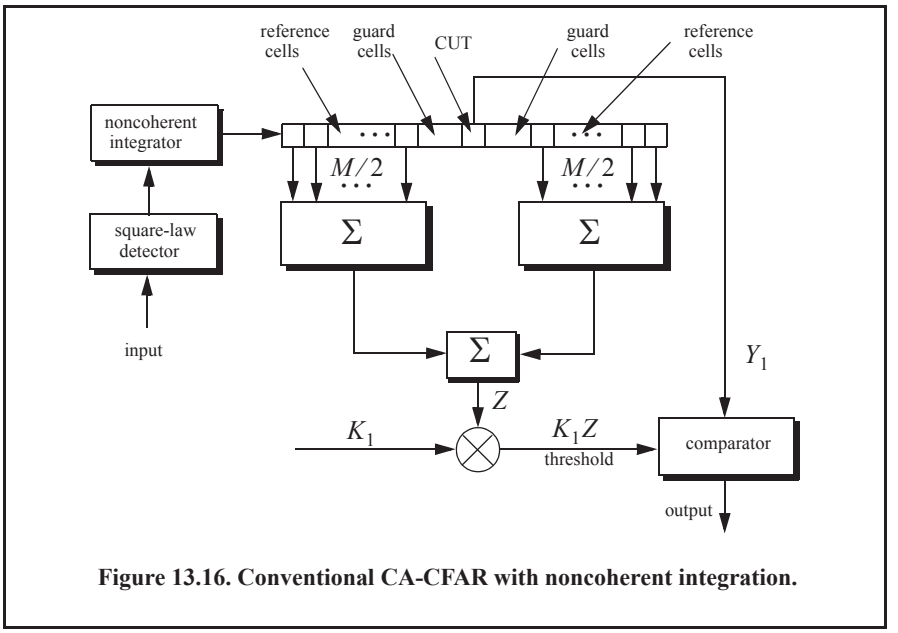


Figure 13.16. Conventional CA-CFAR with noncoherent integration.

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

$$P_{fa} = \frac{1}{(1+K_1)^{Mn_P}} \sum_{k=0}^{n_P-1} \frac{1}{k!} \frac{\Gamma(Mn_P+k)}{\Gamma(Mn_P)} \left(\frac{K_1}{1+K_1}\right)^k, \qquad \text{Eq. (13.79)}$$

which is identical to Eq. (13.75) when $K_1 = K_0$ and $n_P = 1$.

13.9. M-out-of-N Detection

A few sources in the literature refer to the *M-out-of-N* detection as *binary integration* and / or as *double threshold* detection; nonetheless, *M-out-of-N* is the most commonly used name. The basic idea behind the *M-out-of-N* detection technique is as follows: In any given resolution cell (range, Doppler, or angle) the detection process is repeated *N* times, where the outcome of each decision cycle is either a "detection" or "no detection," hence the term *binary* is used in the literature. For each decision cycle, the probability of detection and the probability of false alarm are computed. The final decision criterion declares a target detection if *M* out of *N* decision cycles have resulted in a detection. Clearly, the decision criterion associated with this technique follows a binomial distribution.

To elaborate further on this concept of detection, assume a non-fluctuating target whose single trial probability of detection is P_D and its probability of false alarm is P_{fa}. Denote the total probability of detection resulting from the *M-out-of-N* detection technique as P_{Dmn}. It follows that after *N* independent trials of detection one gets

$$P_{Dmn} = 1 - (1 - P_D)^N. \qquad \text{Eq. (13.80)}$$

Similarly, the probability of false alarm after the same number of trials is

$$P_{FA} = 1 - (1 - P_{fa})^N. \qquad \text{Eq. (13.81)}$$

For example, if the desired P_{Dmn} is *0.99*, then by using Eq. (13.80), one finds that a $P_D = 0.9$ will accomplish the desired P_{Dmn} after 2 trials (i.e., *N=2*); alternatively, when using a $P_D = 0.2$, it will take 20 trials to reach the desired P_{Dmn}. Furthermore, Eq. (13.80) implicitly indicates that as the number of trials increases so does P_{Dmn}, but this buildup in detection probability is somewhat costly. That is true because as the number of trials is increased, the overall probability of false alarm P_{FA} is also increased. Obviously, a very undesirable result (the proof is left as an exercise, see Problem 13.20).

A slight modification to the *M-out-of-N* detection process that guarantees an increase or buildup in P_{Dmn} while simultaneously keeping P_{FA} in check is as follows:

1. A specific P_{fa} value is chosen; typically it is a design constraint.
2. For each value M, compute the corresponding P_{FA} from Eq. (13.83).
3. Using any of the techniques developed in this book to calculate the threshold value V_T so that P_{fa} is maintained, compute its corresponding SNR.
4. Calculate P_D that corresponds to the SNR computed in step 3.
5. Use Eq. (13.82) to compute the probability of detection P_{Dmn}, and from any of the techniques developed in this book, compute the corresponding SNR so that the threshold value computed in step 3 is maintained, therefore, P_{Dmn} is also maintained.

6. Repeat for each M to establish the specific combination of M (i.e., yielding P_{FA}) so that the SNR is minimized for a given P_{Dmn}.

Following this modified approach, P_{Dmn} and P_{FA} are given by

$$P_{Dmn} = \sum_{k=M}^{N} C_k^N \; P_D^k \; (1-P_D)^{N-k} \qquad \text{Eq. (13.82)}$$

$$P_{FA} = \sum_{k=M}^{N} C_k^N \; P_{fa}^k \; (1-P_{fa})^{N-k} \qquad \text{Eq. (13.83)}$$

where

$$C_k^N = \frac{N!}{k!(N-k)!}. \qquad \text{Eq. (13.84)}$$

For small values of P_D, Eq. (13.82) keeps the overall detection probability P_{Dmn} to less than or equal to P_D. Alternatively, for larger values of P_D, a quick buildup in the value of P_{Dmn} occurs.

Selecting the specific combination of N and M that yields a desired P_{Dmn} is typically a design constraint. In any case, once the choice is made, one must take target fluctuating into account. In this case, the optimal value for M is

$$M_{opt} = 10^{\alpha} N^{\beta} \qquad \text{Eq. (13.85)}$$

where α and β are constants that vary depending on the target fluctuation type, Table 13.1 shows their values corresponding to different Swerling targets.

Table 13.1. Parameters of Eq. (13.85)

Fluctuation Type	α	β	Range of N
Swerling 0	0.8	-0.02	5-700
Swerling I	0.8	-0.02	6-700
Swerling II	0.91	-0.38	9-700
Swerling III	0.8	-0.02	6-700
Swerling IV	0.873	-0.27	10-700

13.10. The Radar Equation Revisited

The radar equation developed in Chapter 2 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

$$R^4 = \frac{P_{av} G_t G_r \lambda^2 \sigma I(n_P)}{(4\pi)^3 k T_o F B \tau f_r L_t L_f \; (SNR)_1} \qquad \text{(4.86)}$$

where $P_{av} = P_t \tau f_r$ is the average transmitted power, P_t is the peak transmitted power, τ is the pulse width, f_r is PRF, G_t is the transmitting antenna gain, G_r is the receiving antenna gain, λ is the wavelength, σ is the target cross section, $I(n_P)$ is the improvement factor, n_P is the number of integrated pulses, k is Boltzman's constant, T_o is 290 Kelvin, F is the system noise figure, B is the receiver bandwidth, L_t is the total system losses including integration loss, L_f is the loss due to target fluctuation, and $(SNR)_1$ is the minimum single pulse SNR required for detection.

Assuming that the radar parameters such as power, antenna gain, wavelength, losses, bandwidth, effective temperature, and noise figure are known, the steps one should follow to solve for range are shown in Fig. 13.17. Note that both sides of the bottom half of Fig. 13.17 are identical. Nevertheless, two paths are purposely shown so that a distinction between scintillating and nonfluctuating targets is made.

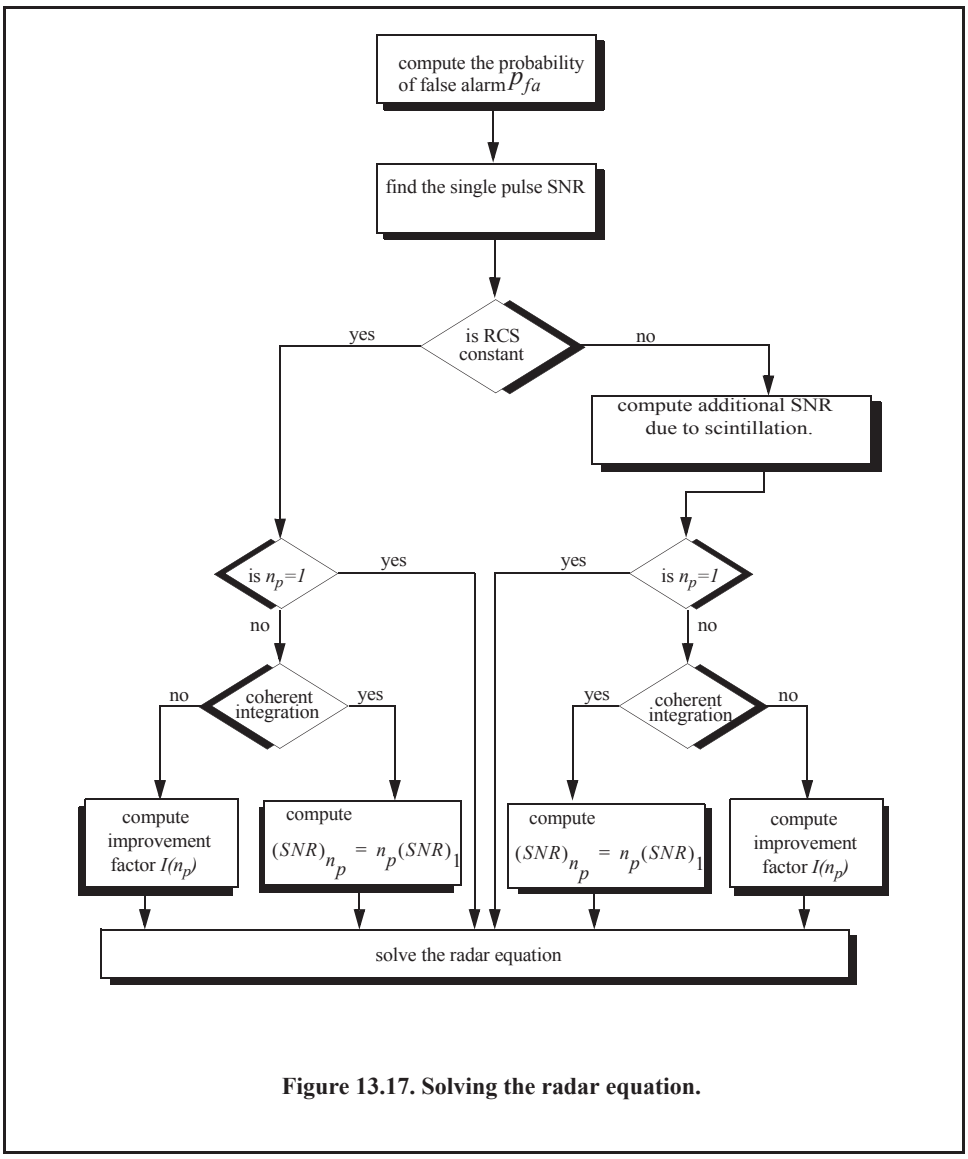


Figure 13.17. Solving the radar equation.

Problems

13.1. A pulsed radar has the following specifications: time of false alarm $T_{fa} = 10\ min$, probability of detection $P_D = 0.95$, operating bandwidth $B = 1MHz$. (a) What is the probability of false alarm P_{fa}? (b) What is the single pulse SNR? (c) Assuming noncoherent integration of 100 pulses, what is the SNR reduction so that P_D and P_{fa} remain unchanged?

13.2. An L-band radar has the following specifications: operating frequency $f_0 = 1.5GHz$, operating bandwidth $B = 2MHz$, noise figure $F = 8dB$, system losses $L = 4dB$, time of false alarm $T_{fa} = 12\ minutes$, detection range $R = 12Km$, probability of detection $P_D = 0.5$, antenna gain $G = 5000$, and target RCS $\sigma = 1m^2$. (a) Determine the PRF f_r, the pulse width τ, the peak power P_t, the probability of false alarm P_{fa}, 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 noncoherently? (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 $9Km$.

13.3. A certain radar utilizes 10 pulses for noncoherent integration. The single pulse SNR is $15dB$ and the probability of miss is $P_m = 0.15$. (a) Compute the probability of false alarm P_{fa}. (b) Find the threshold voltage V_T.

13.4. (a) Show how you can use the radar equation to determine the PRF f_r, the pulse width τ, the peak power P_t, the probability of false alarm P_{fa}, and the minimum detectable signal level S_{min}. Assume the following specifications: operating frequency $f_0 = 1.5MHz$, operating bandwidth $B = 1MHz$, noise figure $F = 10dB$, system losses $L = 5dB$, time of false alarm $T_{fa} = 20\ min$, detection range $R = 12Km$, probability of detection $P_D = 0.5$ (three pulses). (b) If post-detection integration is assumed, determine the SNR.

13.5. Consider a scanning low PRF radar. The antenna half-power beam width is $1.5°$, and the antenna scan rate is $35°$ per second. The pulse width is $\tau = 2\mu s$, and the PRF is $f_r = 400Hz$. (a) Compute the radar operating bandwidth. (b) Calculate the number of returned pulses from each target illumination. (c) Compute the SNR improvement due to post-detection integration (assume 100% efficiency). (d) Find the number of false alarms per minute for a probability of false alarm $P_{fa} = 10^{-6}$.

13.6. Show that the detection probability for a SW 1&2 target is given by the equation

$$P_D = \exp\left\{\frac{\ln(P_{fa})}{1+SNR}\right\}.$$

13.7. A certain radar has the following specifications: single pulse SNR corresponding to a reference range $R_0 = 200Km$ is $10dB$. The probability of detection at this range is $P_D = 0.95$. Assume a Swerling I type target. Use the radar equation to compute the required pulse widths at ranges $R = 220Km, 250Km$, and $175Km$, so that the probability of detection is maintained.

13.8. Repeat Problem 13.8 for a Swerling IV type target.

13.9. Utilizing the MATLAB functions presented in this chapter, plot the actual value for the improvement factor versus the number of integrated pulses. Pick three different values for the probability of false alarm.

13.10. A circularly scanning, fan beam radar has a rotation rate of 2 seconds per revolution. The azimuth beamwidth is 1.5 degrees and the radar uses a PRF of 12.5*KHz*. The radar uses an unmodulated pulse with a width of 1.2μs and searches a range window that extends from 15*Km* to 100*Km*. The range cells used during search are separated by one pulse width. It is desired that the false alarm probability be set so that the radar experiences only one false alarm every 2*min*. What is the required P_{fa} for each range cell? What is the threshold-to-noise ratio, in *dB*, needed to maintain that P_{fa}?

13.11. The probability of recording a detection in a particular range-angle cell of the scanning radar of Problem 13.11 is 0.7. What is the cumulative detection probability if the cell is checked on three successive scans? If the false alarm probability for a certain range-angle cell of the same radar is 10^{-6} what is the cumulative false alarm probability for that cell over three scans?

13.12. A radar with a phased array antenna conducts a search using a 1500-beam search raster. That is, it steps through 1500 beam positions that span a certain angular area. It transmits one pulse per beam. The radar uses range gates separated by 10*m*. The output of each range gate is sent to a bank of Doppler filters with a width of 1000*Hz* each. Thus, the signal processor consists of a set of range gates with a bank of Doppler filters connected to each range gate output. The output of the signal processor consists of a range-Doppler array of signals that consists of MN elements where M is the number of range gates and N is the number of Doppler filter outputs. During the particular search of interest, the detection processor covers a range extent of 10*Km* and a Doppler extent of 25*Km*. The design specifications state that, in this mode, the radar must have less than one false alarm every 10 scans through the search raster. What is the required P_{fa} in each range-Doppler-beam cell needed to support this requirement?

13.13. A certain radar employs a noncoherent integrator that integrates 25 pulses. What are the integrator gains, in dB, for a SW0, a SW1, a SW2, a SW3, and a SW4 target? Briefly discuss how you arrived at each of your answers. If needed, assume that the radar is to operate with a desired detection probability of 0.9.

13.14. A certain radar has the following parameters: Peak power $P_t = 500KW$, total losses $L = 12dB$, operating frequency $f_o = 5.6GHZ$, PRF $f_r = 2KHz$, pulse width $\tau = 0.5\mu s$, antenna beamwidth $\theta_{az} = 2°$ and $\theta_{el} = 7°$, noise figure $F = 6dB$, and scan time $T_{sc} = 2s$. The radar can experience one false alarm per scan. (a) What is the probability of false alarm? Assume that the radar searches a minimum range of 10*Km* to its maximum unambiguous range. (b) Plot the detection range versus RCS in dBsm. The detection range is defined as the range at which the single scan probability of detection is equal to 0.94. Generate curves for Swerling I, II, III, and IV type targets. (c) Repeat part (b) above when noncoherent integration is used.

13.15. A certain circularly scanning radar with a fan beam has a rotation rate of 3 seconds per revolution. The azimuth beamwidth is 3 degrees, and the radar uses a PRI of 600 microseconds. The radar pulse width is 2 microseconds and the radar searches a range window that extends from 15*Km* to 100*Km*. It is desired that the false alarm rate not be higher than two false alarms per revolution. What is the required probability of false alarm? What is the minimum SNR so that the minimum probability of false alarm can be maintained?

13.16. Write a MATLAB program to compute the CA-CFAR threshold value. Use similar approach to that used in the case of a fixed threshold.

13.17. Develop a MATLAB program to calculate the cumulative probability of detection.

13.18. Derive Eq. (13.79).

13.19. The sum inside Eq. (13.79) presents a very formidable challenge. It can be, however, computed recursively with relative ease. Develop a recursive algorithm to calculate this sum.

13.20. Starting with Eq. (13.81), show that as N is increased so is the over all probability of false alarm. More specifically, prove that $P_{FA} \approx NP_{fa}$.

Appendix 13-A The Incomplete Gamma Function

The Gamma Function

Define the Gamma function (not the incomplete Gamma function) of the variable z (generally complex) as

$$\Gamma(z) = \int_0^\infty x^{z-1} \; e^{-x} \; dx \qquad \textbf{Eq. (13.87)}$$

and when z is a positive integer, then

$$\Gamma(z) = (z-1)! \, . \qquad \textbf{Eq. (13.88)}$$

One very useful and frequently used property is

$$\Gamma(z+1) = z\Gamma(z) \qquad \textbf{Eq. (13.89)}$$

The Incomplete Gamma Function

The incomplete gamma function $\Gamma_I(u, q)$ used in this text is given by

$$\Gamma_I(u, q) = \int_0^{u\sqrt{q+1}} \frac{e^{-x} \; x^q}{q!} \; dx \, . \qquad \textbf{Eq. (13.90)}$$

Another definition, which is often used in the literature, for the incomplete Gamma function is

$$\Gamma_I[z, q] = \int_q^\infty x^{z-1} \; e^{-x} \; dx \, . \qquad \textbf{Eq. (13.91)}$$

It follows that

$$\Gamma(z) = \Gamma_I[z, 0] = \int_0^\infty x^{z-1} \; e^{-x} dx \, , \qquad \textbf{Eq. (13.92)}$$

which is the same as Eq. (13.80). Furthermore, for a positive integer n, the incomplete Gamma function can be represented by

$$\Gamma_I[n, z] = (n-1)! e^{-z} \sum_{k=0}^{n-1} \frac{z^k}{k!} \, . \qquad \textbf{Eq. (13.93)}$$

In order to relate $\Gamma_I[n, z]$ and $\Gamma_I(u, q)$, compute the following relation

$$\Gamma_I[a, 0] - \Gamma_I[a, z] = \int_0^\infty x^{a-1} \; e^{-x} dx - \int_z^\infty x^{a-1} \; e^{-x} dx = \int_0^z x^{a-1} \; e^{-x} dx \, . \qquad \textbf{Eq. (13.94)}$$

Applying the change of variables $a = q + 1$ and $z = u\sqrt{q+1}$ yields

$$\Gamma_I[q+1,0] - \Gamma_I[q+1, u\sqrt{q+1}] = \int_0^{u\sqrt{q+1}} x^q \; e^{-x} dx\,, \qquad \textbf{Eq. (13.95)}$$

and if q is a positive integer then

$$\frac{\Gamma_I[q+1,0] - \Gamma_I[q+1, u\sqrt{q+1}]}{q!} = \int_0^{u\sqrt{q+1}} \frac{x^q \; e^{-x}}{q!} dx = \Gamma_I(u,q)\,. \qquad \textbf{Eq. (13.96)}$$

Using Eqs. (13.81) and (7.86) in Eq. (13.89) yields

$$\Gamma_I(u,q) = 1 - \frac{(q+1-1)!\, e^{-u\sqrt{q+1}}}{q!} \sum_{k=0}^{q} \frac{(u\sqrt{q+1})^k}{k!}\,. \qquad \textbf{Eq. (13.97)}$$

Finally, the incomplete Gamma function can be written as

$$\Gamma_I(u,q) = 1 - e^{-u\sqrt{q+1}} \sum_{k=0}^{q} \frac{(u\sqrt{q+1})^k}{k!}\,. \qquad \textbf{Eq. (13.98)}$$

The two limiting values for Eq. (13.91) are

$$\Gamma_I(0,q) = 0 \qquad \Gamma_I(\infty,q) = 1\,. \qquad \textbf{Eq. (13.99)}$$

Figure 13A.1 shows the incomplete gamma function for $q = 1, 3, 5, 8$. This figure can be reproduced using the MATLAB program *"Fig13A_1.m"* listed in Appendix 13-B, which utilizes the built-in MATLAB function *"gammainc.m."*

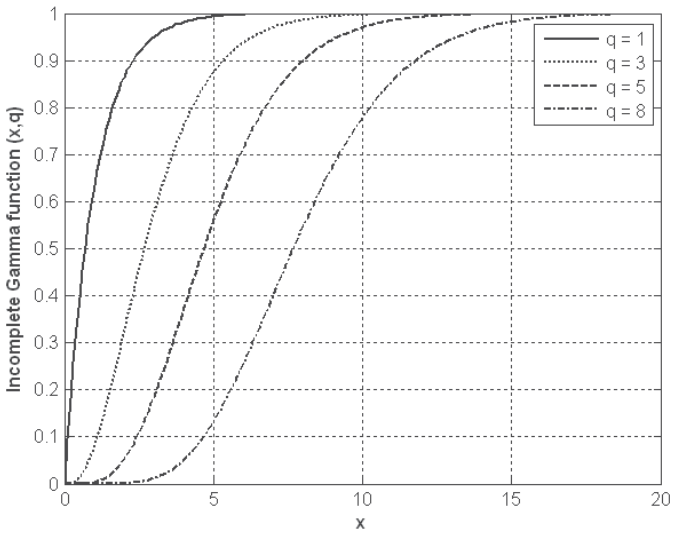


Figure 13A.1. The incomplete Gamma function for four values of q.

Appendix 13-B: Chapter 13 MATLAB Code Listings

The MATLAB code provided in this chapter was designed as an academic standalone tool and is not adequate for other purposes. The code was written in a way to assist the reader in gaining a better understanding of the theory. The code was not developed, nor is it intended to be used as part of an open-loop or a closed-loop simulation of any kind. The MATLAB code found in this textbook can be downloaded from this book's web page on the CRC Press web-site. Simply use your favorite web browser, go to www.crcpress.com, and search for keyword *"Mahafza"* to locate this book's web page.

MATLAB Function "improv_fac.m" Listing

```
function impr_of_np = improv_fac (np, pfa, pd)
% This function computes the non-coherent integration improvement
% factor using the empirical formula defind in Eq. (13.10)
% Inputs
   % np          == number of pulses
   % pfa         == probability of false alaram
   % pd          == probability of detection
%% Output
   % impr_of_np    == improvement factor for np pulses
fact1 = 1.0 + log10( 1.0 / pfa) / 46.6;
fact2 = 6.79 .* (1.0 + 0.253 .* pd);
fact3 = 1.0 - 0.14 .* log10(np) + 0.0183 .* (log10(np)).^2;
impr_of_np = fact1 .* fact2 .* fact3 .* log10(np);
end
```

MATLAB Program "Fig13_2.m" Listing

```
% This program is used to produce Fig. 13.2
% It uses the function "improv_fac".
clc
clear all
close all
Pfa = [1e-2, 1e-6, 1e-8, 1e-10];
Pd = [.5 .8 .95 .99];
np = linspace(1,1000,10000);
I(1,:) = improv_fac (np, Pfa(1), Pd(1));
I(2,:) = improv_fac (np, Pfa(2), Pd(2));
I(3,:) = improv_fac (np, Pfa(3), Pd(3));
I(4,:) = improv_fac (np, Pfa(4), Pd(4));
index = [1 2 3 4];
L(1,:) = 10.*log10(np) - I(1,:);
L(2,:) = 10.*log10(np) - I(2,:);
L(3,:) = 10.*log10(np) - I(3,:);
L(4,:) = 10.*log10(np) - I(4,:);
subplot(2,1,2)
semilogx (np, L(1,:), 'k:', np, L(2,:), 'k-.', np, L(3,:), 'k-.', np, L(4,:), 'k','linewidth',1.5)
xlabel ('\bfNumber of pulses');
ylabel ('\bfIntegration loss in dB')
axis tight
grid
subplot(2,1,1)
semilogx (np, I(1,:), 'k:', np, I(2,:), 'k-.', np, I(3,:), 'k--', np, I(4,:), 'k','linewidth',1.5)
```

```
%set (gca,'xtick',[1 2 3 4 5 6 7 8  10 20 30 100]);
xlabel ('\bfNumber of pulses');
ylabel ('\bfImprovement factor in dB')
legend ('P_D=.5, P_f_a=10^-^2','P_D=.8, P_f_a=10^-^6','P_D=.95, P_f_a=10^-^8','P_D=.99, P_f_a=10^-^1^0');
grid
axis tight
```

MATLAB Function "threshold.m" Listing

```
function [pfa, vt] = threshold (nfa, np)
% This function calculates the threshold value from nfa and np.
% The newton-Raphson  recursive formula
% This function uses "gammainc.m".
% Inputs
   % nfa          == number of false alarm
   % np           == number of pulses
%% Outputs
   % Pfa          == probability of false alarm
   % vt           == threshold
%
delta = eps;
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 (delta < (vt0/10000));
  igf = gammainc(vt0,np);
  num = 0.5^(np/nfa) - igf;
  deno = -exp(-vt0) * vt0^(np-1) /factorial(np-1);
  vt = vt0 - (num / (deno+eps));
  delta = abs(vt - vt0);
  vt0 = vt;
end
```

MATLAB Program "Fig13_4.m" Listing

```
% Use this program to reproduce Fig. 13.4 of text
clear all
close all
for n= 1: 1:10000
  [pfa1 y1(n)] = threshold(1e4,n);
  [pfa2 y3(n)] = threshold(1e8,n);
  [pfa3 y4(n)] = threshold(1e12,n);
end
n =1:1:10000;
loglog(n,y1,'k',n,y3,'k--',n,y4,'k-.','linewidth',1.5);
xlabel ('\bfNumber of pulses');
ylabel('\bfThreshold')
legend('nfa=10^1^2','nfa=10^8','nfa=10^8')
grid
```

MATLAB Function "pd_swerling5.m" Listing

```
function pd = pd_swerling5 (input1, indicator, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 5 or 0 targets for np>1.
%
% Inputs
   % input1       == Pfa or nfa
   % indicator    == 1 when input1 = Pfa; 2 when input1 = nfa
   % np           == number of pulses
   % snrbar       == SNR
% Outputs
   % pd           == probability of detection
if(np == 1)
  'Stop, np must be greater than 1'
  return
end
format long
snrbar = 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 (delta < (vt0/10000));
  igf = gammainc(vt0,np);
  num = 0.5^(np/nfa) - igf;
  deno = -exp(-vt0) * vt0^(np-1) /factorial(np-1);
  vt = vt0 - (num / (deno+eps));
  delta = abs(vt - vt0);
  vt0 = vt;
end
% Calculate the Gram-Chrlier 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;
Vsqr = V .*V;
val1 = exp(-Vsqr ./ 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;
return
```

MATLAB Program "Fig13_5.m" Listing

```
% This program is used to produce Fig. 13.5
clc
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:','linewidth',1.5);
axis([0 20 0 1])
xlabel ('\bfSNR in dB')
ylabel ('\bfProbability of detection')
legend('n_p = 1','n_p = 10')
grid on
```

MATLAB Function "pd_swerling1.m" Listing

```
function pd = pd_swerling1 (nfa, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 1 targets.
%
% Inputs
   % nfa          == Marcum's false alarm number
   % np           == number of integrated pulses
   % snrbar       == SNR
%
% outputs
   % pd           == probability of detection
format long
snrbar = 10.0^(snrbar/10.);
eps = 0.00000001;
delta = eps;
% 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 (delta < (vt0/10000));
   igf = gammainc(vt0,np);
   num = 0.5^(np/nfa) - igf;
   deno = -exp(-vt0) * vt0^(np-1) /factorial(np-1);
   vt = vt0 - (num / (deno+eps));
   delta = abs(vt - vt0);
   vt0 = vt;
end
```

```
if (np == 1)
   temp = -vt / (1.0 + snrbar);
   pd = exp(temp);
   return
end
   temp1 = 1.0 + np * snrbar;
   temp2 = 1.0 / (np *snrbar);
   temp = 1.0 + temp2;
   val1 = temp^(np-1.);
   igf1 = gammainc(vt,np-1);
   igf2 = gammainc(vt/temp,np-1);
   pd = 1.0 - igf1 + val1 * igf2 * exp(-vt/temp1);
   return
```

MATLAB Program "Fig13_6.m" Listing

```
% This program is used to reproduce Fig. 13.6
clc
close all
clear all
pfa = 1e-9;
nfa = log(2) / pfa;
b = sqrt(-2.0 * log(pfa));
index = 0;
for snr = 0:.01:22
   index = index +1;
   a = sqrt(2.0 * 10^(.1*snr));
   swer0(index) = marcumsq(a,b);
   swer1(index) =  pd_swerling1 (nfa, 1, snr);
end
x = 0:.01:22;
%figure(10)
plot(x, swer0,'k',x,swer1,'k:','linewidth', 1.5);
axis([2 22 0 1])
xlabel ('\bfSNR in dB')
ylabel ('\bfProbability of detection')
legend('Swerling 0','Swerling I')
grid on
```

MATLAB Program "Fig13_7.m" Listing

```
% This program is used to produce Fig. 13.7
clc
clear all
close all
pfa = 1e-6;
nfa = log(2) / pfa;
index = 0;
for snr = -10:.5:30
   index = index +1;
   prob1(index) =  pd_swerling1 (nfa, 15, snr);
   prob0(index) =  pd_swerling5 (nfa, 2, 15, snr);
  end
x = -10:.5:30;
```

```
plot(x, prob1,'k',x,prob0,'k:','linewidth',1.5);
axis([-10 30 0 1])
xlabel ('\bfSNR in dB')
ylabel ('\bfProbability of detection')
legend('Swerling I','Swerling 0')
title('\bfP_f_a =10^-^6;  n_p=5')
grid
```

MATLAB Function "pd_swerling2.m" Listing

```
function pd = pd_swerling2 (nfa, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 2 targets.
% Inputs
   % nfa          == number of fals alarm
   % np           == number of pulses
   % snrbar       == SNR
%
% Outputs
   % pd           == proability of detection
format long
snrbar = 10.0^(snrbar/10.);
eps = 0.00000001;
delta = eps;
% 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 (delta < (vt0/10000));
  igf = gammainc(vt0,np);
  num = 0.5^(np/nfa) - igf;
  deno = -exp(-vt0) * vt0^(np-1) /factorial(np-1);
  vt = vt0 - (num / (deno+eps));
  delta = abs(vt - vt0);
  vt0 = vt;
end
if (np <= 50)
  temp = vt / (1.0 + snrbar);
  pd = 1.0 - gammainc(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;
  Vsqr = V *V;
  val1 = exp(-Vsqr / 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
return
```

MATLAB Program "Fig13_8.m" Listing

```
% This program is used to produce Fig. 13.8
clc
clear all
close all
pfa = 1e-7;
nfa = log(2) / pfa;
index = 0;
for snr = -10:.5:30
   index = index +1;
   prob1(index) =  pd_swerling1 (nfa, 5, snr);
   prob0(index) =  pd_swerling5 (nfa, 2, 5, snr);
   prob2(index) =  pd_swerling2 (nfa, 5, snr);
end
x = -10:.5:30;
plot(x, prob0,'k',x,prob1,'k:',x,prob2,'k--','linewidth',1.5);
axis([-10 30 0 1])
xlabel ('\bfSNR in dB')
ylabel ('\bfProbability of detection')
legend('Swerling 0','Swerling I','Swerling II')
title('P_f_a =10^-^7;  n=5')
grid
```

MATLAB Program "Fig13_9.m" Listing

```
% This program is used to produce Fig. 13.9
clear all
close all
pfa = 1e-6;
nfa = log(2) / pfa;
index = 0;
b = sqrt(-2.0 * log(pfa));
for snr = -10:.5:30
   a = sqrt(2.0 * 10^(.1*snr));
   index = index +1;
   prob1(index) =  pd_swerling1 (nfa, 2, snr);
   prob0(index) =  marcumsq(a,b);
   prob2(index) =  pd_swerling2 (nfa, 2, snr);
end
x = -10:.5:30;
plot(x, prob0,'k',x,prob1,'k:',x,prob2,'k--','linewidth',1.5);
axis([-10 30 0 1])
xlabel ('\bfSNR in dB')
ylabel ('\bfProbability of detection')
legend('Swerling 0','Swerling I','Swerling II')
title('P_f_a =10^-^6;  n_p = 2')
grid on
```

MATLAB Function "pd_swerling3.m" Listing

```
function pd = pd_swerling3 (nfa, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 2 targets.
% Inputs
   % nfa      == false alarm number
   % np       == number of pulses
   % snrbar   == SNR
% Outputs
   % pd       == probability of detection
format long
snrbar = 10.0^(snrbar/10.);
eps = 0.00000001;
delta = eps;
% 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 (delta < (vt0/10000));
  igf = gammainc(vt0,np);
  num = 0.5^(np/nfa) - igf;
  deno = -exp(-vt0) * vt0^(np-1) /factorial(np-1);
  vt = vt0 - (num / (deno+eps));
  delta = abs(vt - vt0);
  vt0 = vt;
end
temp1 = vt / (1.0 + 0.5 * np *snrbar);
temp2 = 1.0 + 2.0 / (np * snrbar);
temp3 = 2.0 * (np - 2.0) / (np * snrbar);
ko = exp(-temp1) * temp2^(np-2.) * (1.0 + temp1 - temp3);
if (np <= 2)
  pd = ko;
  return
else
  ko = exp(-temp1) * temp2^(np-2.) * (1.0 + temp1 - temp3);
  temp4 = vt^(np-1.) * exp(-vt) / (temp1 * (factorial(np-2.)));
  temp5 =  vt / (1.0 + 2.0 / (np *snrbar));
  pd = temp4 + 1.0 - gammainc(vt,np-1.) + ko * gammainc(temp5,np-1.);
end; return
```

MATLAB Program "Fig13_10.m" Listing

```
% This program is used to produce Fig. 13.10
clc
close all
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-.','linewidth',1.5);
axis([-10 30 0 1])
xlabel ('SNR in dB')
ylabel ('Probability of detection')
legend('np = 1','np = 10','np = 50','np = 100')
grid on
```

MATLAB Program "Fig13_11.m" Listing

```
% This program is used to produce Fig. 13.11
clc
clear all
close all
pfa = 1e-7;
nfa = log(2) / pfa;
index = 0;
for snr = -10:.5:30
  index = index +1;
  prob1(index) =  pd_swerling1 (nfa, 5, snr);
  prob0(index) =  pd_swerling5 (nfa, 2, 5, snr);
  prob2(index) =  pd_swerling2 (nfa, 5, snr);
  prob3(index) =  pd_swerling3 (nfa, 5, snr);
end
x = -10:.5:30;
plot(x, prob0,'k',x,prob1,'k:',x,prob2,'k--',x,prob3,'k-.','linewidth',1,'linewidth',1.5);
axis([-10 30 0 1])
xlabel ('\bfSNR in dB')
ylabel ('P\bfrobability of detection')
legend('Swerling 0','Swerling I','Swerling II', 'Swerling III')
title('P_f_a =10^-^7;  n=5')
grid on
```

MATLAB Function "pd_swerling4.m" Listing

```
function pd = pd_swerling4 (nfa, np, snrbar)
% This function is used to calculate the probability of detection
% for Swerling 4 targets.
% Inputs
   % nfa          == number of false alarm
   % np           == number of pulses
   % snrbar       == SNR
% Output
   % pd           == probability of detection
format long
snrbar = 10.0^(snrbar/10.);
eps = 0.00000001;
delta = eps;
% 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 (delta < (vt0/10000));
  igf = gammainc(vt0,np);
  num = 0.5^(np/nfa) - igf;
  deno = -exp(-vt0) * vt0^(np-1) /factorial(np-1);
  vt = vt0 - (num / (deno+eps));
  delta = abs(vt - vt0);
  vt0 = vt;
end
h8 = snrbar /2.0;
beta = 1.0 + h8;
beta2 = 2.0 * beta^2 - 1.0;
beta3 = 2.0 * beta^3;
if (np >= 50)
  temp1 = 2.0 * beta -1;
  omegabar = sqrt(np * temp1);
  c3 = (beta3 - 1.) / 3.0 / beta2 / omegabar;
  c4 = (beta3 * beta3 - 1.0) / 4. / np /beta2 /beta2;;
  c6 = c3 * c3 /2.0;
  V = (vt - np * (1.0 + snrbar)) / omegabar;
  Vsqr = V *V;
  val1 = exp(-Vsqr / 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;
  return
else
  gamma0 = gammainc(vt/beta,np);
  a1 = (vt / beta)^np / (factorial(np) * exp(vt/beta));
  sum = gamma0;
  for i = 1:1:np
    temp1 = gamma0;
    if (i == 1)
      ai = a1;
    else
      ai = (vt / beta) * a1 / (np + i -1);
    end
    gammai = gamma0 - ai;
    gamma0 = gammai;
    a1 = ai;
    for ii = 1:1:i
      temp1 = temp1 * (np + 1 - ii);
    end
    term = (snrbar /2.0)^i * gammai * temp1 / (factorial(i));
    sum = sum + term;
  end
  pd = 1.0 - (sum / beta^np);
end
pd = max(pd,0.);
return
```

MATLAB Program "Fig13_12.m" Listing

```
% This program is used to produce Fig. 13.12 of text
clear all
close all
pfa = 1e-6;
nfa = log(2) / pfa;
index = 0;
for snr = -7:.15:10
   index = index +1;
   prob1(index) =  pd_swerling4 (nfa, 5, snr);
   prob10(index) =  pd_swerling4 (nfa, 10, snr);
   prob25(index) =  pd_swerling4(nfa, 25, snr);
   prob75(index) =  pd_swerling4 (nfa, 75, snr);
end
x = -7:.15:10;
plot(x, prob1,'k',x,prob10,'k.',x,prob25,'k:',x, prob75,'k-.','linewidth',1.5);
xlabel ('\bfSNR in dB')
ylabel ('\bfProbability of detection')
legend('np = 5','np = 10','np = 25','np = 75')
grid on; axis tight
```

MATLAB Function "fluct_loss.m" Listing

```
function [SNR] = fluct(pd, nfa, np, sw_case)
% This function calculates the SNR fluctuation loss for Swerling models
% A negative Lf value indicates SNR gain instead of loss
% Inputs
   % pd       == desired probability of detection
   % nfa      == desired number of false alarms
   % np       == number of pulses
   % sw_case   == 0, 1, 2, 3, or 4 depending on the desired Swerling case
% Output
   % SNR      == Resulting SNR
format long
% *************** Swerling 5 case ****************
% check to make sure that np>1
pfa =  np * log(2) / nfa;
if (sw_case == 0)
if (np ==1)
   nfa = 1/pfa;
   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;
   SNR = snr_inc;
else
   % np > 1 use MATLAB function pd_swerling5.m
   snr_inc = 0.1 - 0.001;
   Pd_Sw5 = 0.001;
```

```
   while(Pd_Sw5 <= pd)
      snr_inc = snr_inc + 0.001;
      Pd_Sw5 = pd_swerling5(pfa, 1, np, snr_inc);
   end
   PD_SW5 = Pd_Sw5;
   SNR = snr_inc;
end
end
% *************** End Swerling 5 case ************
% *************** Swerling 1 case ****************
% compute the false alarm number
if (sw_case == 1)
   Pd_Sw1 = 0.001;
   snr_inc = 0.1 - 0.001;
   while(Pd_Sw1 <= pd)
      snr_inc = snr_inc + 0.001;
      Pd_Sw1 = pd_swerling1(nfa, np, snr_inc);
   end
   PD_SW1 = Pd_Sw1;
   SNR = snr_inc;
end
 % *************** End Swerling 1 case ************
% *************** Swerling 2 case ****************
if (sw_case == 2)
   Pd_Sw2 = 0.001;
   snr_inc = 0.1 - 0.001;
   while(Pd_Sw2 <= pd)
      snr_inc = snr_inc + 0.001;
      Pd_Sw2 = pd_swerling2(nfa, np, snr_inc);
   end
   PD_SW2 = Pd_Sw2;
   SNR = snr_inc;
end
 % *************** End Swerling 2 case ************
% *************** Swerling 3 case ****************
if (sw_case == 3)
   Pd_Sw3 = 0.001;
   snr_inc = 0.1 - 0.001;
   while(Pd_Sw3 <= pd)
      snr_inc = snr_inc + 0.001;
      Pd_Sw3 = pd_swerling3(nfa, np, snr_inc);
   end
   PD_SW3 = Pd_Sw3;
   SNR = snr_inc;
 end
 % *************** End Swerling 3 case ************
% *************** Swerling 4 case ****************
if (sw_case == 4)
   Pd_Sw4 = 0.001;
   snr_inc = 0.1 - 0.001;
   while(Pd_Sw4 <= pd)
      snr_inc = snr_inc + 0.001;
      Pd_Sw4 = pd_swerling4(nfa, np, snr_inc);
   end
```

```
   PD_SW4 = Pd_Sw4;
   SNR = snr_inc;
 end
 % *************** End Swerling 4 case ************
return
```

MATLAB Program "Fig13_13.m" Listing

```
% Use this program to reproduce Fig. 13.13 of text
clear all
close all
index = 0.;
for pd = 0.01:.05:1
   index = index + 1;
   [Lf,Pd_Sw5] = fluct_loss(pd, 1e-7,1,1);
   Lf1(index) = Lf;
   [Lf,Pd_Sw5] = fluct_loss(pd, 1e-7,1,4);
   Lf4(index) = Lf;
end
pd = 0.01:.05:1;
figure (3)
plot(pd, Lf1, 'k',pd, Lf4,'K:','linewidth',1.5)
xlabel('\bfProbability of detection')
ylabel('\bfFluctuation loss - dB')
legend('Swerling I & II','Swerling III & IV')
title('P_f_a = 10^-^9, n_p = 1')
grid on
```

MATLAB Program "Fig13A_1.m" Listing

```
% This program can be used to reproduce Fig. 13A.1
clc
close all
clear all
x=linspace(0,20,200);
y1 = gammainc(x,1);
y2 = gammainc(x,3);
y3 = gammainc(x,5);
y4 = gammainc(x,8);
plot(x,y1,'k',x,y2,'k:',x,y3,'k--',x,y4,'k-.','linewidth',1.5)
legend('q = 1','q = 3','q = 5','q = 8')
xlabel('\bfx')
ylabel('\bfIncomplete Gamma function (x,q)')
grid
```

Part V

Radar Special Topics

Chapter 14

Radar Cross Section (RCS)

This chapter was coauthored with Mr. Walton C. Gibson.[1]

14.1. RCS Definition

Electromagnetic waves, with any specified polarization, are normally diffracted or scattered in all directions when incident on a target. These scattered waves are broken down into two parts. The first part is made of waves that have the same polarization as the receiving antenna. The other portion of the scattered waves will have a different polarization to which the receiving antenna does not respond. The two polarizations are orthogonal and are referred to as the Principal Polarization (PP) and Orthogonal Polarization (OP), respectively. The intensity of the *backscattered* energy that has the same polarization as the radar's receiving antenna is used to define the target RCS. When a target is illuminated by RF energy, it acts like an antenna, and will have near and far scattered fields. Waves reflected and measured in the near field are, in general, spherical. Alternatively, in the far field the wavefronts are decomposed into a linear combination of plane waves.

Assume the power density of a wave incident on a target located at range R away from the radar is P_{Di}, as illustrated in Fig. 14.1. The amount of reflected power from the target is

$$P_r = \sigma P_{Di} \qquad \text{Eq. (14.1)}$$

where σ denotes the target cross section. Define P_{Dr} as the power density of the scattered waves at the receiving antenna. It follows that

$$P_{Dr} = P_r/(4\pi R^2)\,. \qquad \text{Eq. (14.2)}$$

Equating Eqs. (14.1) and (14.2) yields

$$\sigma = 4\pi R^2\left(\frac{P_{Dr}}{P_{Di}}\right) \qquad \text{Eq. (14.3)}$$

1. Mr. Gibson is associated with Tripoint Industries, Inc. in Huntsville, Alabama, *www.tripointindustries.com.*

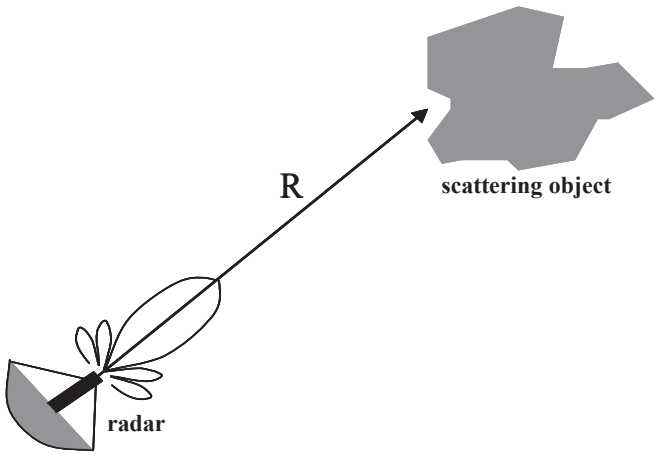


Figure 14.1. Scattering object located at range R.

and in order to ensure that the radar receiving antenna is in the far field (i.e., scattered waves received by the antenna are planar), Eq. (14.3) is modified to

$$\sigma = 4\pi R^2 \lim_{R \to \infty} \left(\frac{P_{Dr}}{P_{Di}} \right). \qquad \textbf{Eq. (14.4)}$$

The RCS defined by Eq. (14.4) is often referred to as the monostatic RCS, the backscattered RCS, or simply the target RCS.

The backscattered RCS is measured from all waves scattered in the direction of the radar and has the same polarization as the receiving antenna. It represents a portion of the total scattered target RCS σ_t, where $\sigma_t > \sigma$. Assuming a spherical coordinate system defined by (ρ, θ, φ), then at range ρ, the target scattered cross section is a function of (θ, φ). Let the angles (θ_i, φ_i) define the direction of propagation of the incident waves. Also, let the angles (θ_s, φ_s) define the direction of propagation of the scattered waves. The special case, when $\theta_s = \theta_i$ and $\varphi_s = \varphi_i$, defines the monostatic RCS. The RCS measured by the radar at angles $\theta_s \neq \theta_i$ and $\varphi_s \neq \varphi_i$ is called the bistatic RCS. The total target scattered RCS is given by

$$\sigma_t = \frac{1}{4\pi} \int_{\varphi_s = 0}^{2\pi} \int_{\theta_s = 0}^{\pi} \sigma(\theta_s, \varphi_s) \sin\theta_s \, d\theta \, d\varphi_s. \qquad \textbf{Eq. (14.5)}$$

The amount of backscattered waves from a target is proportional to the ratio of the target extent (size) to the wavelength, λ, of the incident waves. In fact, a radar will not be able to detect targets much smaller than its operating wavelength. For example, if weather radars use L-band frequency, rain drops become nearly invisible to the radar since they are much smaller than the wavelength. The frequency region, where the target extent and the wavelength are comparable, is referred to as the Rayleigh region. Alternatively, the frequency region where the target extent is much larger than the radar operating wavelength is referred to as the optical region. In practice, the majority of radar applications fall within the optical region.

The analysis presented in this book mainly assumes far field monostatic RCS measurements in the optical region. Near field RCS, bistatic RCS, and RCS measurements in the Rayleigh region will not be considered since their treatment falls beyond this book's intended scope. Additionally, RCS treatment in this chapter is mainly concerned with Narrow Band (NB) cases. In other words, the extent of the target under consideration falls within a single range bin of the radar. Wideband (WB) RCS measurements will be briefly addressed in a later section. Wideband radar range bins are small (typically 10 - 50 cm); hence, the target under consideration may cover many range bins. The RCS value in an individual range bin corresponds to the portion of the target falling within that bin.

14.2. RCS Dependency on Aspect Angle and Frequency

Radar cross section fluctuates as a function of radar aspect angle and frequency. For the purpose of illustration, isotropic point scatterers are considered. An isotropic scatterer is one that scatters incident waves equally in all directions. Consider the geometry shown in Fig. 14.2. In this case, two unity ($1m^2$) isotropic scatterers are aligned and placed along the radar line of sight (zero aspect angle) at a far field range R. The spacing between the two scatterers is 1 meter. The radar aspect angle is then changed from zero to 180 degrees, and the composite RCS of the two scatterers measured by the radar is computed.

This composite RCS consists of the superposition of the two individual radar cross sections. At zero aspect angle, the composite RCS is $2m^2$. Taking scatterer-1 as a phase reference, when the aspect angle is varied, the composite RCS is modified by the phase that corresponds to the electrical spacing between the two scatterers. For example, at aspect angle $10°$, the electrical spacing between the two scatterers is

$$elec\text{–}spacing = \frac{2 \times (1.0 \times \cos(10°))}{\lambda}. \qquad \textbf{Eq. (14.6)}$$

λ is the radar operating wavelength.

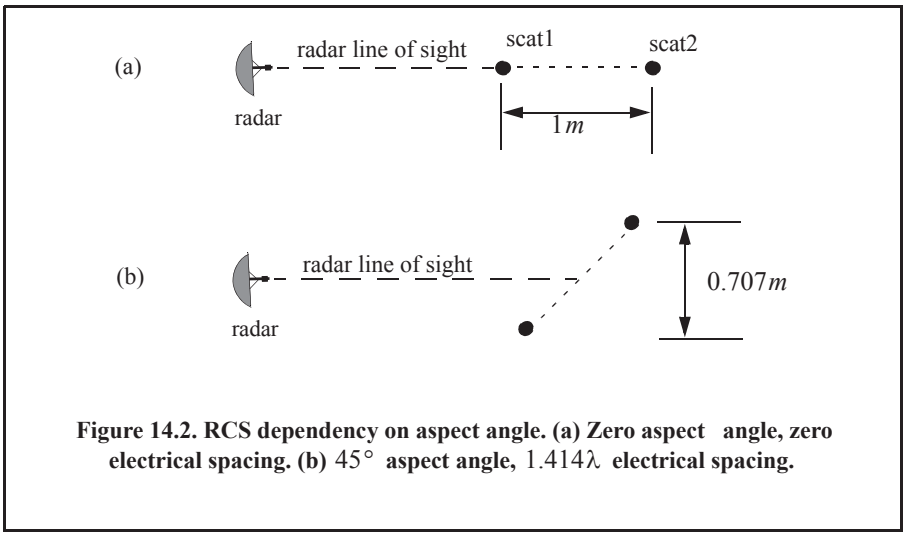


Figure 14.2. RCS dependency on aspect angle. (a) Zero aspect angle, zero electrical spacing. (b) $45°$ **aspect angle,** 1.414λ **electrical spacing.**

MATLAB Function "rcs_aspect.m"

The function *"rcs_aspect.m"* computes the RCS dependency on the aspect angle. Its syntax is as follows:

[rcs] = rcs_aspect (scat_spacing, freq)

where

Symbol	Description	Units	Status
scat_spacing	*scatterer spacing*	*meters*	*input*
freq	*radar frequency*	*Hz*	*input*
rcs	*array of RCS versus aspect angle*	*dBsm*	*output*

Figure 14.3 shows the composite RCS corresponding to this experiment. This plot can be reproduced using MATLAB program *"Fig.14.3.m"* listed in Appendix 14-A. As clearly indicated by Fig. 14.3, RCS is dependent on the radar aspect angle; thus, knowledge of this constructive and destructive interference between the individual scatterers can be very critical when a radar tries to extract the RCS of complex or maneuvering targets. This is true because of two reasons. First, the aspect angle may be continuously changing. Second, complex target RCS can be viewed as made up from contributions of many individual scattering points distributed on the target surface. These scattering points are often called scattering centers. Many approximate RCS prediction methods generate a set of scattering centers that define the backscattering characteristics of such complex targets.

Next, to demonstrate RCS dependency on frequency, consider the experiment shown in Fig. 14.4. In this case, two far field unity isotropic scatterers are aligned with radar line of sight, and the composite RCS is measured by the radar as the frequency is varied from 8*GHz* to 12.5*GHz* (X-band).

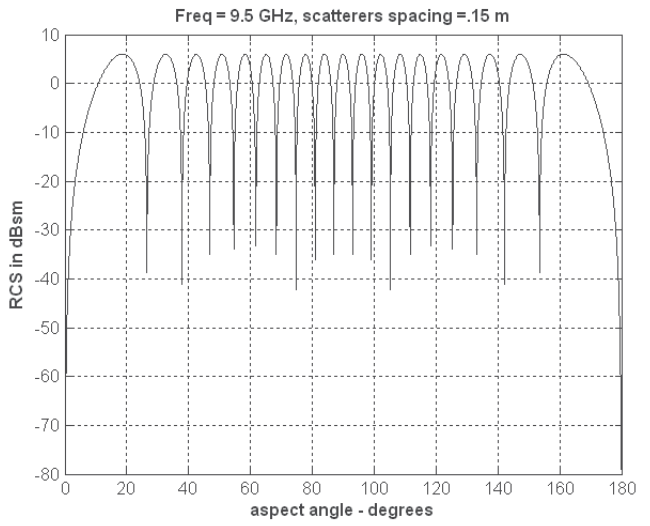


Figure 14.3. Illustration of RCS dependency on aspect angle.

MATLAB Function "rcs_frequency.m"

The function *"rcs_frequency.m"* computes the RCS dependency on frequency. Its syntax is as follows:

[rcs] = rcs_frequency (scat_spacing, frequ, freql)

where

Symbol	Description	Units	Status
scat_spacing	*scatterer spacing*	*meters*	*input*
freql, frequ	*start and end of frequency band*	*Hz*	*input*

Figures 14.5 and 14.6 show the composite RCS versus frequency for scatterer spacing of 0.25 and 0.75 meters. The plots shown in Figs. 14.5 and 14.6 can be reproduced using the MATLAB program *"Fig.14_5_6.m,"* listed in Appendix 14-A. From those two Figures, RCS fluctuation as a function of frequency is evident. A small frequency change can cause serious RCS fluctuation when the scatterer spacing is large. Alternatively, when scattering centers are relatively close, it requires more frequency variation to produce significant RCS fluctuation.

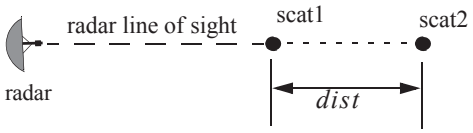


Figure 14.4. Experiment setup which demonstrates RCS dependency on frequency; dist = 0.25, or 0.75 m.

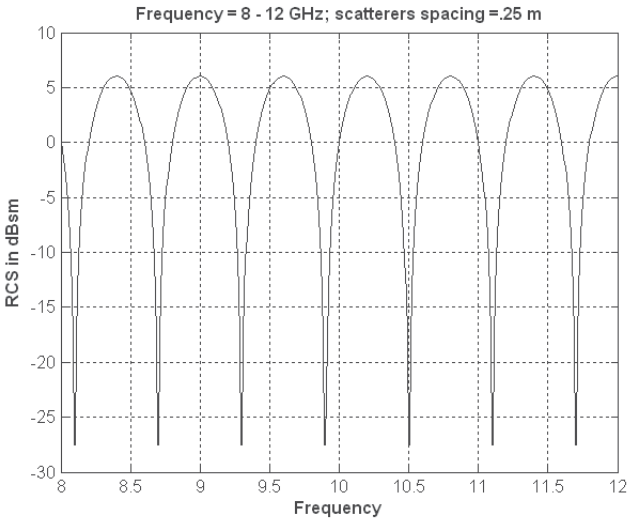


Figure 14.5. Illustration of RCS dependency on frequency.

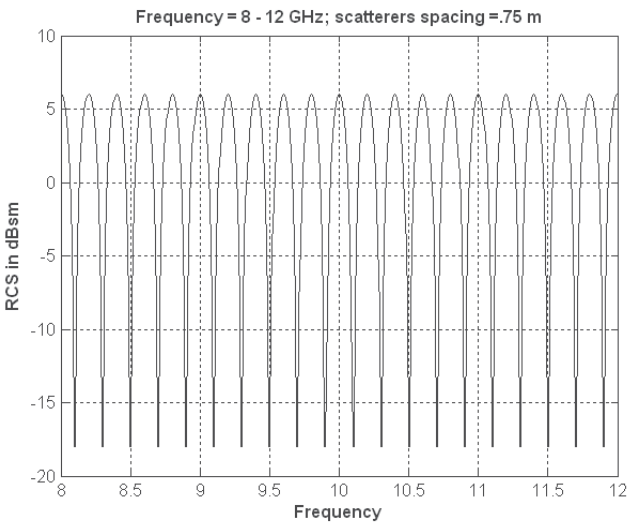


Figure 14.6. Illustration of RCS dependency on frequency.

14.3. RCS Dependency on Polarization

The material in this section covers two topics. First, a review of polarization fundamentals is presented. Second, the concept of the target scattering matrix is introduced.

14.3.1. Normalized Electric Field

In most radar simulations, it is desirable to obtain the complex-valued electric field scattered by the target at the radar. In such cases, it is useful to use a quantity called the normalized electric field. It is assumed that the incident electric field has a magnitude of unity, and is phase centered at a point at the target (usually the center of gravity). More precisely,

$$E_i = e^{jk(\vec{r}_i \cdot \hat{r})} \qquad \textbf{Eq. (14.7)}$$

where $\vec{r}_i$ is the direction of incidence and $\hat{r}$ as a location at the target, each with respect to the phase center. The normalized scattered field is then given by

$$E_s = \sigma E_i \qquad \textbf{Eq. (14.8)}$$

The quantity E_s is independent of radar and target location. It may be combined with an incident magnitude and phase.

14.3.2. Polarization

The *x* and *y* electric field components for a wave traveling along the positive z direction are given by

$$E_x = E_1 \sin(\omega t - kz) \qquad \textbf{Eq. (14.9)}$$

$$E_y = E_2 \sin(\omega t - kz + \delta) \qquad \textbf{Eq. (14.10)}$$

where $k = 2\pi/\lambda$, ω is the wave frequency, the angle δ is the time phase angle at which E_y leads E_x, and finally, E_1 and E_2 are, respectively, the wave amplitudes along the x and y directions. When two or more electromagnetic waves combine, their electric fields are integrated vectorially at each point in space for any specified time. In general, the combined vector traces an ellipse when observed in the *x*-*y* plane. This is illustrated in Fig. 14.7.

The ratio of the major to the minor axes of the polarization ellipse is called the Axial Ratio (AR). When AR is unity, the polarization ellipse becomes a circle, and the resultant wave is then called circularly polarized. Alternatively, when $E_1 = 0$ and $AR = \infty$, the wave becomes linearly polarized.

Eqs. (14.9) and (14.10) can be combined to give the instantaneous total electric field,

$$\vec{E} = \hat{a}_x E_1 \sin(\omega t - kz) + \hat{a}_y E_2 \sin(\omega t - kz + \delta) \qquad \textbf{Eq. (14.11)}$$

where $\hat{a}_x$ and $\hat{a}_y$ are unit vectors along the x and y directions, respectively. At $z = 0$, $E_x = E_1 \sin(\omega t)$ and $E_y = E_2 \sin(\omega t + \delta)$, then by replacing $\sin(\omega t)$ by the ratio E_x/E_1 and by using trigonometry properties Eq. (14.11) can be rewritten as

$$\frac{E_x^2}{E_1^2} - \frac{2E_x E_y \cos\delta}{E_1 E_2} + \frac{E_y^2}{E_2^2} = (\sin\delta)^2 . \qquad \textbf{Eq. (14.12)}$$

Note that Eq. (14.12) has no dependency on ωt. In the most general case, the polarization ellipse may have any orientation, as illustrated in Fig. 14.8. The angle ξ is called the tilt angle of the ellipse. In this case, AR is given by

$$AR = \frac{OA}{OB} \qquad (1 \le AR \le \infty) . \qquad \textbf{Eq. (14.13)}$$

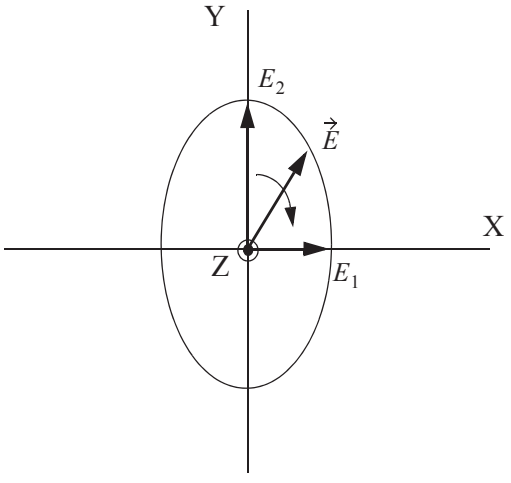


Figure 14.7. Electric field components along the *x* and *y* directions. The positive *z* direction is off of the page.

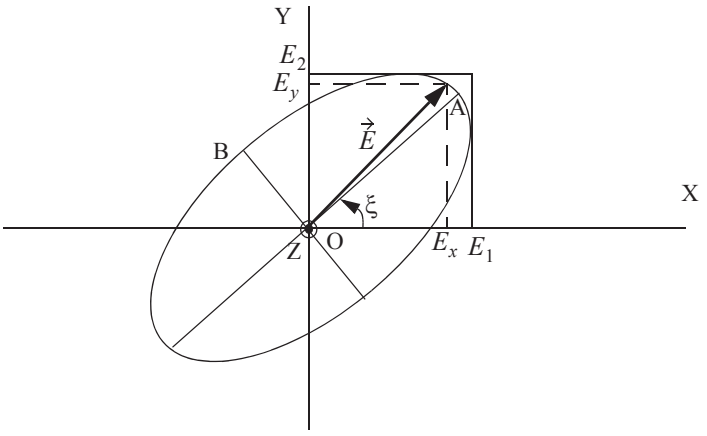


Figure 14.8. Polarization ellipse in the general case.

When $E_1 = 0$, the wave is said to be linearly polarized in the y direction, while if $E_2 = 0$, the wave is said to be linearly polarized in the x direction. Polarization can also be linear at an angle of $45°$ when $E_1 = E_2$ and $\xi = 45°$. When $E_1 = E_2$ and $\delta = 90°$, the wave is said to be Left Circularly Polarized (LCP), while if $\delta = -90°$ the wave is said to Right Circularly Polarized (RCP). It is a common notation to call the linear polarizations along the x and y directions by the names horizontal and vertical polarizations, respectively.

In general, an arbitrarily polarized electric field may be written as the sum of two circularly polarized fields. More precisely,

$$\vec{E} = \overrightarrow{E_R} + \overrightarrow{E_L} \qquad \textbf{Eq. (14.14)}$$

where $\overrightarrow{E_R}$ and $\overrightarrow{E_L}$ are the RCP and LCP fields, respectively. Similarly, the RCP and LCP waves can be written as

$$\overrightarrow{E_R} = \overrightarrow{E_V} + j\overrightarrow{E_H} \qquad \textbf{Eq. (14.15)}$$

$$\overrightarrow{E_L} = \overrightarrow{E_V} - j\overrightarrow{E_H} \qquad \textbf{Eq. (14.16)}$$

where $\overrightarrow{E_V}$ and $\overrightarrow{E_H}$ are the fields with vertical and horizontal polarizations, respectively. Combining Eqs. (14.15) and (14.16) yields

$$E_R = \frac{E_H - jE_V}{\sqrt{2}} \qquad \textbf{Eq. (14.17)}$$

$$E_L = \frac{E_H + jE_V}{\sqrt{2}}. \qquad \textbf{Eq. (14.18)}$$

Using matrix notation, Eqs. (14.17) and (14.18) can be rewritten as

$$\begin{bmatrix} E_R \\ E_L \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \begin{bmatrix} E_H \\ E_V \end{bmatrix} = [T] \begin{bmatrix} E_H \\ E_V \end{bmatrix} \qquad \textbf{Eq. (14.19)}$$

$$\begin{bmatrix} E_H \\ E_V \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ j & -j \end{bmatrix} \begin{bmatrix} E_R \\ E_L \end{bmatrix} = [T]^{-1} \begin{bmatrix} E_H \\ E_V \end{bmatrix}. \quad \text{Eq. (14.20)}$$

For many targets, the scattered waves will have different polarization than the incident waves. This phenomenon is known as depolarization or cross-polarization. However, perfect reflectors reflect waves in such a fashion that an incident wave with horizontal polarization remains horizontal, and an incident wave with vertical polarization remains vertical but is phase shifted $180°$. Additionally, an incident wave that is RCP becomes LCP when reflected, and a wave that is LCP becomes RCP after reflection from a perfect reflector. Therefore, when a radar uses LCP waves for transmission, the receiving antenna needs to be RCP polarized in order to capture the PP RCS, and LCR to measure the OP RCS.

14.3.3. Target Scattering Matrix

Target backscattered RCS is commonly described by a matrix known as the scattering matrix, and is denoted by $[S]$. When an arbitrarily linearly polarized wave is incident on a target, the backscattered field is then given by

$$\begin{bmatrix} E_1^s \\ E_2^s \end{bmatrix} = [S] \begin{bmatrix} E_1^i \\ E_2^i \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \begin{bmatrix} E_1^i \\ E_2^i \end{bmatrix}. \quad \text{Eq. (14.21)}$$

The superscripts i and s denote incident and scattered fields. The quantities s_{ij} are in general complex, and the subscripts 1 and 2 represent any combination of orthogonal polarizations. More precisely, $1 = H, R$, and $2 = V, L$. From Eq. (14.3), the backscattered RCS is related to the scattering matrix components by the following relation:

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = 4\pi R^2 \begin{bmatrix} |s_{11}|^2 & |s_{12}|^2 \\ |s_{21}|^2 & |s_{22}|^2 \end{bmatrix} \quad \text{Eq. (14.22)}$$

It follows that once a scattering matrix is specified, the target backscattered RCS can be computed for any combination of transmitting and receiving polarizations. The reader is advised to see Ruck (1970) for ways to calculate the scattering matrix $[S]$.

Rewriting Eq. (14.22) in terms of the different possible orthogonal polarizations yields

$$\begin{bmatrix} E_H^s \\ E_V^s \end{bmatrix} = \begin{bmatrix} s_{HH} & s_{HV} \\ s_{VH} & s_{VV} \end{bmatrix} \begin{bmatrix} E_H^i \\ E_V^i \end{bmatrix} \quad \text{Eq. (14.23)}$$

$$\begin{bmatrix} E_R^s \\ E_L^s \end{bmatrix} = \begin{bmatrix} s_{RR} & s_{RL} \\ s_{LR} & s_{LL} \end{bmatrix} \begin{bmatrix} E_R^i \\ E_L^i \end{bmatrix}. \quad \text{Eq. (14.24)}$$

By using the transformation matrix $[T]$ in Eq. (14.19), the circular scattering elements can be computed from the linear scattering elements

$$\begin{bmatrix} s_{RR} & s_{RL} \\ s_{LR} & s_{LL} \end{bmatrix} = [T] \begin{bmatrix} s_{HH} & s_{HV} \\ s_{VH} & s_{VV} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} [T]^{-1} \quad \text{Eq. (14.25)}$$

and the individual components are

$$\begin{aligned} s_{RR} &= \frac{-s_{VV} + s_{HH} - j(s_{HV} + s_{VH})}{2} \\ s_{RL} &= \frac{s_{VV} + s_{HH} + j(s_{HV} - s_{VH})}{2} \\ s_{LR} &= \frac{s_{VV} + s_{HH} - j(s_{HV} - s_{VH})}{2} \\ s_{LL} &= \frac{-s_{VV} + s_{HH} + j(s_{HV} + s_{VH})}{2} \end{aligned} \quad \text{Eq. (14.26)}$$

Similarly, the linear scattering elements are given by

$$\begin{bmatrix} s_{HH} & s_{HV} \\ s_{VH} & s_{VV} \end{bmatrix} = [T]^{-1} \begin{bmatrix} s_{RR} & s_{RL} \\ s_{LR} & s_{LL} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} [T] \quad \text{Eq. (14.27)}$$

and the individual components are

$$\begin{aligned} s_{HH} &= \frac{s_{RR} - s_{RL} + s_{LR} - s_{LL}}{2} \\ s_{VH} &= \frac{j(s_{RR} - s_{LR} - s_{RL} + s_{LL})}{2} \\ s_{HV} &= \frac{-j(s_{RR} + s_{LR} + s_{RL} + s_{LL})}{2} \\ s_{VV} &= \frac{s_{RR} + s_{LL} - s_{RL} - s_{LR}}{2} \end{aligned} \quad \text{Eq. (14.28)}$$

14.4. RCS of Simple Objects

Electromagnetic wave scattering from simple objects has historically received a great amount of attention as analytic expressions since the scattered fields can often be derived. Among these are objects such as spheres and ellipsoids, and two-dimensional cylinders, half planes and wedges. The study of such shapes is of great value as they lend insight into the important scattering mechanisms inherent in wave interactions with real-world objects. These analytic scattering equations are also used to test and verify Computational Electromagnetic (CEM) software codes. Readers interested in these subjects should consider sources such as Bowman and Ruck, which summarize research into the scattering from such bodies.

This section presents a sample set of simple object radar cross section. Most of the expressions presented represent the radar cross section of the object when it is large compared to the wavelength. These are derived from analytic expressions, often series or complex-plane integrations, using asymptotic limits for wavelength or empirical fits to simplify their evaluation. These approximations are said to operate in the “high-frequency” or “optical” scattering

regime. Computational methods will be discussed later in this chapter. In this case, other methods in the "low-frequency" or "resonance" regime are used to calculate the RCS.

This section presents examples of a backscattered radar cross section for a number of simple shape objects. In all cases, except for the perfectly conducting sphere, only optical region approximations are presented. Radar designers and RCS engineers consider the perfectly conducting sphere to be the simplest target to examine. Even in this case, the complexity of the exact solution, when compared to the optical region approximation, is overwhelming. Most formulas presented are Physical Optics (PO) approximation for the backscattered RCS measured by a far field radar in the direction (θ, φ), as illustrated in Fig. 14.9. In this section, it is assumed that the radar is always illuminating an object from the positive *z*-direction.

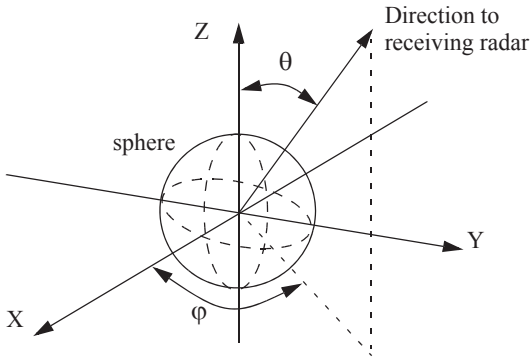


Figure 14.9. Direction of antenna receiving backscattered waves.

14.4.1. Sphere

Due to symmetry, waves scattered from a perfectly conducting sphere are co-polarized (have the same polarization) with the incident waves. This means that the cross-polarized backscattered waves are practically zero. For example, if the incident waves were Left Circularly Polarized (LCP), then the backscattered waves will also be LCP. However, because of the opposite direction of propagation of the backscattered waves, they are considered to be Right Circularly Polarized (RCP) by the receiving antenna. Therefore, the PP backscattered waves from a sphere are LCP, while the OP backscattered waves are negligible.

The normalized exact backscattered RCS for a perfectly conducting sphere is a Mie series given by

$$\frac{\sigma}{\pi r^2} = \left(\frac{j}{kr}\right)\sum_{n=1}^{\infty}(-1)^n(2n+1)\left[\left(\frac{krJ_n(kr)-nJ_n(kr)}{krH_{n-1}(kr)-nH_n^{(1)}(kr)}\right)-\left(\frac{J_n(kr)}{H_n^{(1)}(kr)}\right)\right] \qquad \textbf{Eq. (14.29)}$$

where r is the radius of the sphere, $k = 2\pi/\lambda$, λ is the wavelength, J_n is the spherical Bessel of the first kind of order n, and $H_n^{(1)}$ is the Hankel function of order n, and is given by

$$H_n^{(1)}(kr) = J_n(kr) + jY_n(kr)\,. \qquad \textbf{Eq. (14.30)}$$

Y_n is the spherical Bessel function of the second kind of order n. Plots of the normalized perfectly conducting sphere RCS as a function of its circumference in wavelength units are shown in Figs. 14.10a and 14.10b. These plots can be reproduced using the function *"rcs_sphere.m."* In Fig. 14.10, three regions are identified. First is the optical region (corresponds to a large sphere). In this case,

$$\sigma = \pi r^2 \qquad r \gg \lambda . \qquad \textbf{Eq. (14.31)}$$

Second is the Rayleigh region (small sphere). In this case,

$$\sigma \approx 9\pi r^2 (kr)^4 \qquad r \ll \lambda . \qquad \textbf{Eq. (14.32)}$$

The region between the optical and Rayleigh regions is oscillatory in nature and is called the Mie or resonance region.

The backscattered RCS for a perfectly conducting sphere is constant in the optical region. For this reason, radar designers typically use spheres of known cross sections to experimentally calibrate radar systems. For this purpose, spheres are flown attached to balloons. In order to obtain Doppler shift, spheres of known RCS are dropped out of an airplane and towed behind the airplane whose velocity is known to the radar.

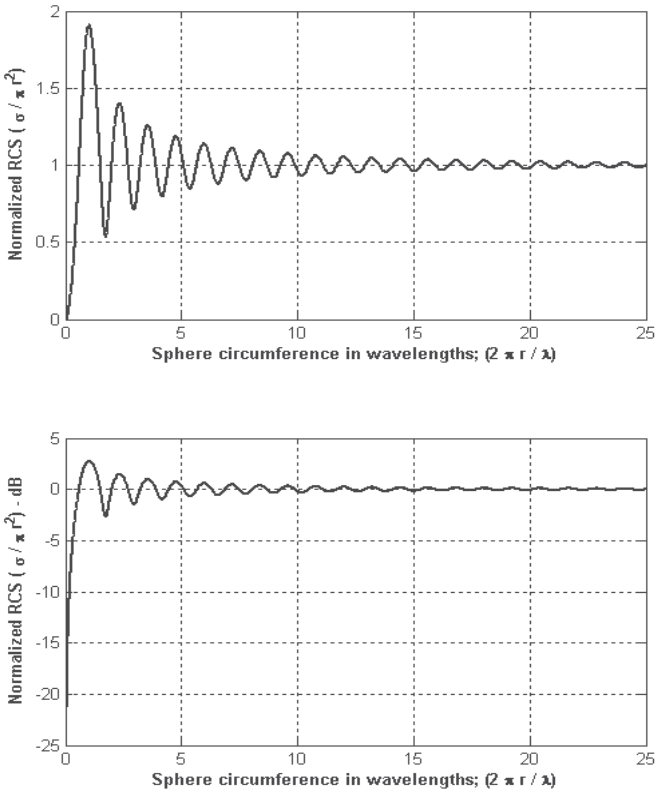


Figure 14.10a. Normalized backscattered RCS for a perfectly conducting sphere.

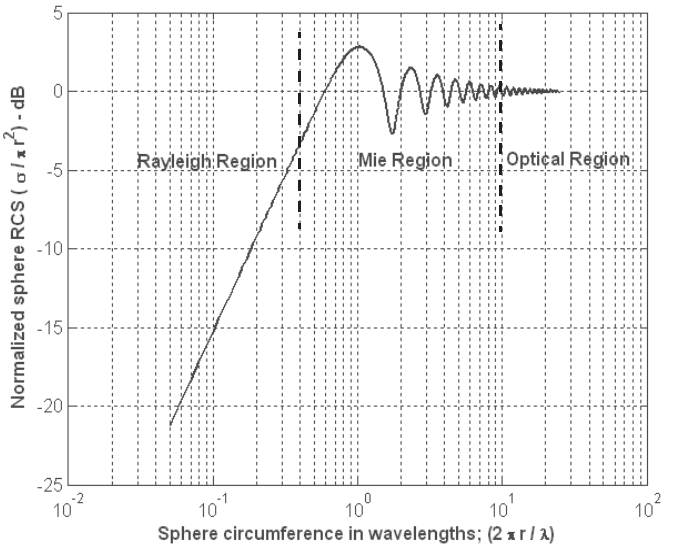


Figure 14.10b. Normalized backscattered RCS for a perfectly conducting sphere using semi-log scale.

14.4.2. Ellipsoid

An ellipsoid centered at (0,0,0) is shown in Fig. 14.11. It is defined by the following equation:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1 . \qquad \textbf{Eq. (14.33)}$$

One widely accepted approximation for the ellipsoid backscattered RCS is given by

$$\sigma = \frac{\pi a^2 b^2 c^2}{\left(a^2(\sin\theta)^2(\cos\varphi)^2 + b^2(\sin\theta)^2(\sin\varphi)^2 + c^2(\cos\theta)^2\right)^2} . \qquad \textbf{Eq. (14.34)}$$

When $a = b$, the ellipsoid becomes roll symmetric. Thus, the RCS is independent of φ, and Eq. (14.34) is reduced to

$$\sigma = \frac{\pi b^4 c^2}{\left(a^2(\sin\theta)^2 + c^2(\cos\theta)^2\right)^2} , \qquad \textbf{Eq. (14.35)}$$

and for the case when $a = b = c$,

$$\sigma = \pi c^2 . \qquad \textbf{Eq. (14.36)}$$

Note that Eq. (14.36) defines the backscattered RCS of a sphere. This should be expected, since under the condition $a = b = c$ the ellipsoid becomes a sphere.

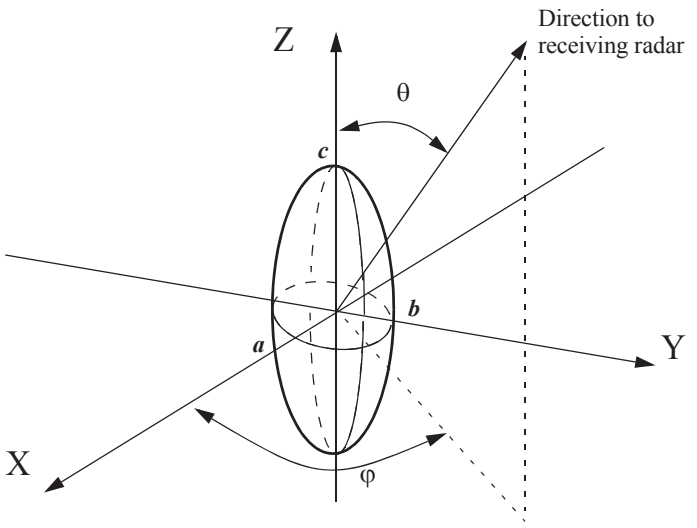


Figure 14.11. Ellipsoid.

MATLAB Function "rcs_ellipsoid.m"

The MATLAB function *"rcs_ellipsoid.m"* computes the RCS of an ellipsoid by implementing Eqs. (14.34) and (14.35). Its syntax is as follows:

[rcs] = rcs_ellipsoid (a, b, c, phi)

where

Symbol	Description	Units	Status
a	*ellipsoid a-radius*	*meters*	*input*
b	*ellipsoid b-radius*	*meters*	*input*
c	*ellipsoid c-radius*	*meters*	*input*
phi	*ellipsoid roll angle*	*degrees*	*input*
rcs	*array of RCS versus aspect angle*	*dBsm*	*output*

Figure 14.12a shows the backscattered RCS for an ellipsoid versus the aspect angle θ for $\varphi = 0°$, $\varphi = 45°$, and $\theta = 90°$. Note that at normal incidence ($\theta = 90°$), the RCS corresponds to that of a sphere of radius c, and is often referred to as the broadside specular RCS value. This figure can be reproduced using MATLAB program *"Fig14_12a.m,"* listed in Appendix 14-A.

A MATLAB-based graphical user interface (GUI) was developed for this purpose. Figure 14.12b shows the GUI workspace associated with the function. To execute this GUI, first download its MATLAB code from this book's web page on the CRC Press website, then in the MATLAB command window, type *"rcs_ellipsoid_gui."*

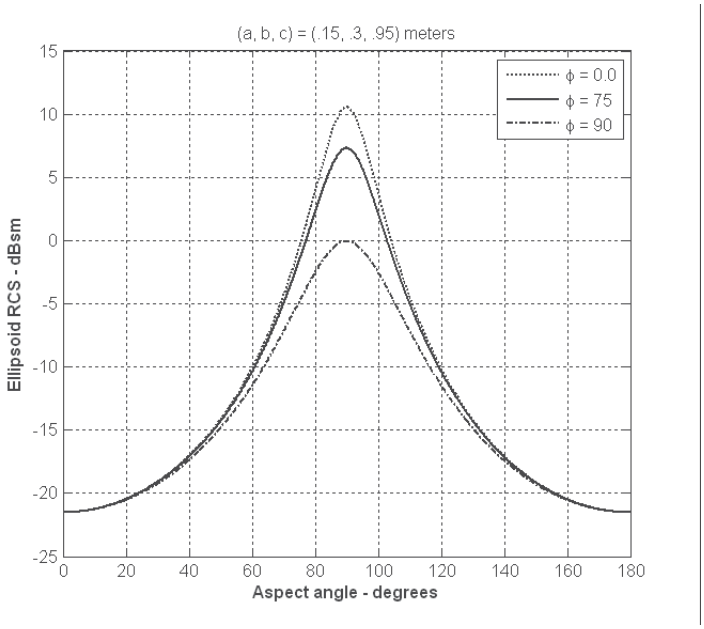


Figure 14.12a. Ellipsoid backscattered RCS versus aspect angle.

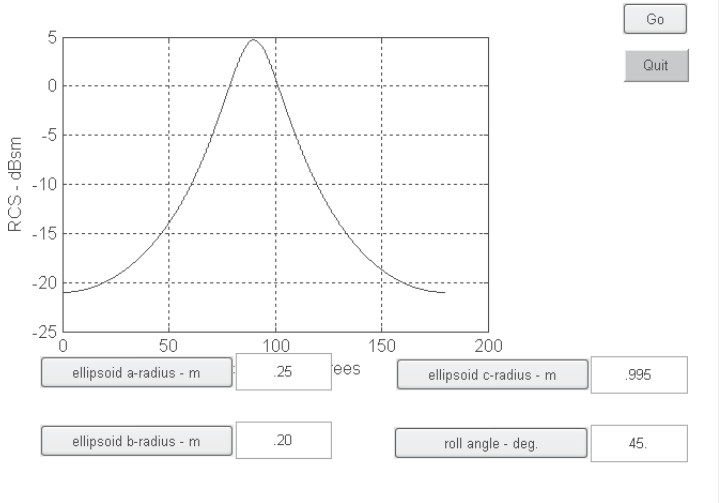


Figure 14.12b. GUI workspace associated with the function *"rcs_ellipsoid.m."*

14.4.3. Circular Flat Plate

Figure 14.13 shows a circular flat plate of radius r, centered at the origin. Due to the circular symmetry, the backscattered RCS of a circular flat plate has no dependency on φ. The RCS is only aspect angle dependent. For normal incidence (i.e., zero aspect angle), the backscattered RCS for a circular flat plate is

$$\sigma = \frac{4\pi^3 r^4}{\lambda^2} \qquad \theta = 0^\circ . \qquad \textbf{Eq. (14.37)}$$

For non-normal incidence, two approximations for the circular flat plate backscattered RCS for any linearly polarized incident wave are

$$\sigma = \frac{\lambda r}{8\pi \sin\theta (\tan(\theta))^2} \qquad \textbf{Eq. (14.38)}$$

$$\sigma = \pi k^2 r^4 \left(\frac{2J_1(2kr\sin\theta)}{2kr\sin\theta} \right)^2 (\cos\theta)^2 \qquad \textbf{Eq. (14.39)}$$

where $k = 2\pi/\lambda$, and J_1 is the first-order spherical Bessel function.

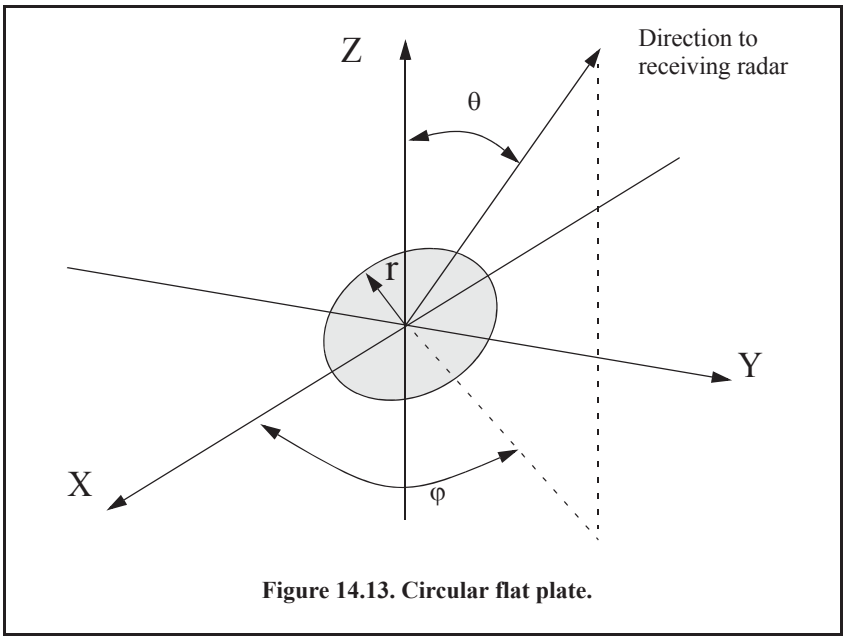


Figure 14.13. Circular flat plate.

MATLAB Function "rcs_circ_plate.m"

The function *"rcs_circ_plate.m"* calculates and plots the backscattered RCS from a circular plate. The syntax is as follows:

[rcs] = rcs_circ_plate (r, freq)

where

Symbol	Description	Units	Status
r	*radius of circular plate*	*meters*	*input*
freq	*frequency*	*Hz*	*input*
rcs	*array of RCS versus aspect angle*	*dBsm*	*output*

A MATLAB-based GUI was developed to implement this function. Figure 14.14 shows the GUI workspace associated with function and a typical output. To execute this GUI, first download its MATLAB code from this book's web page on the CRC Press website, then in the MATLAB command window type, *"rcs_circ_gui.m."*

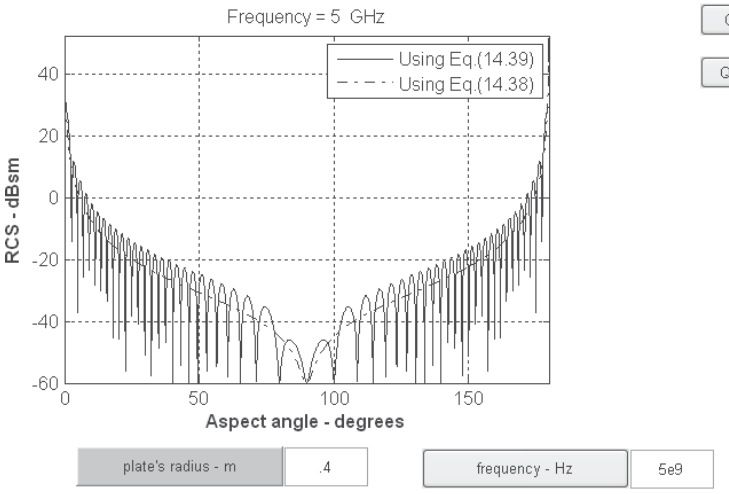


Figure 14.14. Backscattered RCS for a circular flat plate.

14.4.4. Truncated Cone (Frustum)

Figures 14.15 and 14.16 show the geometry associated with a frustum. The half cone angle α is given by

$$\tan\alpha = \frac{(r_2 - r_1)}{H} = \frac{r_2}{L}. \qquad \textbf{Eq. (14.40)}$$

Define the aspect angle at normal incidence with respect to the frustum's surface (broadside) as θ_n. Thus, when a frustum is illuminated by a radar located at the same side as the cone's small end, the angle θ_n is

$$\theta_n = 90° - \alpha. \qquad \textbf{Eq. (14.41)}$$

Alternatively, normal incidence occurs at

$$\theta_n = 90° + \alpha\,. \qquad \textbf{Eq. (14.42)}$$

At normal incidence, one approximation for the backscattered RCS of a truncated cone due to a linearly polarized incident wave is

$$\sigma_{\theta_n} = \frac{8\pi(z_2^{3/2} - z_1^{3/2})^2}{9\lambda \sin\theta_n} \tan\alpha(\sin\theta_n - \cos\theta_n \tan\alpha)^2 \qquad \textbf{Eq. (14.43)}$$

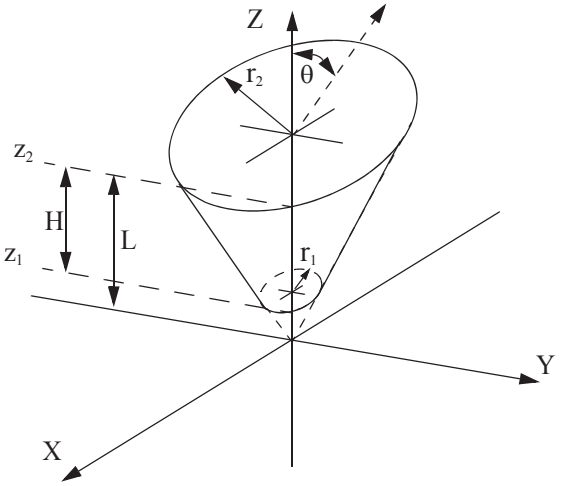


Figure 14.15. Truncated cone (frustum).

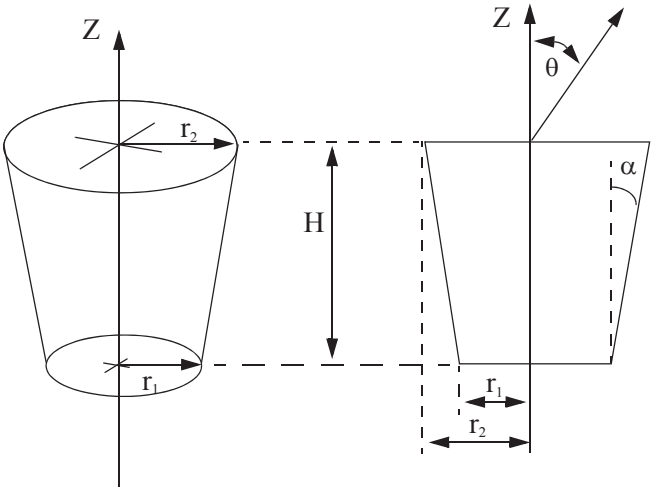


Figure 14.16. Definition of half cone angle.

$$\sigma_{\theta_n} = \frac{8\pi(z_2^{3/2} - z_1^{3/2})^2}{9\lambda} \frac{\sin\alpha}{(\cos\alpha)^4}. \quad \text{Eq. (14.44)}$$

For non-normal incidence, the backscattered RCS due to a linearly polarized incident wave is

$$\sigma = \frac{\lambda z \tan\alpha}{8\pi\sin\theta}\left(\frac{\sin\theta - \cos\theta\tan\alpha}{\sin\theta\tan\alpha + \cos\theta}\right)^2 \quad \text{Eq. (14.45)}$$

where z is equal to either z_1 or z_2, depending on whether the RCS contribution is from the small or the large end of the cone. Again, using trigonometric identities Eq. (14.45) (assuming the radar illuminates the frustum starting from the large end) is reduced to

$$\sigma = \frac{\lambda z \tan\alpha}{8\pi\sin\theta} (\tan(\theta - \alpha))^2 \quad \text{Eq. (14.46)}$$

where λ is the wavelength, and z_1, z_2 are defined in Fig. 14.15.

When the radar illuminates the frustum starting from the small end (i.e., the radar is in the negative z direction in Fig. 14.15), Eq. (14.46) should be modified to

$$\sigma = \frac{\lambda z \tan\alpha}{8\pi\sin\theta} (\tan(\theta + \alpha))^2. \quad \text{Eq. (14.47)}$$

MATLAB Function "rcs_frustum.m"

The function *"rcs_frustum.m"* computes and plots the backscattered RCS of a truncated conic section. The syntax is as follows:

[rcs] = rcs_frustum (r1, r2, freq, indicator)

where

Symbol	Description	Units	Status
r1	*small end radius*	*meters*	*input*
r2	*large end radius*	*meters*	*input*
freq	*frequency*	*Hz*	*input*
indicator	*indicator = 1 when viewing from large end* *indicator = 0 when viewing from small end*	*none*	*input*
rcs	*array of RCS versus aspect angle*	*dBsm*	*output*

For example, consider a frustum defined by $H = 20.945cm$, $r_1 = 2.057cm$, and $r_2 = 5.753cm$. It follows that the half cone angle is $10°$. Figure 14.17a shows a plot of its RCS when illuminated by a radar in the positive z direction. Figure 14.17b shows the same thing, except in this case, the radar is in the negative z direction. Note that for the first case, normal incidence occurs at $100°$, while for the second case it occurs at $80°$. A MATLAB-based GUI was developed to implement this function. To execute this GUI, first download the its MATLAB code from this book's web page on the CRC Press website, then in the MATLAB command window type, *"rcs_frustum_gui.m."*

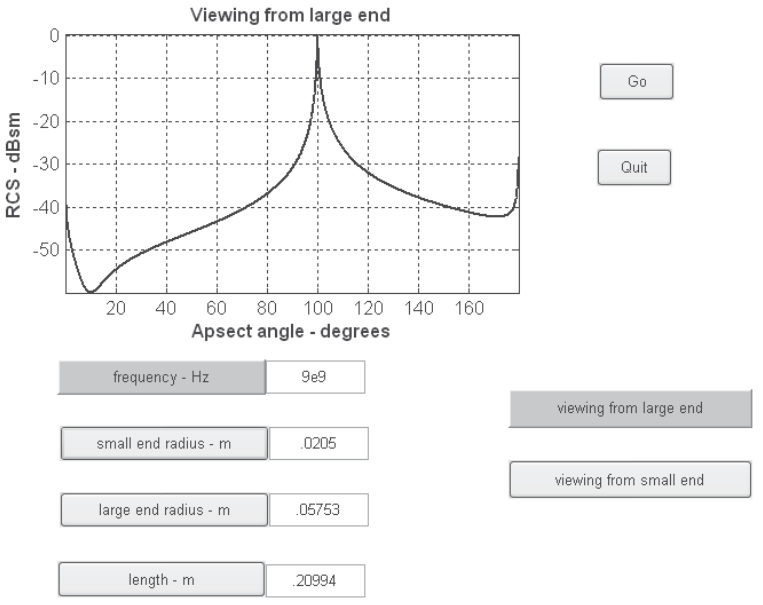


Figure 14.17a. Backscattered RCS for a frustum.

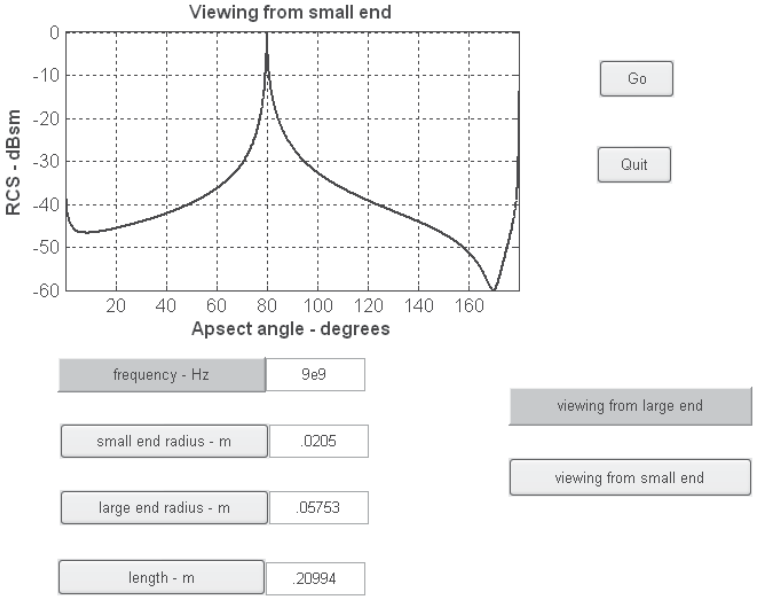


Figure 14.17b. Backscattered RCS for a frustum.

14.4.5. Cylinder

Figure 14.18 shows the geometry associated with a finite-length conducting cylinder. Two cases are presented: first, the general case of an elliptical cross section cylinder; second, the case of a circular cross section cylinder. The normal and non-normal incidence backscattered RCS due to a linearly polarized incident wave from an elliptical cylinder with minor and major radii being r_1 and r_2 are, respectively, given by

$$\sigma_{\theta_n} = \frac{2\pi H^2 r_2^2 r_1^2}{\lambda[r_1^2(\cos\varphi)^2 + r_2^2(\sin\varphi)^2]^{1.5}} \qquad \text{Eq. (14.48)}$$

$$\sigma = \frac{\lambda r_2^2 r_1^2 \sin\theta}{8\pi\ (\cos\theta)^2[r_1^2(\cos\varphi)^2 + r_2^2(\sin\varphi)^2]^{1.5}} \qquad \text{Eq. (14.49)}$$

For a circular cylinder of radius r, due to roll symmetry, Eqs. (14.48) and (14.49), respectively, reduce to

$$\sigma_{\theta_n} = \frac{2\pi H^2 r}{\lambda} \qquad \text{Eq. (14.50)}$$

$$\sigma = \frac{\lambda r \sin\theta}{8\pi(\cos\theta)^2}. \qquad \text{Eq. (14.51)}$$

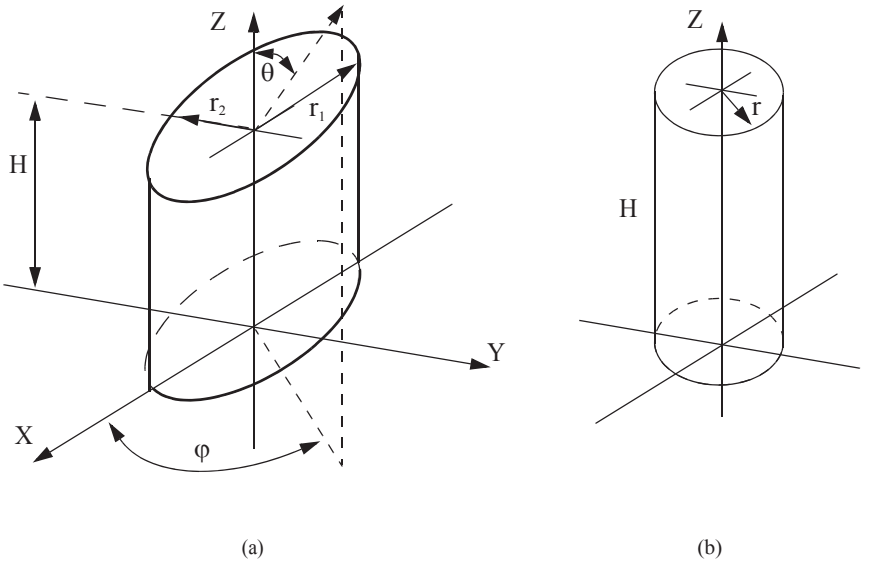


Figure 14.18. (a) Elliptical cylinder; (b) circular cylinder.

MATLAB Function "rcs_cylinder.m"

The function *"rcs_cylinder.m"* computes and plots the backscattered RCS of a cylinder. The syntax is as follows:

[rcs] = rcs_cylinder(r1, r2, h, freq, phi, CylinderType)

where

Symbol	Description	Units	Status
r1	*radius r1*	*meters*	*input*
r2	*radius r2*	*meters*	*input*
h	*length of cylinder*	*meters*	*input*
freq	*frequency*	*Hz*	*input*
phi	*roll viewing angle*	*degrees*	*input*
Cylinder Type	*"Circular," i.e.,* $r_1 = r_2$ *; "Elliptic," i.e.,* $r_1 \neq r_2$	*none*	*input*
rcs	*array of RCS versus aspect angle*	*dBsm*	*output*

Figure 14.19a shows a plot of the cylinder backscattered RCS for a symmetrical cylinder. Figure 14.19b shows the backscattered RCS for an elliptical cylinder. Figure 14.19 can be reproduced using the MATLAB program *"Fig14_19.m,"* listed in Appendix 14-A.

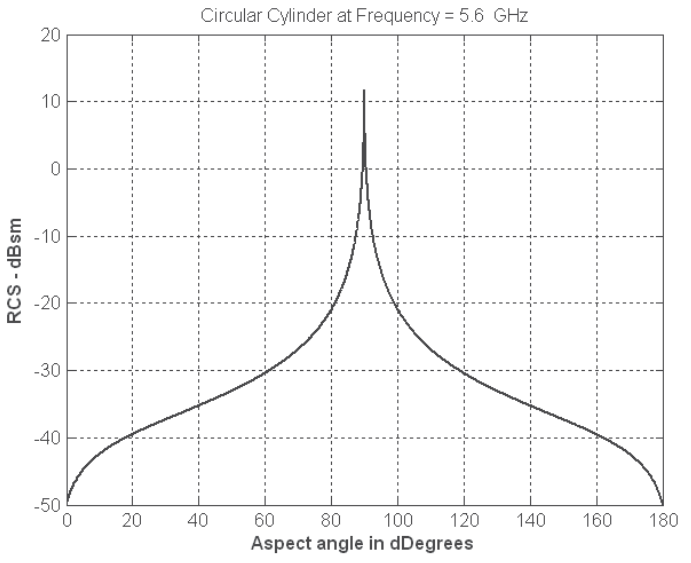


Figure 14.19a. Backscattered RCS for a symmetrical cylinder, $r = 0.125m$ **and** $H = 1m$**.**

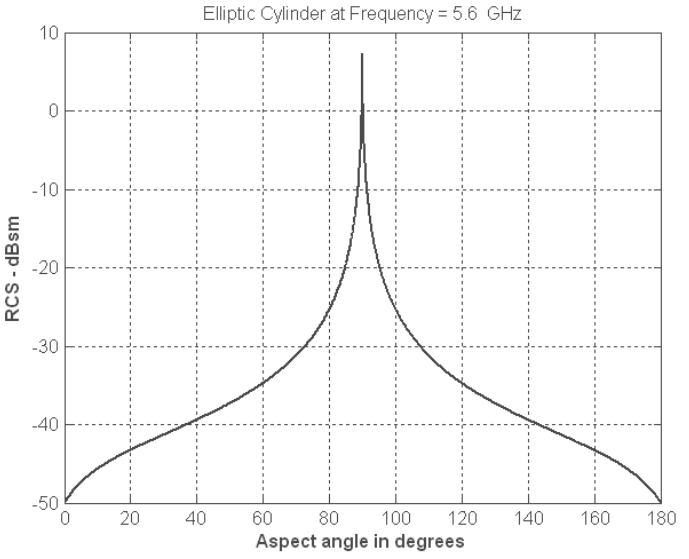


Figure 14.19b. Backscattered RCS for an elliptical cylinder, $r_1 = 0.125m$, $r_2 = 0.05m$, and $H = 1m$.

14.4.6. Rectangular Flat Plate

Consider a perfectly conducting rectangular thin flat plate in the *x-y* plane as shown in Fig. 14.20. The two sides of the plate are denoted by $2a$ and $2b$. For a linearly polarized incident wave in the *x-z* plane, the horizontal and vertical backscattered RCS are, respectively, given by

$$\sigma_V = \frac{b^2}{\pi}\left|\sigma_{1V} - \sigma_{2V}\left[\frac{1}{\cos\theta} + \frac{\sigma_{2V}}{4}(\sigma_{3V} + \sigma_{4V})\right]\sigma_{5V}^{-1}\right|^2 \qquad \textbf{Eq. (14.52)}$$

$$\sigma_H = \frac{b^2}{\pi}\left|\sigma_{1H} - \sigma_{2H}\left[\frac{1}{\cos\theta} - \frac{\sigma_{2H}}{4}(\sigma_{3H} + \sigma_{4H})\right]\sigma_{5H}^{-1}\right|^2 \qquad \textbf{Eq. (14.53)}$$

where $k = 2\pi/\lambda$ and

$$\sigma_{1V} = \cos(k\,a\sin\theta) - j\frac{\sin(k\,a\sin\theta)}{\sin\theta} = (\sigma_{1H})^* \qquad \textbf{Eq. (14.54)}$$

$$\sigma_{2V} = \frac{e^{j(ka - \pi/4)}}{\sqrt{2\pi}(ka)^{3/2}} \qquad \textbf{Eq. (14.55)}$$

$$\sigma_{3V} = \frac{(1 + \sin\theta)e^{-jk\,a\sin\theta}}{(1 - \sin\theta)^2} \qquad \textbf{Eq. (14.56)}$$

$$\sigma_{4V} = \frac{(1 - \sin\theta)e^{jk\,a\sin\theta}}{(1 + \sin\theta)^2} \qquad \textbf{Eq. (14.57)}$$

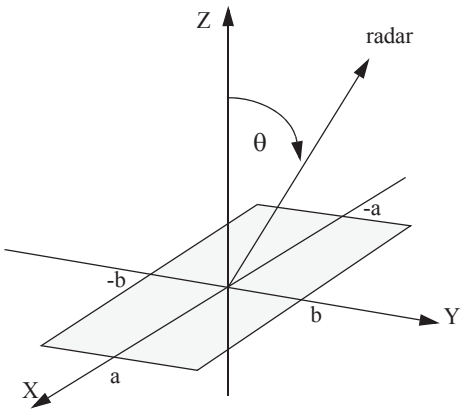


Figure 14.20. Rectangular flat plate.

$$\sigma_{5V} = 1 - \frac{e^{j(2ka - \pi/2)}}{8\pi(ka)^3} \qquad \textbf{Eq. (14.58)}$$

$$\sigma_{2H} = \frac{4e^{j(ka + \pi/4)}}{\sqrt{2\pi}(ka)^{1/2}} \qquad \textbf{Eq. (14.59)}$$

$$\sigma_{3H} = \frac{e^{-jk\,a\sin\theta}}{1 - \sin\theta} \qquad \textbf{Eq. (14.60)}$$

$$\sigma_{4H} = \frac{e^{jk\,a\sin\theta}}{1 + \sin\theta} \qquad \textbf{Eq. (14.61)}$$

$$\sigma_{5H} = 1 - \frac{e^{j(2ka + (\pi/2))}}{2\pi(ka)}. \qquad \textbf{Eq. (14.62)}$$

Equations (14.52) and (14.53) are valid and quite accurate for aspect angles $0° \le \theta \le 80$. For aspect angles near 90°, Ross[1] obtained, by extensive fitting of measured data, an empirical expression for the RCS. It is given by

$$\sigma_H \to 0$$

$$\sigma_V = \frac{ab^2}{\lambda}\left\{\left[1 + \frac{\pi}{2(2a/\lambda)^2}\right] + \left[1 - \frac{\pi}{2(2a/\lambda)^2}\right]\cos\left(2ka - \frac{3\pi}{5}\right)\right\}. \qquad \textbf{Eq. (14.63)}$$

The backscattered RCS for a perfectly conducting thin rectangular plate for incident waves at any θ, φ, can be approximated by

$$\sigma = \frac{4\pi a^2 b^2}{\lambda^2}\left(\frac{\sin(ak\sin\theta\cos\varphi)}{ak\sin\theta\cos\varphi}\ \frac{\sin(bk\sin\theta\sin\varphi)}{bk\sin\theta\sin\varphi}\right)^2(\cos\theta)^2 \qquad \textbf{Eq. (14.64)}$$

1. Ross, R. A., Radar Cross Section of Rectangular Flat Plate as a Function of Aspect Angle, *IEEE Trans*., AP-14, 320, 1966.

Note that, Eq. (14.64) is independent of the polarization, and is only valid for aspect angles $\theta \leq 20°$.

MATLAB Function "rcs_rect_plate.m"

The function *"rcs_rect_plate.m"* calculates and plots the backscattered RCS of a rectangular flat plate. Its syntax is as follows:

[rcs] = rcs_rect_plate (a, b, freq)

where

Symbol	Description	Units	Status
a	*short side of plate*	*meters*	*input*
b	*long side of plate*	*meters*	*input*
freq	*frequency*	*Hz*	*input*
rcs	*array of RCS versus aspect angle*	*dBsm*	*output*

Figure 14.21 shows an example for the backscattered RCS of a rectangular flat plate, for both vertical (Fig. 14.21a) and horizontal (Fig. 14.21b) polarizations, using Eqs. (14.52), (14.53), and (14.64). In this example, $a = b = 10.16cm$ and wavelength $\lambda = 3.33cm$. This plot can be reproduced using MATLAB function *"rcs_rect_plate."* Figure 14.21c shows the GUI workspace associated with this function.

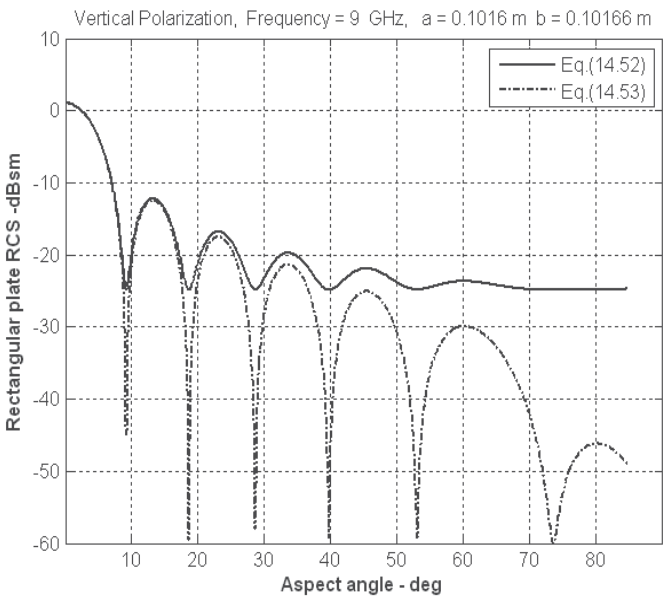


Figure 14.21a. Backscattered RCS for a rectangular flat plate.

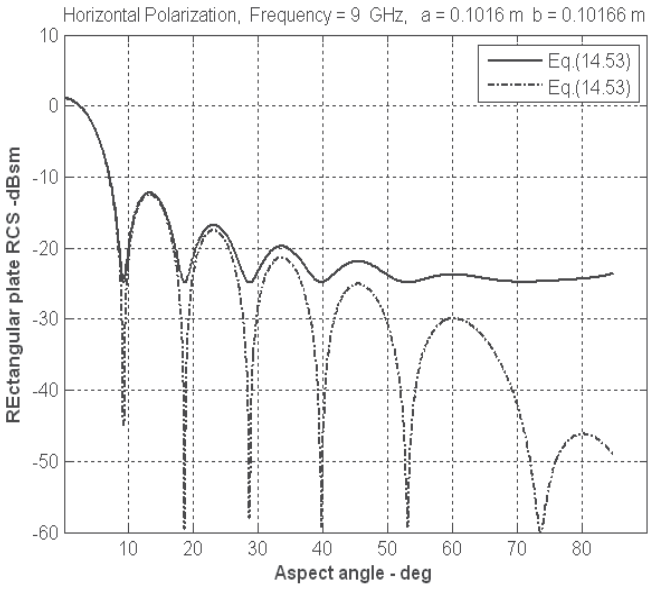


Figure 14.21b. Backscattered RCS for a rectangular flat plate.

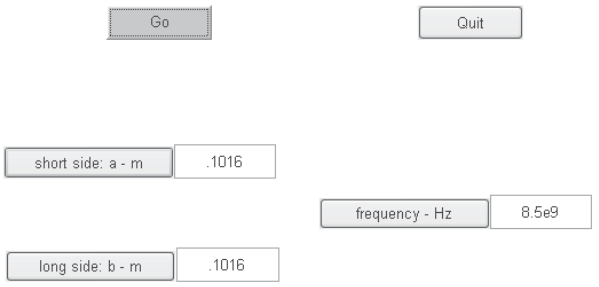


Figure 14.21c. GUI workspace associated with the function *"rcs_rect_plate.m."*

14.4.7. Triangular Flat Plate

Consider the triangular flat plate defined by the isosceles triangle as oriented in Fig. 14.22. The backscattered RCS can be approximated for small aspect angles ($\theta \leq 30°$) by

$$\sigma = \frac{4\pi A^2}{\lambda^2}(\cos\theta)^2\sigma_0 \qquad \textbf{Eq. (14.65)}$$

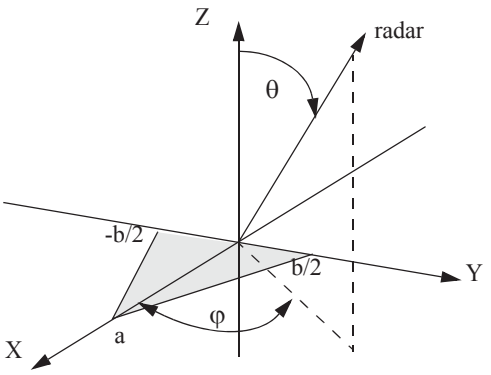


Figure 14.22. Coordinates for a perfectly conducting isosceles triangular plate.

$$\sigma_0 = \frac{[(\sin\alpha)^2 - (\sin(\beta/2))^2]^2 + \sigma_{01}}{\alpha^2 - (\beta/2)^2} \qquad \textbf{Eq. (14.66)}$$

$$\sigma_{01} = 0.25(\sin\varphi)^2[(2a/b)\cos\varphi\sin\beta - \sin\varphi\sin 2\alpha]^2 \qquad \textbf{Eq. (14.67)}$$

where $\alpha = k\text{asin}\theta\cos\varphi$, $\beta = kb\sin\theta\sin\varphi$, and $A = ab/2$. For waves incident in the plane $\varphi = 0$, the RCS reduces to

$$\sigma = \frac{4\pi A^2}{\lambda^2}(\cos\theta)^2\left[\frac{(\sin\alpha)^4}{\alpha^4} + \frac{(\sin 2\alpha - 2\alpha)^2}{4\alpha^4}\right], \qquad \textbf{Eq. (14.68)}$$

and for incidence in the plane $\varphi = \pi/2$,

$$\sigma = \frac{4\pi A^2}{\lambda^2}(\cos\theta)^2\left[\frac{(\sin(\beta/2))^4}{(\beta/2)^4}\right]. \qquad \textbf{Eq. (14.69)}$$

MATLAB Function "rcs_isosceles.m"

The function *"rcs_isosceles.m"* calculates and plots the backscattered RCS of a triangular flat plate. Its syntax is as follows:

[rcs] = rcs_isosceles (a, b, freq, phi)

where

Symbol	Description	Units	Status
a	*height of plate*	*meters*	*input*
b	*base of plate*	*meters*	*input*
freq	*frequency*	*Hz*	*input*
phi	*roll angle*	*degrees*	*input*
rcs	*array of RCS versus aspect angle*	*dBsm*	*output*

Figure 14.23 shows a plot for the normalized backscattered RCS from a perfectly conducting isosceles triangular flat plate. In this example $a = 0.2m$, $b = 0.75m$. This plot can be reproduced using MATLAB GUI *"rcs_isosceles_gui.m."*

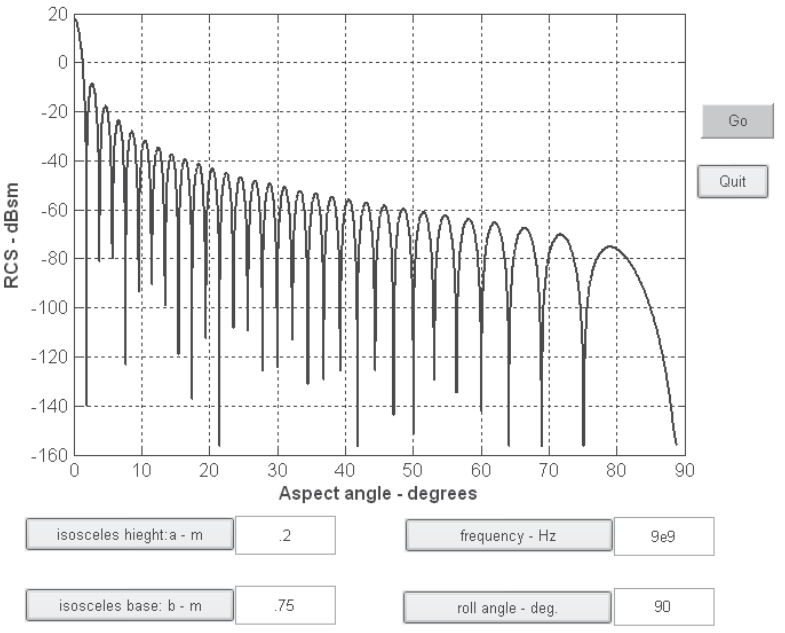


Figure 14.23. Backscattered RCS for a perfectly conducting triangular flat plate, $a = 20cm$ and $b = 75cm$.

14.5. RCS of Complex Objects

A complex target RCS is normally computed by coherently combining the cross sections of the simple shapes that make that target. In general, a complex target RCS can be modeled as a group of individual scattering centers distributed over the target. The scattering centers can be modeled as isotropic point scatterers (N-point model) or as simple shape scatterers (N-shape model). In any case, knowledge of the scattering centers' locations and strengths is critical in determining complex target RCS. This is true because as seen in Section 14.3, relative spacing and aspect angles of the individual scattering centers drastically influence the overall target RCS. Complex targets that can be modeled by many equal scattering centers are often called Swerling 1 or 2 targets. Alternatively, targets that have one dominant scattering center and many other smaller scattering centers are known as Swerling 3 or 4 targets.

In narrowband (NB) radar applications, contributions from all scattering centers combine coherently to produce a single value for the target RCS at every aspect angle. However, in wideband (WB) applications, a target may straddle many range bins. For each range bin, the average RCS extracted by the radar represents the contributions from all scattering centers that fall within that bin.

As an example, consider a circular cylinder with two perfectly conducting circular flat plates on both ends. Assume linear polarization and let $H = 1m$ and $r = 0.125m$. The backscattered RCS for this object versus aspect angle is shown in Fig. 14.24. Note that at aspect angles close to $0°$ and $180°$, the RCS is mainly dominated by the circular plate, while at aspect angles close to normal incidence, the RCS is dominated by the cylinder broadside specular return. The reader can reproduce this plot using the MATLAB program *"rcs_cylinder_complex.m,"* listed in Appendix 14-A.

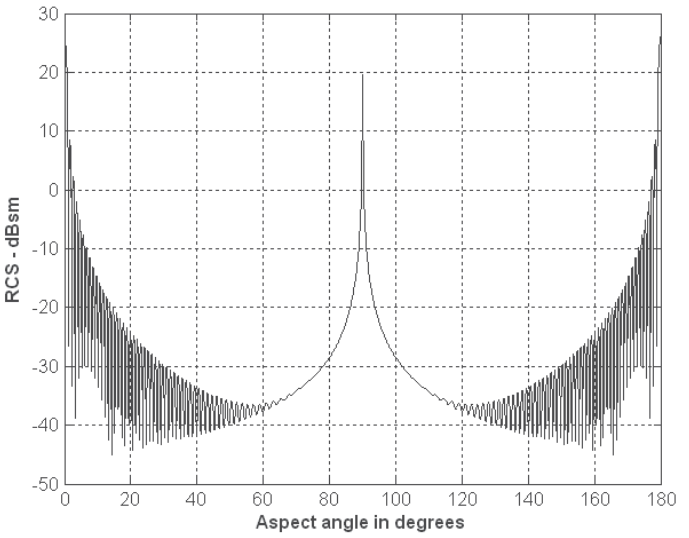


Figure 14.24. Backscattered RCS for a cylinder with flat plates.

14.6. RCS Prediction Methods

Before presenting the different RCS calculation methods, it is important to understand the significance of RCS prediction. Most radar systems use RCS as a means of discrimination. Therefore, accurate prediction of target RCS is critical in order to design and develop robust discrimination algorithms. Additionally, measuring and identifying the scattering centers (sources) for a given target aid in developing RCS reduction techniques. Another reason of lesser importance is that RCS calculations require broad and extensive technical knowledge; thus, many scientists and scholars find the subject challenging and intellectually motivating. Two categories of RCS prediction methods are available: exact and approximate.

Exact methods of RCS prediction are very complex, even for simple shape objects. This is because they require solving either differential or integral equations that describe the scattering problem under the proper set of boundary conditions. Such boundary conditions are governed by Maxwell's equations. Even when exact solutions are achievable, they are often difficult to interpret and to program using digital computers.

Due to the difficulties associated with the exact RCS prediction, approximate methods become the viable alternative. The majority of the approximate methods are valid in the optical

region, and each has its own strengths and limitations. Most approximate methods can predict RCS within a few *dB* of the truth. In general, such a variation is quite acceptable to radar engineers and designers. Approximate methods are usually the main source for predicting the RCS of complex and extended targets such as aircrafts, ships, and missiles. When experimental results are available, they can be used to validate and verify the approximations.

Some of the most commonly used approximate methods are Geometrical Optics (GO), Physical Optics (PO), Geometrical Theory of Diffraction (GTD), Physical Theory of Diffraction (PTD), and Method of Equivalent Currents (MEC). Interested readers may consult Knott or Ruck for more details on these and other approximate methods.

14.6.1. Computational Electromagnetics

Most scattering problems involve radar targets with very complicated shapes. Among these are ground-based targets such as trucks, tanks, and artillery; air targets such as aircraft, helicopters, and missiles; and space-based targets, such as reentry vehicles and satellites. For such an object, there is generally no analytic method available to predict the radar cross section. The field of Computational Electromagnetics (CEM) uses the power of a computer to implement Maxwell's Equations and solve these problems. CEM has applications in other areas, too, such as antennas and waveguide design, wave propagation, and medical imaging.

There exist many CEM techniques to solve scattering problems, each employing a different numerical analysis technique. Among the most popular methods used are the Finite Difference Time Domain (FDTD) method, the Finite Element Method (FEM), integral equation methods such as the Method of Moments (MoM), and asymptotic techniques such as Physical Optics (PO), the Physical Theory of Diffraction (PTD), and Shooting and Bouncing Rays (SBR).

14.6.2. Finite Difference Time Domain Method

The Finite Difference Time Domain (FDTD) method is useful for solving scattering problems involving objects composed of complex, often inhomogeneous media. It uses a finite difference scheme to discretize Maxwell's equations in the time domain. This has the advantage of allowing waveforms with wide bandwidths to be used as an excitation.

The main drawbacks for the FDTD method include the requirements on the grid size and non-conformal grid shape, which often results in poor discretization of target geometry and high memory requirements, particularly in three-dimensional cases. The object and its adjacent region must be discretized, and an artificial absorbing layer used to truncate the grid to simulate an unbounded space. It is also challenging to create a purely planar wave in such simulations.

The FDTD method makes use of finite difference approximations to directly discretize Maxwell's equations in the time domain. Consider the "forward difference" approximation for the first derivative:

$$\dot{f}(x_o) \approx \frac{f(x_o + \Delta x) - f(x_o)}{\Delta x}. \qquad \textbf{Eq. (14.70)}$$

The backward difference approximation is

$$\dot{f}(x_o) \approx \frac{f(x_o) - f(x_o - \Delta x)}{\Delta x}. \qquad \textbf{Eq. (14.71)}$$

The central difference approximation is

$$\dot{f}(x_o) \approx \frac{f(x_o + \Delta x) - f(x_o - \Delta x)}{2\Delta x}. \qquad \textbf{Eq. (14.72)}$$

Second derivatives can be approximated by a similar procedure

$$\ddot{f}(x_o) = \frac{[f(x_o + \Delta x) - f(x_o)] - [f(x_o) - f(x_o) - \Delta x]}{(\Delta x)^2}, \qquad \textbf{Eq. (14.73)}$$

where the second-order derivative makes use of forward and backward first derivatives.

Next consider an example of using FDTD to implement two-dimensional simulation. First, consider the time domain form of Maxwell's equations in a charge and conductive-free region

$$\nabla \times \vec{E} = -\mu \frac{\partial}{\partial t} \vec{H} \qquad \textbf{Eq. (14.74)}$$

$$\nabla \times \vec{H} = -\varepsilon \frac{\partial}{\partial t} \vec{E} + \vec{J} \qquad \textbf{Eq. (14.75)}$$

$$\nabla \cdot \vec{D} = 0 \qquad \textbf{Eq. (14.76)}$$

$$\nabla \cdot \vec{B} = 0 \qquad \textbf{Eq. (14.77)}$$

where $\vec{E}$ is the electric field intensity in volts/meter, $\vec{H}$ is the magnetic field intensity in ampere/meter^2, $\vec{J}$ is the current density in coulombs/meter^3, $\vec{D}$ is the displacement flux in coulombs/meter^2, $\vec{B}$ is the magnetic induction flux in Tesla or Weber/meter^2, μ is the permeability, and ε is the permittivity. Note that the region may comprise several homogeneous areas, each with its own μ and ε.

In rectangular coordinates, one can write these equations as

$$\mu \frac{\partial H_x}{\partial t} = \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} \qquad \textbf{Eq. (14.78)}$$

$$\mu \frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} \qquad \textbf{Eq. (14.79)}$$

$$\mu \frac{\partial H_z}{\partial t} = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \qquad \textbf{Eq. (14.80)}$$

$$\varepsilon \frac{\partial E_x}{\partial t} = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - \vec{J}_x \qquad \textbf{Eq. (14.81)}$$

$$\varepsilon \frac{\partial E_y}{\partial t} = \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} - \vec{J}_y \qquad \textbf{Eq. (14.82)}$$

$$\varepsilon \frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \vec{J}_z. \qquad \textbf{Eq. (14.83)}$$

Consider a filamentary current source $\vec{J}$ with direction $\hat{z}$, which excites TM-polarized (E_z) waves only. The above three equations then reduce to

$$\varepsilon \frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \vec{J}_z \qquad \text{Eq. (14.84)}$$

$$\mu \frac{\partial H_x}{\partial t} = -\frac{\partial E_z}{\partial y} \qquad \text{Eq. (14.85)}$$

$$\mu \frac{\partial H_y}{\partial t} = -\frac{\partial E_z}{\partial x} \qquad \text{Eq. (14.86)}$$

These expressions can be discretized using a 2-D Yee Algorithm.[1] In this scheme, the electric and magnetic fields are arranged on grids that are a half point in distance and time away from each other.

First-order derivatives may be applied to the above equations to obtain values at these time and grid points. More precisely,

$$E_{z,i,j}^{t+1/2} = E_{z,i,j}^{t-1/2} + \frac{\Delta t}{\varepsilon_{i,j}} \left\{ \left(\frac{1}{\Delta x} [H_{y,i+1/2,j}^{t} - H_{y,i-1/2,j}^{t}] \right) \right. \qquad \text{Eq. (14.87)}$$

$$\left. - \frac{1}{\Delta y} [H_{x,i+1/2,j}^{t} - H_{x,i-1/2,j}^{t}] \right\} - J_{z,i,j}^{t}$$

$$H_{x,i,j+1/2}^{t+1} = H_{x,i,j+1/2}^{t} + \frac{\Delta t}{\mu_{i,j+1/2} \Delta y} [E_{z,i,j+1}^{t+1/2} - E_{z,i,j}^{t+1/2}] \qquad \text{Eq. (14.88)}$$

$$H_{y,i,j+1/2}^{t+1} = H_{y,i+1/2,j}^{t} + \frac{\Delta t}{\mu_{i+1,j} \Delta x} [E_{z,i+1,j}^{t+1/2} - E_{z,i,j}^{t+1/2}] . \qquad \text{Eq. (14.89)}$$

As an example, consider a two-dimensional box with width and height of 2 meters. In the center of the box, place a $\hat{z}$ directed current source and assign to it the excitation function,

$$J(t) = \left(4 \left(\frac{t}{\tau} \right)^3 - \left(\frac{t}{\tau} \right)^4 \right) e^{-t/\tau} \qquad \text{Eq. (14.90)}$$

where $\tau = 1/(4\pi f_o)$ and $f_o = 600 MHz$. Place a 1×0.2 meter dielectric slab with $\varepsilon = 8\varepsilon_o$ at (1.0, 0.7) to introduce an obstruction to the spreading wavefront. The current J_z is shown in Figs. 14.25 and 14.26 at different times. Note that the wave slows down inside the slab by a factor of $\sqrt{8}$, and hence the wavelength is compressed. The wavefront starts penetrating the slab at time $t = 2ns$. At time $t = 3.36ns$ the wave starts to leave the dielectric slab and the unobstructed wavefronts reach the walls of the box.

The MATLAB program *"fdtd.m"* was developed to simulate this example. It is listed in Appendix 14-A. Readers are strongly advised to run this program and observe how the wave front spreads through the dielectric slab.

1. Taflove, Allen, *Computational Electromagnetic: The Finite-Difference-Time-Domain Method*, Artech House, 1995.

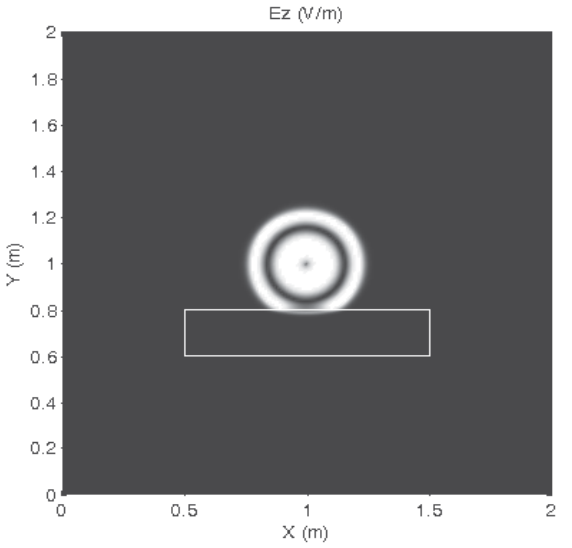


Figure 14.25. The wavefront at time $t = 0.2ns$.

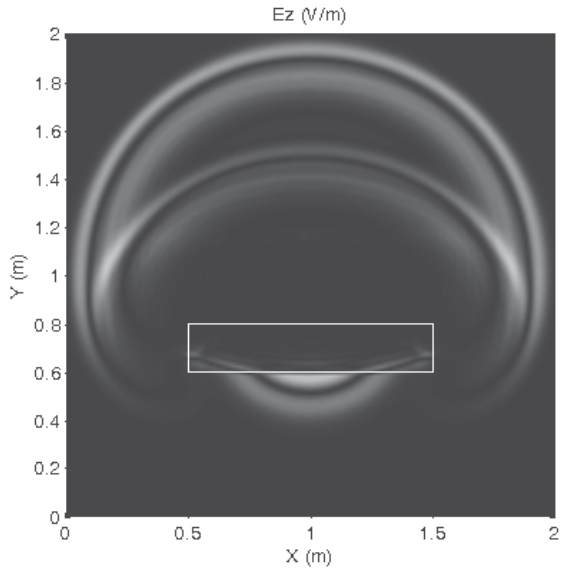


Figure 14.26. The wavefront at time $t = 3.36ns$.

14.6.3. Finite Element Method

The Finite Element Method (FEM) is a popular CEM technique for solving boundary-valued problems. In this method, the problem is formulated in terms of a variational expression, or functional, which has a minimum corresponding to the governing differential equation under the given boundary conditions. A trial function comprising a set of weighted basis functions is assigned to the unknown quantity in the region of study, typically the electric or magnetic field distribution. A matrix equation is then formed that can be solved for the unknown coefficients. In FEM, the basis functions are typically assigned to surface or volume elements, such as triangles or tetrahedrons. The coefficients are usually defined at the vertices (or edges in some vector FEM formulations).

The FEM is attractive for solving both static and time-harmonic electromagnetic problems, as well as eigenvalue problems such as determining the fundamental modes in a waveguide of arbitrary cross section. Most regions are easily discretized into triangular or quadrilateral elements, which conform very well to object boundaries and dielectric interfaces. Like the FDTD method however, the object and the adjoining space must be discretized, requiring an absorbing boundary condition imposed at the terminating boundary if an unbounded radiation problem is to be studied.

14.6.4. Integral Equations

There exists a set of auxiliary scattering equations that assists the solution of scattering problems in unbounded regions. One of the most widely used is the frequency-domain magnetic vector potential $\vec{A}$, derived from Maxwell's equations,

$$\vec{A} = \frac{\mu}{4\pi}\int\int_{S} \vec{J}\ \frac{e^{-jkR}}{R} ds \qquad \text{Eq. (14.91)}$$

where $\vec{J}$ is an electric surface current, and S is the surface in free space on which the current resides. Using this definition, the scattered electric field at all points in space is given by the well-known Electric Field Integral Equation (EFIE)

$$\vec{E_s} = -j\omega\vec{A} - \frac{j}{\varepsilon\mu\omega}\nabla\nabla\cdot\vec{A}\,. \qquad \text{Eq. (14.92)}$$

This equation relates the scattered field $\vec{E_s}$ to a known current $\vec{J}$. In a general scattering problem, it is typically the incident electric field that is known, and the surface current and scattered field that are the unknowns. If we assume a conducting surface for the currents, the tangential electric field must vanish, producing

$$\vec{E_i}^{\tan} = -\vec{E_s}^{\tan}\,. \qquad \text{Eq. (14.93)}$$

The EFIE can be rewritten using the known incident field $\vec{E_i}$

$$-\vec{E_s}^{\tan} = \left(-j\omega\vec{A} - \frac{j}{\varepsilon\mu\omega}\nabla\nabla\cdot\vec{A}\right)^{\tan}\,. \qquad \text{Eq. (14.94)}$$

This represents an integral equation for the unknown current $\vec{J}$.

The Method of Moments (MoM) is a technique used to solve such integral equations, and has received much attention in the last 30 years. In using the MoM to solve the EFIE, Max-

well's equations are represented exactly, and the solution is described as “exact” or “full wave.” This means that all electromagnetic effects and dominant scattering mechanisms are represented in the result. To solve the EFIE, the current is usually discretized according to

$$\vec{J} = \sum_{n=1}^{N} I_n \vec{f}_n(\vec{r}) \qquad \text{Eq. (14.95)}$$

where the $\vec{f}_n(\vec{r})$ are basis functions chosen to represent the behavior of the current, and the I_n are unknown coefficients. The target surface is typically broken up into small subdomains and a basis function assigned to each. Inserting Eq. (14.95) into Eq. (14.93) yields a single equation in N unknowns.

To create N equations in N unknowns, the EFIE is tested or enforced over all subdomains by employing an inner product of Eq. (14.93) by a set of testing functions. Most often, the basis functions $\vec{f}_n(\vec{r})$ are used (the Galerkin method). The resulting system may then be solved for the unknown coefficients by Gaussian elimination, or an iterative technique.

The MoM has been used extensively to solve scattering problems involving rotationally symmetric objects. The McDonnell Douglas[1] code (CICERO), solves the body-of-revolution scattering problem for objects that have various conducting and dielectric coated surfaces. The MoM has also been applied to three-dimensional bodies, and this is often done according to the method proposed by Rao, Wilton, and Glisson, who introduced a basis function suitable for use with surfaces described by connected triangular patches.

While the MoM achieves excellent accuracy, the size of the matrix system is proportional to the square of the radar wavelength. Until recently this has limited the maximum object size that could be stored in system memory, typically a few wavelengths at most for three-dimensional problems. Methods that approximate the system’s Green’s Function have been developed in recent years in an attempt to reduce the required memory. The Adaptive Integral Method (AIM) and the Fast Multipole Method (FMM) are two methods that were developed to alleviate this problem. The FMM has proved quite successful and is used in the Fast Illinois Solver Code (FISC) at the University of Illinois.

In the next few sections a brief discussion of asymptotic, or so, called high-frequency techniques is presented.

14.6.5. Geometrical Optics

The method of Geometrical Optics (GO) treats the radar energy as small ray tubes that propagate according to Fermat’s Principle. The specular reflection points on the target are found and divergence and spreading of energy are accounted for by analyzing the radii of curvature at the reflection points. GO is limited by its applicability at caustic points, and it does not handle diffractions from tips and edges, or account for creeping waves. Keller introduced the Geometrical Theory of Diffraction (GTD) in an attempt to handle diffraction effects; however, GTD suffers from the same problem at caustics and shadow boundaries. The Uniform Theory of Diffraction (UTD) was introduced to further improve this method.

1. Medgyesi-Mitschang, Louis and Putnam, John, Electromagnetic Scattering from Axially Inhomogeneous Bodies of Revolution, *IEEE Trans. Antennas Propagation,* Vol. 32, pp.707-806, August 1984.

14.6.6. Physical Optics

Physical Optics (PO) is a technique that approximates that surface current $\vec{J}$ in the illuminated portion of the target by assuming that locally, the target's surface can be considered flat and planar. If at each point the surface is considered to be an infinite half plane, image theory allows the surface current to be written directly in terms of the incident magnetic field and the local surface normal,

$$\vec{J} = 2\hat{n} \times \vec{H_i}. \qquad \text{Eq. (14.96)}$$

This is called the physical optics approximation. With the current known, the scattered electric field is obtained directly via Eq. (14.91).

While the PO method is used extensively in high-frequency CEM computations, it is limited in its accuracy to near-specular observations. It does not treat diffractions, traveling or creeping waves, multiple bounces, or other scattering phenomena. These mechanisms are often supplemented in the PO solution by other techniques, some of which are discussed later.

Rectangular Plate

Consider a rectangular plate of length b and width a in the *xy*-plane. The $\hat{\theta}$ polarized incident electric and magnetic fields on the plate are, respectively, given by

$$\vec{E_i} = \hat{\theta} e^{jk(\hat{r}_i \cdot \vec{r})} \qquad \text{Eq. (14.97)}$$

$$\vec{H_i} = -\frac{\hat{r}_i}{z_o} \times \vec{E_i} = \left(-\frac{2}{z_o}\right) \hat{\phi} e^{jk(\hat{r}_i \cdot \vec{r})} \qquad \text{Eq. (14.98)}$$

where $\hat{r}_i$ is the direction on incidence, and z_o is the free space impedance. This field generates the current

$$\vec{J} = \frac{2}{z_o} \hat{\phi} \times \hat{n} e^{jk(\hat{r}_i \cdot \vec{r})}, \qquad \text{Eq. (14.99)}$$

resulting in magnetic vector potential given by

$$\vec{A} = \frac{\mu}{2\pi z_o} \frac{e^{-jkR_s}}{R_s} \int_{-a/2}^{a/2} \int_{a/2}^{-b/2} \vec{J} e^{jk(\hat{r}_i \cdot \vec{r})} \, dxdy = \qquad \text{Eq. (14.100)}$$

$$\frac{\mu}{2\pi z_o} \frac{e^{-jkR_s}}{R_s} \hat{\phi} \times \hat{n} \int_{-a/2}^{a/2} \int_{-b/2}^{b/2} e^{2jk(\hat{r}_i \cdot \vec{r})} \, dxdy$$

where $\vec{r} = x\hat{x} + y\hat{y}$, and for phase variation, the range R_s is approximated as

$$R_s - (\hat{r}_i \cdot \vec{r}). \qquad \text{Eq. (14.101)}$$

Evaluating Eq. (14.100) analytically yields

$$\vec{A} = \frac{\mu}{2\pi z_o} \frac{e^{-jkR_s}}{R_s} \hat{\phi} \times \hat{n} ab \frac{\sin(k \sin\theta \cos\phi)}{ak \sin\theta \cos\phi} \frac{\sin(k \sin\theta \sin\phi)}{bk \sin\theta \sin\phi}. \qquad \text{Eq. (14.102)}$$

In the far field, the $\hat{\theta}$ polarized scattered field is given by

$$\vec{E}_s = -j\omega(\hat{\theta} \cdot \vec{A}) . \qquad \textbf{Eq. (14.103)}$$

Figure 14.27 shows the backscattered RCS for a rectangular plate versus incident angle, using the technique presented in this section. This plot can be reproduced using the MATLAB program *"rectplate.m,"* listed in Appendix 14-A.

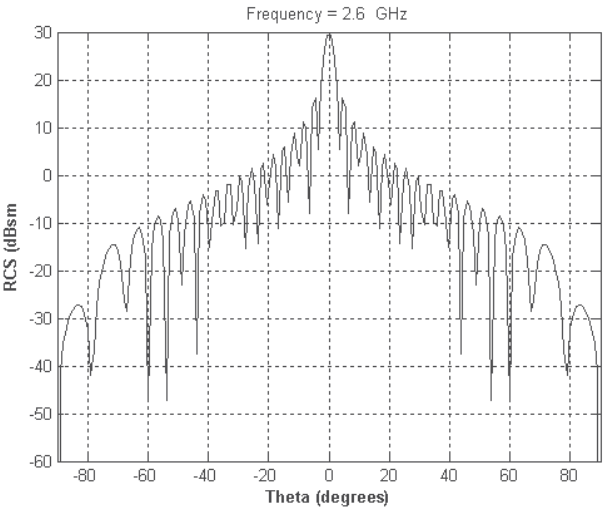


Figure 14.27. Rectangular plate backscattered RCS.

N-Sided Polygon

The integral in Eq. (14.100) has the form of a Fourier transform over the planar extent of the rectangular plate. In general, this 2-D Fourier transform is given by

$$S(u, v) = \iint_{x\,y} e^{j(ux+vy)} dx dy . \qquad \textbf{Eq. (14.104)}$$

The expression for an arbitrarily N-sided polygon in a local coordinate system has been evaluated analytically as

$$S(u, v) = \sum_{n=1}^{N} e^{j(\vec{\omega} \cdot \vec{\gamma})} \left[\frac{\hat{n} \times \hat{\alpha}_n \cdot \hat{\alpha}_{n-1}}{(\vec{\omega} \cdot \hat{\alpha}_n)(\vec{\omega} \cdot \hat{\alpha}_{n-1})} \right] \qquad \textbf{Eq. (14.105)}$$

where $\vec{\gamma}_n$ are the polygon vertices and α_n are the edge vectors given by

$$\hat{\alpha}_n = \frac{\vec{\gamma}_{n+1} - \vec{\gamma}_n}{\left| \vec{\gamma}_{n+1} - \vec{\gamma}_n \right|} , \qquad \textbf{Eq. (14.106)}$$

and $\vec{\omega} = u\hat{x} + v\hat{y}$. In the summation above, $\alpha_o = \alpha_N$. Figure 14. 28 shows a plot for the backscattered RCS for a N-sided polygon versus angle of incidence. This figure can be reproduced using the MATLAB program *"polygon.m,"* listed in Appendix 14-A.

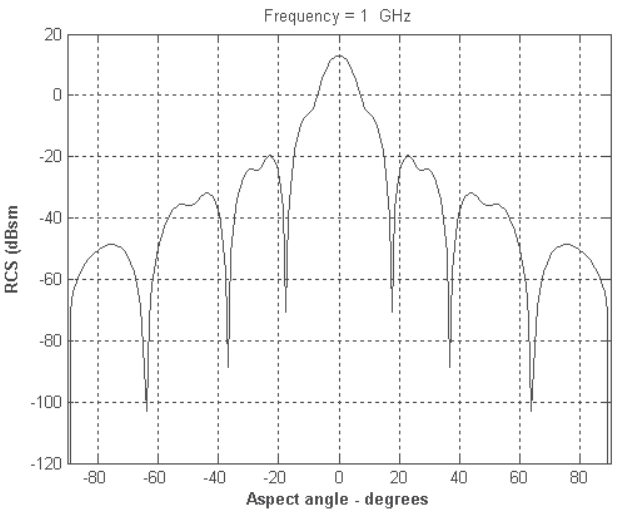


Figure 14.28. Backscattered RCS for an N-sided polygon.

14.6.7. Edge Diffraction

The PO does not treat the diffraction of waves at edges. In the late 1950s the Russian physicist, P. Ufimtsev, published a paper on a technique now known as the Physical Theory of Diffraction. In this paper, Ufimtsev introduced expressions for the edge diffraction at arbitrary incidence and scattering angles that complemented Physical Optics. This method was extended by K. Mitzer at Northrop who applied the PTD to incremental length edges in three dimensions. The PTD method was used in codes such as Northrop's (MISCAT), and was instrumental in the design of low cross-section aircraft such as Lockheed's F-117 Stealth fighter.

14.7. Multiple Bounce

Multiple reflections are a very important scattering mechanism in some complex targets. Many real-world targets have cavities or other concave areas where energy may be reflected and scattered several times. Examples are rocket boosters with nozzles and fuel tanks and aircraft with deep engine inlets. This type of scattering often results in high RCS at certain aspect angles, and significantly delayed returns that may cause the target to appear much longer in the downrange direction than it actually is.

A popular method for modeling these interactions is to treat the incident plane wave as a bundle of "ray tubes" as in GO theory, incorporating material effects and ray tube spreading and divergence. At the exit aperture of the ray bundle, a PO-type integral is performed over the ray tube footprint. This technique is known as Shooting and Bouncing Rays (SBR), and was developed at the University of Illinois in the late 1980s. This technique, as well as the PO and

PTD methods, are used in the well-known software called XPATCH, which has been used in high-frequency signature prediction for many years.

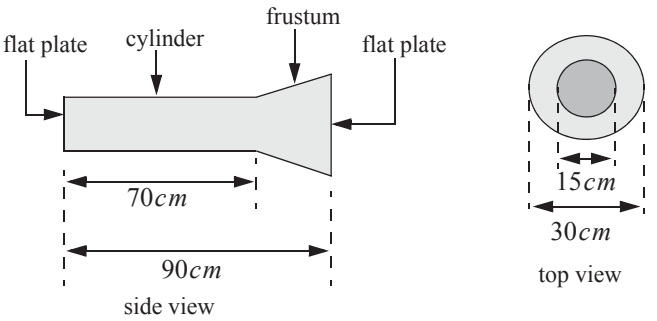
Problems

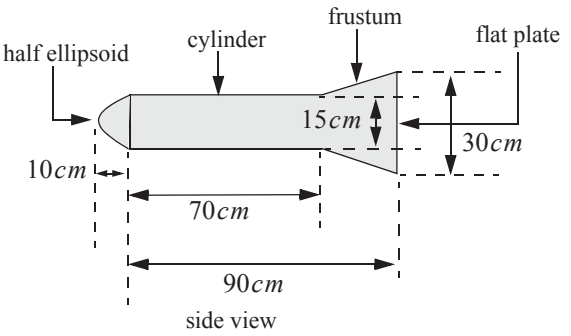
14.1. Design a cylindrical RCS calibration target such that its broadside RCS (cylinder) and end (flat plate) RCS are equal to $10m^2$ at $f = 9.5GHz$. The RCS for a flat plate of area A is $\sigma_{fp} = 4\pi f^2 A^2 / c^2$.

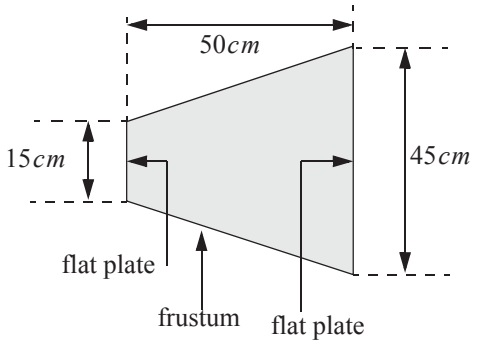
14.2. The following table is constructed from a radar cross-section measurement experiment. Calculate the mean and standard deviation of the radar cross section.

Number of samples	RCS, m^2
2	*55*
6	*67*
12	*73*
16	*90*
20	*98*
24	*110*
26	*117*
19	*126*
13	*133*
8	*139*
5	*144*
3	*150*

14.3. Develop a MATLAB simulation to compute and plot the backscattered RCS for the following objects. Utilize the simple shape MATLAB functions developed in this chapter. Assume that the radar is located on the left side of the page and that its line of sight is aligned with the target body axis. Assume an X-band radar.





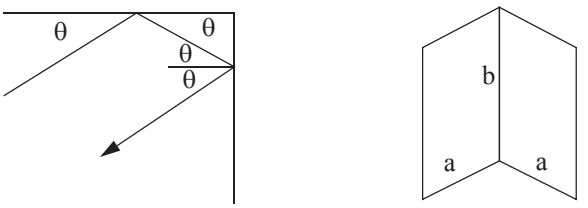


14.4. The backscattered RCS for a corner reflector is given by

$$\sigma = \left[\sqrt{\frac{16\pi a^4}{\lambda^2}(\sin\theta)^2} + \sqrt{\frac{4\pi a^4}{\lambda^2}\left(\frac{\sin\left(\frac{2\pi a}{\lambda}\sin\theta\right)}{\frac{2\pi a}{\lambda}\sin\theta} \right)^2} \right]^2 \qquad 0° \le \theta \le 45°.$$

This RCS is symmetric about the angle $\theta = 45°$. Develop a MATLAB program to compute and plot the RCS for a corner reflector. The RCS at the $\theta = 45°$ is

$$\sigma = \frac{8\pi a^2 b^2}{\lambda^2}.$$



corner reflector

Appendix 14-A: Chapter 14 MATLAB Code Listings

The MATLAB code provided in this chapter was designed as an academic standalone tool and is not adequate for other purposes. The code was written in a way to assist the reader in gaining a better understanding of the theory. The code was not developed, nor is it intended to be used as part of an open-loop or a closed-loop simulation of any kind. The MATLAB code found in this textbook can be downloaded from this book's web page on the CRC Press website. Simply use your favorite web browser, go to www.crcpress.com, and search for keyword *"Mahafza"* to locate this book's web page.

MATLAB Function "rcs_aspect.m" Listing

```
function [rcs] = rcs_aspect(scat_spacing, freq)
% This function demonstrates the effect of aspect angle on RCS
% The default frequency is 3GHz. The radar is observing two unity
% point scatterers separated by 1.0 meters. Initially the two scatterers
% are aligned with radar line of sight. The aspect angle is changed from
% 0 degrees to 180 degress and the equivalant RCS is computed.
% The RCS as measured by the radar versus aspect angle is then plotted.
% Inputs
   % scat_spacing in meters
   % freq radar frequency in Hz
%
% Output
   % rcs in dBsm
% Users may vary frequency, and/or scatteres spacing to observe RCS variation
eps = 0.0001;
wavelength = 3.0e+8 / freq;
% Compute aspect angle vector
aspect_degrees = 0.:.05:180.;
aspect_radians = (pi/180) .* aspect_degrees;
% Compute electrical scatterer spacing vector in wavelength units
elec_spacing = (2.0 * scat_spacing / wavelength) .* cos(aspect_radians);
% Compute RCS (rcs = RCS_scat1 + RCS_scat2)
% Scat1 is taken as phase refernce point
rcs = abs(1.0 + cos((2.0 * pi) .* elec_spacing) ...
          + i * sin((2.0 * pi) .* elec_spacing));
rcs = rcs + eps;
rcs = 20.0*log10(rcs); % RCS in dBsm
end
```

MATLAB Program "Fig.14_3.m" Listing

```
% generates Fig.. 14.3 of text
clc
close all
clear all
% Enter scatterer spacing, in meters
distance = input('Enter scatterer spacing, in meters \n');
% Enter frequency
freq = input('Enter Enter frequency in Hz \n');
rcs = rcs_aspect(distance,freq);
Figure (1);
aspect_degrees = 0.:.05:180.;
```

```
plot(aspect_degrees,rcs);
grid;
xlabel('\bfaspect angle - degrees');
ylabel('\bfRCS in dBsm');
```

MATLAB Function "rcs_frequency.m" Listing

```
function [rcs] = rcs_frequency (scat_spacing, frequ, freql)
% This program demonstrates the dependency of RCS on wavelength
% The default assumes two unity point scatterers separated
% The radar line of sight is aligned with the two scatterers
% Inputs
   % scat_spacing in meters
   % freql lower frequency limit in Hz
   % frequ upper frequency limit in Hz
% Output
   % rcs in dBsm
eps = 0.0001;
freq_band = frequ - freql;
delfreq = freq_band / 500.;
index = 0;
for freq = freql: delfreq: frequ
  index = index +1;
  wavelength(index) = 3.0e+8 / freq;
end
% Compute electrical scatterer spacing vector in wavelength units
elec_spacing = 2.0 * scat_spacing ./ wavelength;
% Compute RCS (RCS = RCS_scat1 + RCS_scat2)
rcs = abs (  1 + cos((2.0 * pi) .* elec_spacing)+ i * sin((2.0 * pi) .* elec_spacing));
rcs = rcs + eps;
rcs = 20.0*log10(rcs); % RCS ins dBsm
end
```

MATLAB Program "Fig.14_5_6.m" Listing

```
% Generates plot like Fig.. 14.5 and Fig. 14.6
% Enter scatterer spacing, in meters
clc
close all
clear all
scat_spacing = input('Enter scatterer spacing, in meters \n');
% Enter frequency band
freql = input('Enter lower frequency limit in Hz \n');
frequ = input('Enter upper frequency limit in Hz \n');
[rcs] = rcs_frequency (scat_spacing, frequ, freql);
N = size(rcs,2) ;
freq = linspace(freql,frequ,N)./1e9;
Figure (1);
plot(freq,rcs);
grid on;
xlabel('\bfFrequency');
ylabel('\bfRCS in dBsm');
```

MATLAB Program "Fig.14_10.m" Listing

```
% This program calculates the back-scattered RCS for a perfectly
% conducting sphere using Eq.(14.28), and produce plots similar to Fig.14.8
% Spherical Bessel functions are computed using series approximation and recursion.
clc
close all
clear all
eps   = 0.00001;
index = 0;
% kr limits are [0.05 - 15] ===> 300 points
for kr = 0.05:0.01:25
   index = index + 1;
   sphere_rcs   = 0. + 0.*i;
   f1    = 0. + 1.*i;
   f2    = 1. + 0.*i;
   m     = 1.;
   n     = 0.;
   q     = -1.;
   % initially set del to huge value
   del =100000+100000*i;
   while(abs(del) > eps)
      q   = -q;
      n   = n + 1;
      m   = m + 2;
      del = (2.*n-1) * f2 / kr-f1;
      f1  = f2;
      f2  = del;
      del = q * m /(f2 * (kr * f1 - n * f2));
      sphere_rcs = sphere_rcs + del;
   end
   rcs(index)   = abs(sphere_rcs);
   sphere_rcsdb(index) = 10. * log10(rcs(index));
   end
Figure(1);
n=0.05:.01:25;
subplot(2,1,1)
plot (n,rcs,'k','linewidth',1.5);
% set (gca,'xtick',[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]);
xlabel ('\bfSphere circumference in wavelengths; (2 \pi r / \lambda)');
ylabel ('\bf Normalized RCS ( \sigma / \pi r^2)');
grid on
subplot(2,1,2)
plot (n,sphere_rcsdb,'k','linewidth',1.5);
%set (gca,'xtick',[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]);
xlabel ('\bfSphere circumference in wavelengths; (2 \pi r / \lambda)');
ylabel ('\bf Normalized RCS ( \sigma / \pi r^2) - dB');
grid;
Figure (2);
semilogx (n,sphere_rcsdb,'k','linewidth',1.5);
xlabel ('\bfSphere circumference in wavelengths; (2 \pi r / \lambda)');
ylabel ('\bf Normalized sphere RCS ( \sigma / \pi r^2) - dB');
grid on
gtext('\bfRayleigh Region')
gtext('\bfMie Region')
```

```
gtext('\bfOptical Region')
```

MATLAB Function "rcs_ellipsoid.m" Listing

```
function [rcs_db] = rcs_ellipsoid (a, b, c, phi)
% This program computes the back-scattered RCS for an ellipsoid.
% The angle phi is fixed, while the angle theta is varied from 0-180 deg.
% Inputs
   % a          == ellipsoid a-radius in meters
   % b          == ellipsoid b-radius in meters
   % c          == ellipsoid c-radius in meters
   % phi        == ellipsoid roll angle in degrees
%Output
   % rcs        == ellipsoid rcs versus aspect angle in dBsm
eps = 0.00001;
sin_phi_s = sin(phi)^2;
cos_phi_s = cos(phi)^2;
% Generate aspect angle vector
theta = 0.:.05:180;
theta = (theta .* pi) ./ 180.;
if(a ~= b & a ~= c)
  rcs = (pi * a^2 * b^2 * c^2) ./ (a^2 * cos_phi_s .* (sin(theta).^2) + ...
  b^2 * sin_phi_s .* (sin(theta).^2) + ...
  c^2 .* (cos(theta).^2)).^2 ;
else
  if(a == b & a ~= c)
    rcs = (pi * b^4 * c^2) ./ ( b^2 .* (sin(theta).^2) + ...
      c^2 .* (cos(theta).^2)).^2 ;
  else
    if (a == b & a ==c)
      rcs = pi * c^2;
    end
  end
end
rcs_db = 10.0 * log10(rcs);
return
```

MATLAB Program "Fig14_12a.m" Listing

```
% generates Fig 14.12a of text
clc
close all
clear all
% Enter the ellpsiod a radius
a = .15;
% Enter the ellpsiod b radius
b = .3;
% Enter the ellpsiod c radius
c = .95;
% Enter the ellpsiod roll angle in degrees
phi = [0 75 90];
[rcs_db1] = rcs_ellipsoid (a, b, c, phi(1));
[rcs_db2] = rcs_ellipsoid (a, b, c, phi(2));
[rcs_db3] = rcs_ellipsoid (a, b, c, phi(3));
```

```
N = size(rcs_db1,2);
theta = linspace(0.0, pi, N);
theta = theta .* 180 ./ pi;
figure (1);
plot(theta,rcs_db1,'k:',theta,rcs_db2,'k',theta,rcs_db3,'k-.','linewidth',1.5);
xlabel ('\bfAspect angle - degrees');
ylabel ('\bfEllipsoid RCS - dBsm');
legend ('\phi = 0.0', '\phi = 75', '\phi = 90')
title('(a, b, c) = (.15, .3, .95) meters')
grid on;
```

MATLAB Function "rcs_circ_plate.m" Listing

```
function [rcsdb] = rcs_circ_plate (r, freq)
% This program calculates and plots the backscattered RCS of
% circular flat plate of radius r.
eps = 0.000001;
% Compute aspect angle vector
% Compute wavelength
lambda = 3.e+8 / freq; % X-Band
index = 0;
for aspect_deg = 0.:.1:180
  index = index +1;
  aspect = (pi /180.) * aspect_deg;
% Compute RCS using Eq. (2.37)
  if (aspect == 0 | aspect == pi)
     rcs_po(index) = (4.0 * pi^3 * r^4 / lambda^2) + eps;
     rcs_mu(index) = rcs_po(1);
   else
     x = (4. * pi * r / lambda) * sin(aspect);
     val1 = 4. * pi^3 * r^4 / lambda^2;
     val2 = 2. * besselj(1,x) / x;
     rcs_po(index) = val1 * (val2 * cos(aspect))^2 + eps;
% Compute RCS using Eq. (2.36)
     val1m = lambda * r;
     val2m = 8. * pi * sin(aspect) * (tan(aspect)^2);
     rcs_mu(index) = val1m / val2m + eps;
   end
 end
 % Compute RCS using Eq. (2.35) (theta=0,180)
rcsdb = 10. * log10(rcs_po);
rcsdb_mu = 10 * log10(rcs_mu);
angle = 0:.1:180;
plot(angle,rcsdb,'k',angle,rcsdb_mu,'k-.')
grid;
xlabel ('\bfAspect angle - degrees');
ylabel ('\bfRCS - dBsm');
axis tight
legend('Using Eq.(14.39)','Using Eq.(14.38)')
freqGH = num2str(freq*1.e-9);
title (['Frequency = ',[freqGH],'  GHz']);
end
```

MATLAB Function "rcs_frustum.m" Listing

```
function [rcs] = rcs_frustum (r1, r2, h, freq, indicator)
% This program computes the monostatic RCS for a frustum.
% Incident linear Polarization is assumed.
% When viewing from the small end of the frustum
% normal incedence occurs at aspect pi/2 - half cone angle
% When viewing from the large end, normal incidence occur at
% pi/2 + half cone angle.
% RCS is computed using Eq. (14.43). This program assumes a geometry
% similar top Fig. 14.13
% Inputs
   % r1       == small end radius in meters
   % r2       == large end radius in meters
   % freq     == frequency in Hz
   % indicator == 1 when viewing from large end 0 when viewing from small end
% Output
   % rcs      == array of RCS versus aspect angle
format long
index = 0;
eps = 0.000001;
lambda = 3.0e+8 /freq;
% Enter frustum's small end radius
%r1 =.02057;
% Enter Frustum's large end radius
%r2 = .05753;
% Compute Frustum's length
%h = .20945;
% Comput half cone angle, alpha
alpha = atan(( r2 - r1)/h);
% Compute z1 and z2
z2 = r2 / tan(alpha);
z1 = r1 / tan(alpha);
delta = (z2^1.5 - z1^1.5)^2;
factor = (8. * pi * delta) / (9. * lambda);
%('enter 1 to view frustum from large end, 0 otherwise')
large_small_end = indicator;
if(large_small_end == 1)
  % Compute normal incidence, large end
  normal_incedence = (180./pi) * ((pi /2) + alpha)
  % Compute RCS from zero aspect to normal incidence
  for theta = 0.001:.1:normal_incedence-.5
    index = index +1;
    theta = theta * pi /180.;
    rcs(index) = (lambda * z1 * tan(alpha) *(tan(theta - alpha))^2) / ...
      (8. * pi *sin(theta)) + eps;
  end
  %Compute broadside RCS
  index = index +1;
  rcs_normal = factor * sin(alpha) / ((cos(alpha))^4) + eps;
  rcs(index) = rcs_normal;
  % Compute RCS from broad side to 180 degrees
  for theta = normal_incedence+.5:.1:180
    index = index + 1;
    theta =  theta * pi / 180. ;
```

```
    rcs(index) = (lambda * z2 * tan(alpha) *(tan(theta - alpha))^2) / ...
      (8. * pi *sin(theta)) + eps;
  end
else
  % Compute normal incidence, small end
  normal_incidence = (180./pi) * ((pi /2) - alpha)
  % Compute RCS from zero aspect to normal incidence (large end of frustum)
  for theta = 0.001:.1:normal_incedence-.5
    index = index +1;
    theta = theta * pi /180.;
    rcs(index) = (lambda * z1 * tan(alpha) *(tan(theta + alpha))^2) / ...
      (8. * pi *sin(theta)) + eps;
  end
  %Compute broadside RCS
  index = index +1;
  rcs_normal = factor * sin(alpha) / ((cos(alpha))^4) + eps;
  rcs(index) = rcs_normal;
  % Compute RCS from broad side to 180 degrees (small end of frustum)
  for theta = normal_incedence+.5:.1:180
    index = index + 1;
    theta =  theta * pi / 180. ;
    rcs(index) = (lambda * z2 * tan(alpha) *(tan(theta + alpha))^2) / ...
      (8. * pi *sin(theta)) + eps;
  end
end
% Plot RCS versus aspect angle
delta = 180 /index;
angle = 0.001:delta:180;
plot (angle,10*log10(rcs),'k','linewidth',1.5);
grid;
xlabel ('\bfApsect angle - degrees');
ylabel ('\bfRCS - dBsm');
axis tight
if(indicator ==1)
   title ('\bfViewing from large end');
else
   title ('\bfViewing from small end');
end
```

MATLAB Function "rcs_cylinder.m" Listing

```
function [rcs] = rcs_cylinder(r1, r2, h, freq, phi, CylinderType)
% rcs_cylinder.m
% This program compute monostatic RCS for a finite length
% cylinder of either circular or elliptical cross-section.
% Plot of RCS versus aspect angle theta is generated at a specified
r = r1;          % radius of the circular cylinder
eps =0.00001;
dtr = pi/180;
phir = phi*dtr;
lambda = 3.0e+8 /freq;      % wavelength
% CylinderType= 'Elliptic';   % 'Elliptic' or 'Circular'

switch CylinderType
```

```
case 'Circular'
   % Compute RCS from 0 to (90-.5)  degrees
   index = 0;
   for theta = 0.0:.1:90-.5
      index = index +1;
      thetar = theta * dtr;
      rcs(index) = (lambda * r * sin(thetar) / ...
         (8. * pi * (cos(thetar))^2)) + eps;
   end
   % Compute RCS for broadside specular at 90 degree
   thetar = pi/2;
   index = index +1;
   rcs(index) = (2. * pi * h^2 * r / lambda )+ eps;
   % Compute RCS from (90+.5) to 180 degrees
   for theta = 90+.5:.1:180.
      index = index + 1;
      thetar = theta * dtr;
      rcs(index) = ( lambda * r * sin(thetar) / ...
         (8. * pi * (cos(thetar))^2)) + eps;
   end
case 'Elliptic'
  r12 = r1*r1;
   r22 = r2*r2;
   h2 = h*h;
   % Compute RCS from 0 to (90-.5)  degrees
   index = 0;
   for theta = 0.0:.1:90-.5
      index = index +1;
      thetar = theta * dtr;
      rcs(index) =  lambda * r12 * r22 * sin(thetar) / ...
            ( 8*pi* (cos(thetar)^2)* ( (r12*cos(phir)^2 + r22*sin(phir)^2)^1.5 ))+ eps;
   end
   % Compute RCS for broadside specular at 90 degree
   index = index +1;
   rcs(index) = 2. * pi * h2 * r12 * r22 / ...
            ( lambda*( (r12*cos(phir)^2 + r22*sin(phir)^2)^1.5 ))+ eps;
   % Compute RCS from (90+.5) to 180 degrees
   for theta = 90+.5:.1:180.
      index = index + 1;
      thetar = theta * dtr;
      rcs(index) =  lambda * r12 * r22 * sin(thetar) / ...
            ( 8*pi* cos(thetar)^2* ( (r12*cos(phir)^2 + r22*sin(phir)^2)^1.5 ))+ eps;
   end
end
end
```

MATLAB Program "Fig14_19.m" Listing

```
% generates Fig. 14.19 of text
clc
close all
clear all
r1 = .125;
r2 = 0.05;
```

```
h = 1;
phi = 45;
freq = 5.6e9;
freqGH = num2str(freq*1.e-9);
% Fig 14.19a
[rcs1] = rcs_cylinder(r1, r1, h, freq, phi,'Circular');
figure(1)
angle = linspace(0,180,size(rcs1,2));
plot(angle,10*log10(rcs1),'k','linewidth',1.5);
grid on;
xlabel ('\bfAspect angle in dDegrees');
ylabel ('\bfRCS - dBsm');
title  (['Circular Cylinder at Frequency = ',[freqGH],'  GHz']);
% Fig. 14.19b
[rcs2] = rcs_cylinder(r1, r2, h, freq, phi,'Elliptic');
figure(2)
angle = linspace(0,180,size(rcs2,2));
plot(angle,10*log10(rcs2),'k','linewidth',1.5);
grid on;
xlabel ('\bfAspect angle in degrees');;
ylabel ('\bfRCS - dBsm');
title  (['Elliptic Cylinder at Frequency = ',[freqGH],'  GHz']);
```

MATLAB Function "rcs_rect_plate.m" Listing

```
function  [rcsdb_h,rcsdb_v] = rcs_rect_plate(a, b, freq)
% This program computes the backscattered RCS for a rectangular
% flat plate. The RCS is computed for vertical and horizontal
% polarization based on Eq.s(14.52)through (14.62). Also Physical
% Optics approximation Eq.(14.64) is computed.
% User may vary frequency, or the plate's dimensions.
% Default values are a=b=10.16cm; lambda=3.25cm.
eps = 0.000001;
% Enter a, b, and lambda
lambda = .0325;
ka = 2. * pi * a / lambda;
% Compute aspect angle vector
theta_deg = 0.05:0.1:85;
theta = (pi/180.) .* theta_deg;
sigma1v = cos(ka .*sin(theta)) - i .* sin(ka .*sin(theta)) ./ sin(theta);
sigma2v = exp(i * ka - (pi /4)) / (sqrt(2 * pi) *(ka)^1.5);
sigma3v = (1. + sin(theta)) .* exp(-i * ka .* sin(theta)) ./ ...
   (1. - sin(theta)).^2;
sigma4v = (1. - sin(theta)) .* exp(i * ka .* sin(theta)) ./ ...
   (1. + sin(theta)).^2;
sigma5v = 1. - (exp(i * 2. * ka - (pi / 2)) / (8. * pi * (ka)^3));
sigma1h = cos(ka .*sin(theta)) + i .* sin(ka .*sin(theta)) ./ sin(theta);
sigma2h = 4. * exp(i * ka * (pi / 4.)) / (sqrt(2 * pi * ka));
sigma3h =  exp(-i * ka .* sin(theta)) ./ (1. - sin(theta));
sigma4h = exp(i * ka * sin(theta)) ./ (1. + sin(theta));
sigma5h = 1. - (exp(j * 2. * ka + (pi / 4.)) / 2. * pi * ka);
% Compute vertical polarization RCS
rcs_v = (b^2 / pi) .* (abs(sigma1v - sigma2v .*((1. ./ cos(theta)) ...
   + .25 .* sigma2v .* (sigma3v + sigma4v)) .* (sigma5v).^-1)).^2 + eps;
```

```
% compute horizontal polarization RCS
rcs_h = (b^2 / pi) .* (abs(sigma1h - sigma2h .*((1. ./ cos(theta)) ...
  - .25 .* sigma2h .* (sigma3h + sigma4h)) .* (sigma5h).^-1)).^2 + eps;
% Compute RCS from Physical Optics, Eq.(2.62)
angle = ka .* sin(theta);
rcs_po = (4. * pi* a^2 * b^2 / lambda^2 ).*  (cos(theta)).^2 .* ...
  ((sin(angle) ./ angle).^2) + eps;
rcsdb_v = 10. .*log10(rcs_v);
rcsdb_h = 10. .*log10(rcs_h);
rcsdb_po = 10. .*log10(rcs_po);
figure
plot (theta_deg, rcsdb_v,'k',theta_deg,rcsdb_po,'k -.','linewidth',1.5);
set(gca,'xtick',[10:10:85]);
freqGH = num2str(freq*1.e-9);
A = num2str(a);
B = num2str(b);
title (['Vertical Polarization,  ','Frequency = ',[freqGH],'  GHz, ', '  a = ', [A], ' m','  b = ',[B],' m']);
ylabel ('\bfRectangular plate RCS -dBsm');
xlabel ('\bfAspect angle - deg');
legend('Eq.(14.52)','Eq.(14.53)')
grid on
figure
plot (theta_deg, rcsdb_h,'k',theta_deg,rcsdb_po,'k -.','linewidth',1.5);
set(gca,'xtick',[10:10:85]);
title (['Horizontal Polarization,  ','Frequency = ',[freqGH],'  GHz, ', '  a = ', [A], ' m','  b = ',[B],' m']);
ylabel ('\bfREctangular plate RCS -dBsm');
xlabel ('\bfAspect angle - deg');
legend('Eq.(14.53)','Eq.(14.53)')
grid on
```

MATLAB Function "rcs_isosceles.m" Listing

```
function [rcs] = rcs_isosceles (a, b, freq, phi)
% This program calculates the backscattered RCS for a perfectly
% conducting triangular flat plate, using Eq.s (14.65) through (14.67)
% The default case is to assume phi = pi/2. These equations are
% valid for aspect angles less than 30 degrees
% Users may vary wavelength, or plate's dimensions
% Inputs
   % a     == height of plate in meters
   % b     == base of plate in meters
   % freq  == frequency in Hz
   % phi   == roll angle in degrees
% Output
   % rcs   == array of RCS versus aspect angle
A = a * b / 2.;
lambda = 3.e+8 / freq;
ka = 2. * pi / lambda;
kb = 2. *pi / lambda;
% Compute theta vector
theta_deg = 0.01:.05:89;
theta = (pi /180.) .* theta_deg;
alpha = ka * cos(phi) .* sin(theta);
beta =  kb * sin(phi) .* sin(theta);
```

```
if (phi == pi / 2)
  rcs = (4. * pi * A^2 / lambda^2) .* cos(theta).^2 .* (sin(beta ./ 2)).^4 ...
     ./ (beta./2).^4 + eps;
end
if (phi == 0)
   rcs = (4. * pi * A^2 / lambda^2) .* cos(theta).^2 .* ...
      ((sin(alpha).^4 ./ alpha.^4) + (sin(2 .* alpha) - 2.*alpha).^2 ...
      ./ (4 .* alpha.^4)) + eps;
end
if (phi ~= 0 & phi ~= pi/2)
   sigmao1 = 0.25 *sin(phi)^2 .* ((2. * a / b) * cos(phi) .* ...
      sin(beta) - sin(phi) .* sin(2. .* alpha)).^2;
   fact1 = (alpha).^2 - (.5 .* beta).^2;
   fact2 = (sin(alpha).^2 - sin(.5 .* beta).^2).^2;
   sigmao = (fact2 + sigmao1) ./ fact1;
   rcs = (4. * pi * A^2 / lambda^2) .* cos(theta).^2 .* sigmao + eps;
end
rcsdb = 10. *log10(rcs);
plot(theta_deg,rcsdb,'k','linewidth', 1.5)
xlabel ('\bfAspect angle - degrees');
ylabel ('\bfRCS - dBsm')
grid on
```

MATLAB Program "rcs_cylinder_cmplx.m" Listing

```
clc
close
clear all
indes = 0;
eps =0.00001;
a1 =.125;
h = 1.;
lambda = 3.0e+8 /9.5e+9;
lambda = 0.00861;
index = 0;
for theta = 0.0:.1:90-.1
   index = index +1;
   theta = theta * pi /180.;
   rcs(index) = (lambda * a1 * sin(theta) / (8 * pi * (cos(theta))^2)) + eps;
end
theta*180/pi
theta = pi/2;
index = index +1
rcs(index) = (2 * pi * h^2 * a1 / lambda )+ eps;
for theta = 90+.1:.1:180.
   index = index + 1;
   theta = theta * pi / 180.;
   rcs(index) = ( lambda * a1 * sin(theta) / (8 * pi * (cos(theta))^2)) + eps;
end
%%%%%%%%%%
r = a1;
index = 0;
for aspect_deg = 0.:.1:180
   index = index +1;
```

```
  aspect = (pi /180.) * aspect_deg;
% Compute RCS using Eq. (2.37)
  if (aspect == 0 | aspect == pi)
     rcs_po(index) = (4.0 * pi^3 * r^4 / lambda^2) + eps;
     rcs_mu(index) = rcs_po(1);
   else
     x = (4. * pi * r / lambda) * sin(aspect);
     val1 = 4. * pi^3 * r^4 / lambda^2;
     val2 = 2. * besselj(1,x) / x;
     rcs_po(index) = val1 * (val2 * cos(aspect))^2 + eps;
   end
 end
rcs_t =(rcs_po + rcs);
 %%%%%%%%%%%%%%
angle = 0:.1:180;
plot(angle,10*log10(rcs_t(1:1801)),'k');
xlabel('\bfAspect angle in degrees')
ylabel('\bfRCS - dBsm')
grid
```

MATLAB Program "fdtd.m" Listing

```
clear all
%
mu_o = pi*4.0e-7;              % free space permeability
epsilon_o = 8.854e-12;         % free space permittivity
%
c = 1.0/sqrt(mu_o * epsilon_o);   % speed of light
%
length_x = 2.0;                % x-width of region
nx = 200;                      % number of x grid points
dx = length_x / (nx - 1);      % x grid size
%
x = linspace(0.0, length_x, nx);  % x array
%
length_y = 2.0;                % y-width of region
ny = 200;                      % number of y grid points
dy = length_y / (ny - 1);      % y grid size
%
y = linspace(0.0, length_y, ny);  % y array
%
max_timestep = c*sqrt(1.0/(dx*dx) + 1.0/(dy*dy));   % max tstep for FDTD
max_timestep = 1.0/max_timestep;
%
delta_t = 0.5*max_timestep;    % delta t a little less than max tstep
%
er = 8.0;                      % relative permittivity of slab
%
epsilon = epsilon_o*ones(ny, nx); % epsilon array
mu = mu_o*ones(ny - 1, nx - 1);   % mu array
%
a1 = [0.5 1.5 1.5 0.5 0.5];    % for drawing slab on plot
a2 = [0.6 0.6 0.8 0.8 0.6];    % for drawing slab on plot
%
```

```
x1 = fix(0.5/dx)+1;              % grid extents for slab
y1 = fix(0.6/dy);                % grid extents for slab
x2 = fix(1.5/dx)+1;              % grid extents for slab
y2 = fix(0.8/dy);                % grid extents for slab
%
epsilon(y1:y2,x1:x2) = er*epsilon_o;  % set epsilon inside slab
%
j_x = nx/2;                      % x location of current source
j_y = ny/2;                      % y location of current source
%
e_z_1 = zeros(ny, nx);           % initialize array. e_z at boundaries will remain 0
h_x_1 = zeros(ny - 1, nx - 1);  % initialize array
h_y_1 = zeros(ny - 1, nx - 1);  % initialize array
e_z_2 = zeros(ny, nx);           % initialize array. e_z at boundaries will remain 0
h_x_2 = zeros(ny - 1, nx - 1);  % initialize array
h_y_2 = zeros(ny - 1, nx - 1);  % initialize array
%
ntim = 300;                      % number of desired time points
f_o = 600e6;                     % base frequency for pulse
tau = 1.0/(4.0*pi*f_o);          % tau for pulse
%
for i_t = 1:ntim
%
  time(i_t) = i_t * delta_t;
%
  i_t
  time(i_t)
%
  if time(i_t) > 3.36e-9
    break
  end
%
  jz(i_t) = (4.0 * (time(i_t)/tau)^3 - (time(i_t)/tau)^4) * exp(-time(i_t)/tau);
%
  for i_x = 2:nx-1     % ez at boundaries remains zero
    for i_y = 2:ny-1  % ez at boundaries remains zero
%
      j = 0.0;
      if i_x == j_x
        if i_y == j_y
          j = jz(i_t);
        end
      end
%
      if rem(i_t, 2) == 1
        a = 1.0/dx*(h_y_1(i_y, i_x) - h_y_1(i_y, i_x - 1));
        b = 1.0/dy*(h_x_1(i_y, i_x) - h_x_1(i_y - 1, i_x));
        e_z_2(i_y, i_x) = e_z_1(i_y, i_x) + (delta_t/epsilon(i_y, i_x))*(a - b) - j;
      else
        a = 1.0/dx*(h_y_2(i_y, i_x) - h_y_2(i_y, i_x - 1));
        b = 1.0/dy*(h_x_2(i_y, i_x) - h_x_2(i_y - 1, i_x));
        e_z_1(i_y, i_x) = e_z_2(i_y, i_x) + (delta_t/epsilon(i_y, i_x))*(a - b) - j;
      end
%
```

```
    end
  end
%
  for i_x = 1:nx-1
    for i_y = 1:ny-1
%
      if rem(i_t, 2) == 1
        h_x_2(i_y, i_x) = h_x_1(i_y, i_x) - (delta_t/mu(i_y, i_x)/dy)*(e_z_2(i_y + 1, i_x) - e_z_2(i_y, i_x));
          h_y_2(i_y, i_x) = h_y_1(i_y, i_x) + (delta_t/mu(i_y, i_x)/dx)*(e_z_2(i_y, i_x + 1) - e_z_2(i_y,
i_x));
      else
        h_x_1(i_y, i_x) = h_x_2(i_y, i_x) - (delta_t/mu(i_y, i_x)/dy)*(e_z_1(i_y + 1, i_x) - e_z_1(i_y, i_x));
          h_y_1(i_y, i_x) = h_y_2(i_y, i_x) + (delta_t/mu(i_y, i_x)/dx)*(e_z_1(i_y, i_x + 1) - e_z_1(i_y,
i_x));
      end
    end
  end
%
  pcolor(x, y, abs(e_z_2))
  line(a1, a2, 'Linewidth', 1.0, 'Color', 'white');
  xlabel('X (m)')
  ylabel('Y (m)')
  title('Ez (V/m)')
  axis square
  shading interp
  %colormap gray
  caxis([0 .1])
  %axis([0 2 0 2 0 .1])
  fr(i_t) = getframe;
end
```

MATLAB Program "rectplate.m" Listing

```
close all
clear all
frequency = 2.6e9;              % desired radar frequency
freqGH = num2str(frequency*1.e-9);
c = 299795645.0;                % speed of light
w = 2.0*pi*frequency;           % radian frequency
wavenumber = w/c;               % free space wavenumber
mu = 4.0*pi*1.0e-7;             % free space permeability
z_o = 376.7343;                 % free space wave impedance
l_x = 1.0;                      % length of plate
l_y = 1.0;                      % width of plate
normal_vect = [0 0 1];          % +z normal for x-y plane
theta_points = 180;             % number of points in theta
phi_points = 1;                 % number of points in phi
 theta = linspace(-0.5*pi, 0.5*pi, theta_points);
phi = linspace(0.0, 2.0*pi, phi_points);
 for i_theta = 1:theta_points
      for i_phi = 1:phi_points
          theta_vect(1) = cos(theta(i_theta))*cos(phi(i_phi));
     theta_vect(2) = cos(theta(i_theta))*sin(phi(i_phi));
     theta_vect(3) = -sin(theta(i_theta));
```

```
            phi_vect(1) = -sin(phi(i_phi));
        phi_vect(2) = cos(phi(i_phi));
        phi_vect(3) = 0.0;
            u = sin(theta(i_theta))*cos(phi(i_phi));
        v = sin(theta(i_theta))*sin(phi(i_phi));
            vect_term = dot(theta_vect, cross(phi_vect, normal_vect));
                        es(i_theta,  i_phi)  =  -j*w*mu/2.0/pi/z_o*vect_term*l_x*l_y*sinc(wavenum-
ber*u*l_x)*sinc(wavenumber*v*l_y);
    end
end
rcs = 20.0*log10(sqrt(4*pi)*abs(es));
plot(180*theta/pi, rcs)
axis([-90 90 -60 30])
xlabel('\bfTheta (degrees)')
ylabel('\bfRCS (dBsm')
grid on
title  (['Frequency = ',[freqGH],'  GHz']);
return
```

MATLAB Program "polygon.m" Listing

```
% this routine calculates the scattered electric field of an arbitrary
% N-sided polygon located in the x-y plane.
clc
clear all
close all
frequency = 1.0e9;              % desired radar frequency
freqGH = num2str(frequency*1.e-9);
c = 299795645.0;                % speed of light
w = 2.0*pi*frequency;           % radian frequency
wavenumber = w/c;               % free space wavenumber
mu = 4.0*pi*1.0e-7;             % free space permeability
z_o = 376.7343;                 % free space wave impedance

nsides = 3;                     % number of polygon sides
vertices(1,:) = [0.0 0.0 0.0];    % vertexes of polygon (counterclockwise)
vertices(2,:) = [1.0 0.5 0.0];    % vertexes of polygon (counterclockwise)
vertices(3,:) = [1.5 0.0 0.0];    % vertexes of polygon (counterclockwise)

for n = 1:nsides
    if n == nsides
        alpha_n(nsides,1) = vertices(1,1) - vertices(nsides,1);
        alpha_n(nsides,2) = vertices(1,2) - vertices(nsides,2);
        alpha_n(nsides,3) = vertices(1,3) - vertices(nsides,3);
    else
        alpha_n(n,1) = vertices(n+1,1) - vertices(n,1);
        alpha_n(n,2) = vertices(n+1,2) - vertices(n,2);
        alpha_n(n,3) = vertices(n+1,3) - vertices(n,3);
    end
    alpha_n(n, 1:3) = alpha_n(n, 1:3)/norm(alpha_n(n, 1:3));
end

normal_vect = [0 0 1];          % +z normal for x-y plane
```

```
theta_points = 180;              % number of points in theta
phi_points = 1;                  % number of points in phi

theta = linspace(-0.5*pi, 0.5*pi, theta_points);
phi = linspace(0.0, 2.0*pi, phi_points);

for i_theta = 1:theta_points

   for i_phi = 1:phi_points

      theta_vect(1) = cos(theta(i_theta))*cos(phi(i_phi));
      theta_vect(2) = cos(theta(i_theta))*sin(phi(i_phi));
      theta_vect(3) = -sin(theta(i_theta));

      phi_vect(1) = -sin(phi(i_phi));
      phi_vect(2) = cos(phi(i_phi));
      phi_vect(3) = 0.0;

      w_vect(1) = 2*wavenumber*sin(theta(i_theta))*cos(phi(i_phi));
      w_vect(2) = 2*wavenumber*sin(theta(i_theta))*sin(phi(i_phi));
      w_vect(3) = 0.0;

      s_term = 0.0;

      for n = 1:nsides
         expterm = exp(i*dot(w_vect, vertices(n,1:3)));
         if n == 1
            num = dot(cross(normal_vect, alpha_n(n,1:3)), alpha_n(nsides,1:3));
            denom = dot(w_vect, alpha_n(n,1:3))*dot(w_vect, alpha_n(nsides,1:3));
         else
            num = dot(cross(normal_vect, alpha_n(n,1:3)), alpha_n(n-1,1:3));
            denom = dot(w_vect, alpha_n(n,1:3))*dot(w_vect, alpha_n(n-1,1:3));
         end
         s_term = s_term + num*expterm/denom;
      end

      vect_term = dot(theta_vect, cross(phi_vect, normal_vect));

      es(i_theta, i_phi) = -j*w*mu/2.0/pi/z_o*vect_term*s_term;

   end
end

rcs = 20.0*log10(sqrt(4*pi)*abs(es));
plot(180*theta/pi, rcs)
axis([-90 90 -120 20])
xlabel('\bfAspect angle - degrees')
ylabel('\bfRCS (dBsm')
grid on
title  (['Frequency = ',[freqGH],'  GHz']);
```

Chapter 15

Phased Array Antennas

15.1. Directivity, Power Gain, and Effective Aperture

Radar antennas can be characterized by the directive gain G_D, power gain G, and effective aperture A_e. Antenna gain is a term used to describe the ability of an antenna to concentrate the transmitted energy in a certain direction. Directive gain, or simply directivity, is more representative of the antenna radiation pattern, while power gain is normally used in the radar equation. Plots of the power gain and directivity, when normalized to unity, are called the *antenna radiation pattern*. The directivity of a transmitting antenna can be defined by

$$G_D = \frac{maximum\ \ radiation\ \ intensity}{average\ \ radiation\ \ intensity}. \qquad \textbf{Eq. (15.1)}$$

The radiation intensity is the power-per-unit solid angle in the direction (θ, ϕ) and denoted by $P(\theta, \phi)$. The average radiation intensity over 4π radians (solid angle) is the total power divided by 4π. Hence, Eq. (15.1) can be written as

$$G_D = \frac{4\pi(maximum\ \ radiated\ \ power/unit\ \ solid\ \ angle)}{total\ \ radiated\ \ power}. \qquad \textbf{Eq. (15.2)}$$

It follows that

$$G_D = 4\pi \frac{P(\theta, \phi)_{max}}{\int_0^{\pi}\int_0^{2\pi} P(\theta, \phi) d\theta d\phi}. \qquad \textbf{Eq. (15.3)}$$

As an approximation, it is customary to rewrite Eq. (15.3) as

$$G_D \approx \frac{4\pi}{\theta_3 \phi_3} \qquad \textbf{Eq. (15.4)}$$

where θ_3 and ϕ_3 are the antenna half-power (3-dB) beamwidths in either direction. The antenna power gain and its directivity are related by

$$G = \rho_r G_D \qquad \textbf{Eq. (15.5)}$$

where ρ_r is the radiation efficiency factor. In this book, the antenna power gain will be denoted as *gain*. The radiation efficiency factor accounts for the ohmic losses associated with

the antenna. Therefore, the definition for the antenna gain is also given in Eq. (15.1). The antenna effective aperture A_e is related to gain by

$$A_e = \frac{G\lambda^2}{4\pi} \quad \text{Eq. (15.6)}$$

where λ is the wavelength. The relationship between the antenna's effective aperture A_e and the physical aperture A is

$$A_e = \rho A \quad \text{Eq. (15.7)}$$
$$0 \le \rho \le 1.$$

ρ is referred to as the aperture efficiency, and good antennas require $\rho \to 1$ (in this book $\rho = 1$ is always assumed, i.e., $A_e = A$).

Using simple algebraic manipulations of Eqs. (15.4) through (15.6) (assuming that $\rho_r = 1$) yields

$$G = \frac{4\pi A_e}{\lambda^2} \approx \frac{4\pi}{\theta_3 \phi_3} \quad \text{Eq. (15.8)}$$

Consequently, the angular cross section of the beam is

$$\theta_3 \phi_3 \approx \frac{\lambda^2}{A_e}. \quad \text{Eq. (15.9)}$$

Eq. (15.9) indicates that the antenna beamwidth decreases as $\sqrt{A_e}$ increases. Thus, in surveillance operations, the number of beam positions an antenna will take on to cover a volume V is

$$N_{Beams} > \frac{V}{\theta_3 \phi_3}. \quad \text{Eq. (15.10)}$$

and when V represents the entire hemisphere, Eq. (15.10) is modified to

$$N_{Beams} > \frac{2\pi}{\theta_3 \phi_3} \approx \frac{2\pi A_e}{\lambda^2} \approx \frac{G}{2}. \quad \text{Eq. (15.11)}$$

15.2. Near and Far Fields

The electric field intensity generated from the energy emitted by an antenna is a function of the antenna physical aperture shape and the electric current amplitude and phase distribution across the aperture. Plots of the modulus of the electric field intensity of the emitted radiation, $|E(\theta, \phi)|$, are referred to as the *intensity pattern* of the antenna. Alternatively, plots of $|E(\theta, \phi)|^2$ are called the *power radiation pattern* (the same as $P(\theta, \phi)$).

Based on the distance from the face of the antenna, where the radiated electric field is measured, three distinct regions are identified. They are the near field, Fresnel, and the Fraunhofer regions. In the near field and the Fresnel regions, rays emitted from the antenna have spherical wavefronts (equiphase fronts). In the Fraunhofer region, the wavefronts can be locally represented by plane waves. The near field and the Fresnel regions are normally of little interest to most radar applications. Most radar systems operate in the Fraunhofer region, which is also known as the far field region. In the far field region, the electric field intensity can be computed from the aperture Fourier transform.

Construction of the far criterion can be developed with the help of Fig. 15.1. Consider a radiating source at point O that emits spherical waves. A receiving antenna of length d is at distance r away from the source. The phase difference between a spherical wave and a local plane wave at the receiving antenna can be expressed in terms of the distance δr. The distance δr is given by

$$\delta r = \overline{AO} - \overline{OB} = \sqrt{r^2 + \left(\frac{d}{2}\right)^2} - r, \qquad \textbf{Eq. (15.12)}$$

and since in the far field $r \gg d$, Eq. (15.12) is approximated via binomial expansion by

$$\delta r = r\left(\sqrt{1 + \left(\frac{d}{2r}\right)^2} - 1\right) \approx \frac{d^2}{8r}. \qquad \textbf{Eq. (15.13)}$$

It is customary to assume far field when the distance δr corresponds to less than $1/16$ of a wavelength (i.e., $22.5°$). More precisely, if

$$\delta r = d^2/8r \le \lambda/16, \qquad \textbf{Eq. (15.14)}$$

then a useful expression for far field is

$$r \ge 2d^2/\lambda. \qquad \textbf{Eq. (15.15)}$$

Note that far field is a function of both the antenna size and the operating wavelength.

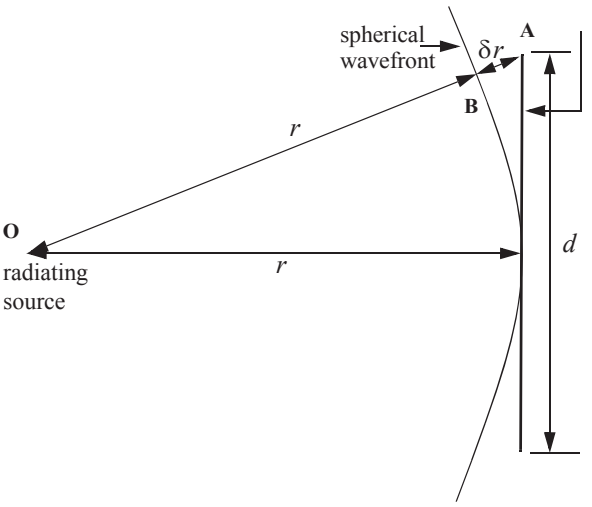


Figure 15.1. Construction of the far field criterion.

15.3. General Arrays

An array is a composite antenna formed from two or more basic radiators. Each radiator is denoted as an element. The elements forming an array could be dipoles, dish reflectors, slots in a wave guide, or any other type of radiator. Array antennas synthesize narrow directive beams that may be steered, mechanically or electronically, in many directions. Electronic steering is

achieved by controlling the phase of the current feeding the array elements. Arrays with electronic beam steering capability are called phased arrays. Phased array antennas, when compared to other simple antennas such as dish reflectors, are costly and complicated to design. However, the inherent flexibility of phased array antennas to steer the beam electronically, and also the need for specialized multifunction radar systems, have made phased array antennas attractive for radar applications.

Figure 15.2 shows the geometrical fundamentals associated with this problem. In general, consider the radiation source located at (x_1, y_1, z_1) with respect to a phase reference at $(0, 0, 0)$. The electric field measured at far field point P is

$$E(\theta, \phi) = I_0 \frac{e^{-jkR_1}}{R_1} f(\theta, \phi) \qquad \textbf{Eq. (15.16)}$$

where I_0 is the complex amplitude, $k = 2\pi/\lambda$ is the wave number, and $f(\theta, \phi)$ is the radiation pattern.

Now, consider the case where the radiation source is an array made of many elements, as shown in Fig. 15.3. The coordinates of each radiator with respect to the phase reference is (x_i, y_i, z_i), and the vector from the origin to the ith element is given by

$$\vec{r}_i = \hat{a}_x x_i + \hat{a}_y y_i + \hat{a}_z z_i . \qquad \textbf{Eq. (15.17)}$$

The far field components that constitute the total electric field are

$$E_i(\theta, \phi) = I_i \frac{e^{-jkR_i}}{R_i} f(\theta_i, \phi_i) \qquad \textbf{Eq. (15.18)}$$

where

$$\begin{aligned} R_i &= \left|\vec{R}_i\right| = \left|\vec{r} - \vec{r}_i\right| = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \\ &= r\sqrt{1 + \frac{(x_i^2 + y_i^2 + z_i^2)}{r^2} - 2\frac{(xx_i + yy_i + zz_i)}{r^2}} \end{aligned} \qquad \textbf{Eq. (15.19)}$$

Using spherical coordinates, where $x = r\sin\theta\cos\varphi$, $y = r\sin\theta\sin\varphi$, and $z = r\cos\theta$, yields

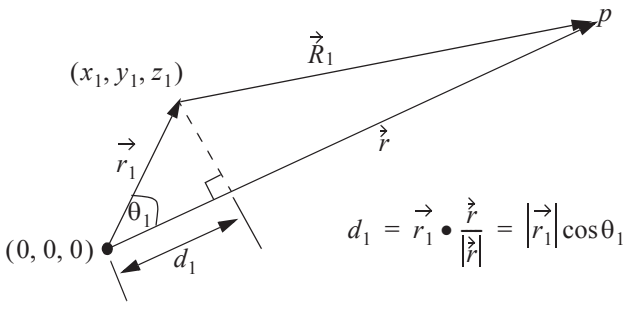


Figure 15.2. Geometry for an array antenna. Single element.

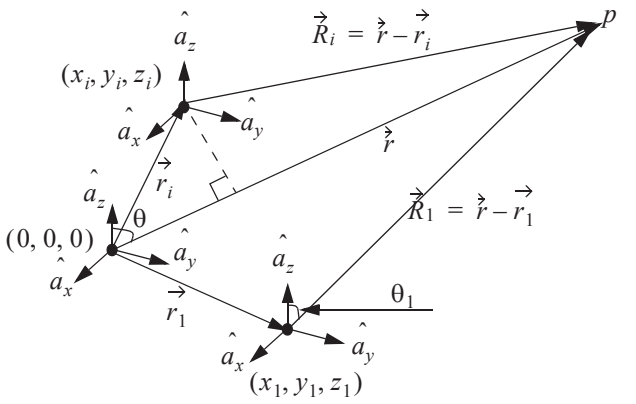


Figure 15.3. Geometry for an array antenna.

$$\frac{(x_i^2 + y_i^2 + z_i^2)}{r^2} = \frac{|\vec{r}_i|^2}{r^2} \ll 1 . \qquad \textbf{Eq. (15.20)}$$

Thus, a good approximation (using binomial expansion) for Eq. (15.19) is

$$R_i = r - r(x_i \sin\theta\cos\phi + y_i \sin\theta\sin\phi + z_i\cos\theta) . \qquad \textbf{Eq. (15.21)}$$

It follows that the phase contribution at the far field point from the ith radiator with respect to the phase reference is

$$e^{-jkR_i} = e^{-jkr}\ e^{jk(x_i \sin\theta\cos\phi + y_i \sin\theta\sin\phi + z_i\cos\theta)} . \qquad \textbf{Eq. (15.22)}$$

Remember, however, that the unit vector $\vec{r}_0$ along the vector $\vec{r}$ is

$$\vec{r}_0 = \frac{\vec{r}}{|\vec{r}|} = \hat{a}_x \sin\theta\cos\phi + \hat{a}_y \sin\theta\sin\phi + \hat{a}_z\cos\theta . \qquad \textbf{Eq. (15.23)}$$

Hence, we can rewrite Eq. (15.22) as

$$e^{-jkR_i} = e^{-jkr}\ e^{jk(\vec{r}_i \bullet \vec{r}_0)} = e^{-jkr} e^{j\Psi_i(\theta, \phi)} . \qquad \textbf{Eq. (15.24)}$$

Finally, by virtue of superposition, the total electric field is

$$E(\theta, \phi) = \sum_{i=1}^{N} I_i e^{j\Psi_i(\theta, \phi)} , \qquad \textbf{Eq. (15.25)}$$

which is known as the array factor for an array antenna where the complex current for the ith element is I_i.

In general, an array can be fully characterized by its array factor. This is true since knowing the array factor provides the designer with knowledge of the array's (1) 3-dB beamwidth; (2) null-to-null beamwidth; (3) distance from the main peak to the first sidelobe; (4) height of the first sidelobe as compared to the main beam; (5) location of the nulls; (6) rate of decrease of the sidelobes; and (7) grating lobe locations.

15.4. Linear Arrays

Figure 15.4 shows a linear array antenna consisting of N identical elements. The element spacing is d (normally measured in wavelength units). Let element #1 serve as a phase reference for the array. From the geometry, it is clear that an outgoing wave at the nth element leads the phase at the $(n+1)th$ element by $kd\sin\theta$, where $k = 2\pi/\lambda$. The combined phase at the far field observation point P is independent of ϕ and is computed from Eq. (15.24) as

$$\Psi(\theta,\phi) = k(\vec{r}_n \bullet \vec{r}_0) = (n-1)kd\sin\theta \,. \qquad \textbf{Eq. (15.26)}$$

Thus, from Eq. (15.25), the electric field at a far field observation point with direction-sine equal to $\sin\theta$ (assuming isotropic elements) is

$$E(\sin\theta) = \sum_{n=1}^{N} e^{j(n-1)(kd\sin\theta)} \,. \qquad \textbf{Eq. (15.27)}$$

Expanding the summation in Eq. (15.27) yields

$$E(\sin\theta) = 1 + e^{jkd\sin\theta} + \ldots + e^{j(N-1)(kd\sin\theta)} \,. \qquad \textbf{Eq. (15.28)}$$

The right-hand side of Eq. (15.28) is a geometric series, which can be expressed in the form

$$1 + a + a^2 + a^3 + \ldots + a^{(N-1)} = \frac{1-a^N}{1-a} \,. \qquad \textbf{Eq. (15.29)}$$

Replacing a by $e^{jkd\sin\theta}$ yields

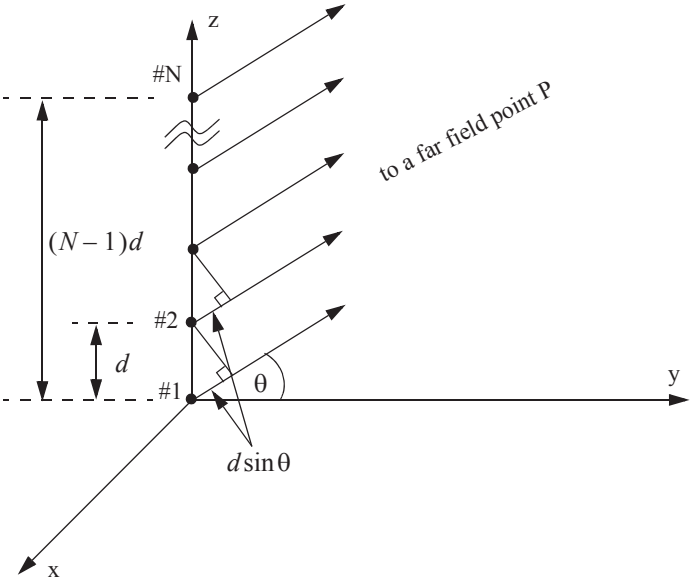


Figure 15.4. Linear array of equally spaced elements.

$$E(\sin\theta) = \frac{1 - e^{jNkd\sin\theta}}{1 - e^{jkd\sin\theta}} = \frac{1 - (\cos Nkd\sin\theta) - j(\sin Nkd\sin\theta)}{1 - (\cos kd\sin\theta) - j(\sin kd\sin\theta)} . \quad \textbf{Eq. (15.30)}$$

The far field array intensity pattern is then given by

$$|E(\sin\theta)| = \sqrt{E(\sin\theta)E^*(\sin\theta)} . \quad \textbf{Eq. (15.31)}$$

Substituting Eq. (15.30) into Eq. (15.31) and collecting terms yields

$$|E(\sin\theta)| = \sqrt{\frac{(1 - \cos Nkd\sin\theta)^2 + (\sin Nkd\sin\theta)^2}{(1 - \cos kd\sin\theta)^2 + (\sin kd\sin\theta)^2}} = \sqrt{\frac{1 - \cos Nkd\sin\theta}{1 - \cos kd\sin\theta}} , \quad \textbf{Eq. (15.32)}$$

and using the trigonometric identity $1 - \cos\theta = 2(\sin\theta/2)^2$ yields

$$|E(\sin\theta)| = \left|\frac{\sin(Nkd\sin\theta/2)}{\sin(kd\sin\theta/2)}\right| , \quad \textbf{Eq. (15.33)}$$

which is a periodic function of $kd\sin\theta$, with a period equal to 2π.

The maximum value of $|E(\sin\theta)|$, which occurs at $\theta = 0$, is equal to N. It follows that the normalized intensity pattern is equal to

$$|E_n(\sin\theta)| = \frac{1}{N}\left|\frac{\sin((Nkd\sin\theta)/2)}{\sin((kd\sin\theta)/2)}\right| . \quad \textbf{Eq. (15.34)}$$

The normalized two-way array pattern (radiation pattern) is given by

$$G(\sin\theta) = |E_n(\sin\theta)|^2 = \frac{1}{N^2}\left(\frac{\sin((Nkd\sin\theta)/2)}{\sin((kd\sin\theta)/2)}\right)^2 . \quad \textbf{Eq. (15.35)}$$

Figure 15.5 shows a plot of Eq. (15.35) versus $\sin\theta$ for $N = 8$. The pattern $G(\sin\theta)$ has cylindrical symmetry about its axis $(\sin\theta = 0)$, and is independent of the azimuth angle. Thus, it is completely determined by its values within the interval $(0 < \theta < \pi)$. This figure can be reproduced using MATLAB program *"Fig15_5.m,"* listed in Appendix 15-A.

The main beam of an array can be steered electronically by varying the phase of the current applied to each array element. Steering the main beam into the direction-sine $\sin\theta_0$ is accomplished by making the phase difference between any two adjacent elements equal to $kd\sin\theta_0$. In this case, the normalized radiation pattern can be written as

$$G(\sin\theta) = \frac{1}{N^2}\left(\frac{\sin[(Nkd/2)(\sin\theta - \sin\theta_0)]}{\sin[(kd/2)(\sin\theta - \sin\theta_0)]}\right)^2 \quad \textbf{Eq. (15.36)}$$

If $\theta_0 = 0$,then the main beam is perpendicular to the array axis, and the array is said to be a broadside array. Alternatively, the array is called an endfire array when the main beam points along the array axis.

The radiation pattern maxima are computed using L'Hopital's rule when both the denominator and numerator of Eq. (15.35) are zeros. More precisely,

$$\left(\frac{kd\sin\theta}{2} = \pm m\pi\right); \; m = 0, 1, 2, \ldots \quad \textbf{Eq. (15.37)}$$

Solving for θ yields

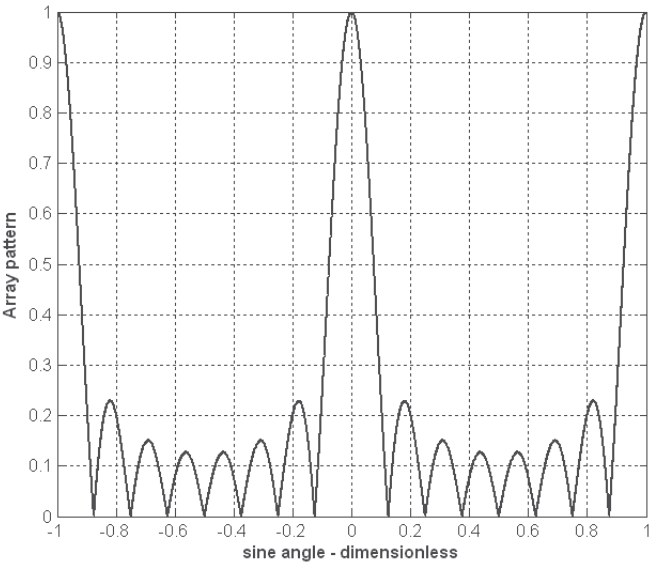


Figure 15.5a. Normalized radiation pattern for a linear array; $N = 8$; $d = \lambda$.

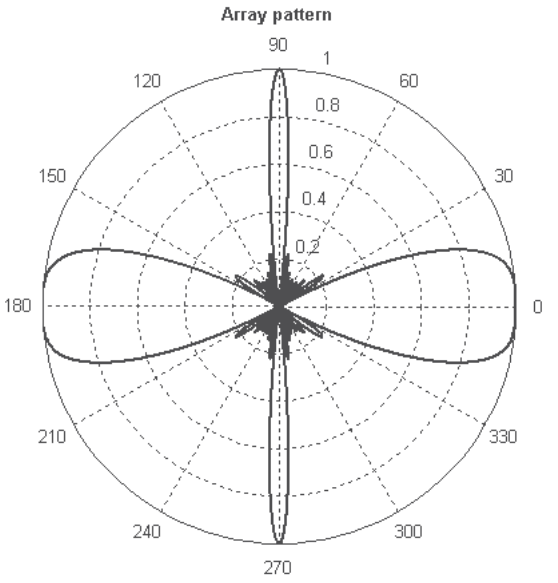


Figure 15.5b. Polar plot for the array pattern in Fig. 15.5a.

$$\theta_m = \text{asin}\left(\pm\frac{\lambda m}{d}\right);\ \ m = 0, 1, 2, \ldots \qquad \textbf{Eq. (15.38)}$$

where the subscript m is used as a maxima indicator. The first maximum occurs at $\theta_0 = 0$, and is denoted as the main beam (lobe). Other maxima occurring at $|m| \geq 1$ are called grating lobes. Grating lobes are undesirable and must be suppressed. The grating lobes occur at non-real angles when the absolute value of the arc-sine argument in Eq. (15.38) is greater than unity; it follows that $d < \lambda$. Under this condition, the main lobe is assumed to be at $\theta = 0$ (broadside array). Alternatively, when electronic beam steering is considered, the grating lobes occur at

$$\left|\sin\theta - \sin\theta_0\right| = \pm\frac{\lambda n}{d};\ \ n = 1, 2, \ldots \qquad \textbf{Eq. (15.39)}$$

Thus, in order to prevent the grating lobes from occurring between $\pm 90°$, the element spacing should be $d < \lambda/2$.

The radiation pattern attains secondary maxima (sidelobes) when the numerator of Eq. (15.35) is maximum, or equivalently

$$\frac{Nkd\sin\theta}{2} = \pm(2l+1)\frac{\pi}{2};\ \ l = 1, 2, \ldots \qquad \textbf{Eq. (15.40)}$$

Solving for θ yields

$$\theta_l = \text{asin}\left(\pm\frac{\lambda}{2d}\frac{2l+1}{N}\right);\ \ l = 1, 2, \ldots \qquad \textbf{Eq. (15.41)}$$

where the subscript l is used as an indication of sidelobe maxima. The nulls of the radiation pattern occur when only the numerator of Eq. (15.35) is zero. More precisely,

$$\frac{N}{2}kd\sin\theta = \pm n\pi;\ \begin{matrix} n = 1, 2, \ldots \\ n \neq N, 2N, \ldots \end{matrix} \qquad \textbf{Eq. (15.42)}$$

Again solving for θ yields

$$\theta_n = \text{asin}\left(\pm\frac{\lambda}{d}\frac{n}{N}\right);\ \begin{matrix} n = 1, 2, \ldots \\ n \neq N, 2N, \ldots \end{matrix} \qquad \textbf{Eq. (15.43)}$$

where the subscript n is used as a null indicator. Define the angle that corresponds to the half power point as θ_h. It follows that the half power ($3dB$) beamwidth is $2\left|\theta_m - \theta_h\right|$. This occurs when

$$\frac{N}{2}kd\sin\theta_h = 1.391\ \ radians \Rightarrow \theta_h = \text{asin}\left(\frac{\lambda}{2\pi d}\frac{2.782}{N}\right). \qquad \textbf{Eq. (15.44)}$$

15.4.1. Array Tapering

Figure 15.6 shows a normalized two-way radiation pattern of a uniformly excited linear array of size $N = 8$, element spacing $d = \lambda/2$. The first sidelobe is $13.46dB$ below the main lobe, and for most radar applications this may not be sufficient, particularly in the presence of a strong source of jamming or high levels of noise. Under such conditions, target detection in the main beam becomes rather challenging, since the SNR is reduced.

In order to reduce the sidelobe levels, the array must be designed to radiate more power toward the center, and much less at the edges. This can be achieved through tapering (windowing) the current distribution over the face of the array. There are many possible tapering sequences that can be used for this purpose. However, as known from spectral analysis, windowing reduces sidelobe levels at the expense of widening the main beam. Thus, for a given radar application, the choice of the tapering sequence must be based on the trade-off between sidelobe reduction and main-beam widening. The same type of windows discussed earlier in Chapter 3 can be used for array tapering. Table 15.1 summarizes the impact of most common windows on the array pattern in terms of main-beam widening and peak reduction. Note that the rectangular window is used as the baseline. This is also illustrated in Fig. 15.7, which can be reproduced using MATLAB program *"Fig15_7.m,"* listed in Appendix 15-A.

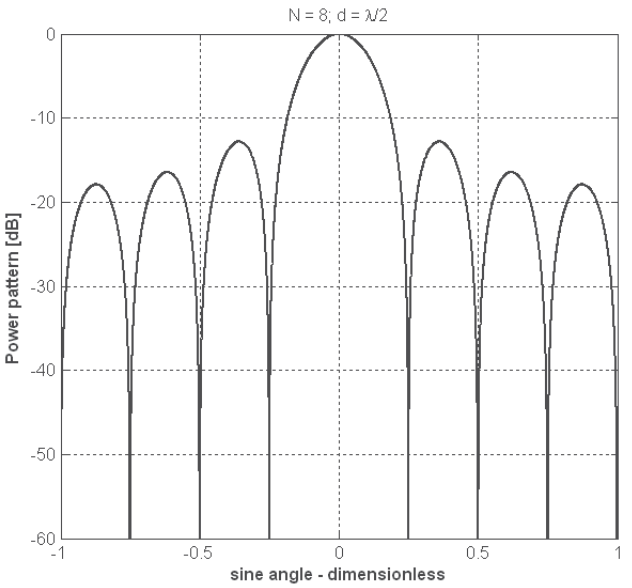


Figure 15.6. Normalized pattern for a linear array. $N = 8$, $d = \lambda/2$.

TABLE 15.1. Common windows.

Window	Null-to-Null Beamwidth	Peak Reduction
Rectangular	*1*	*1*
Hamming	*2*	*0.73*
Hanning	*2*	*0.664*
Blackman	*6*	*0.577*
Kaiser ($\beta = 6$)	*2.76*	*0.683*
Kaiser ($\beta = 3$)	*1.75*	*0.882*

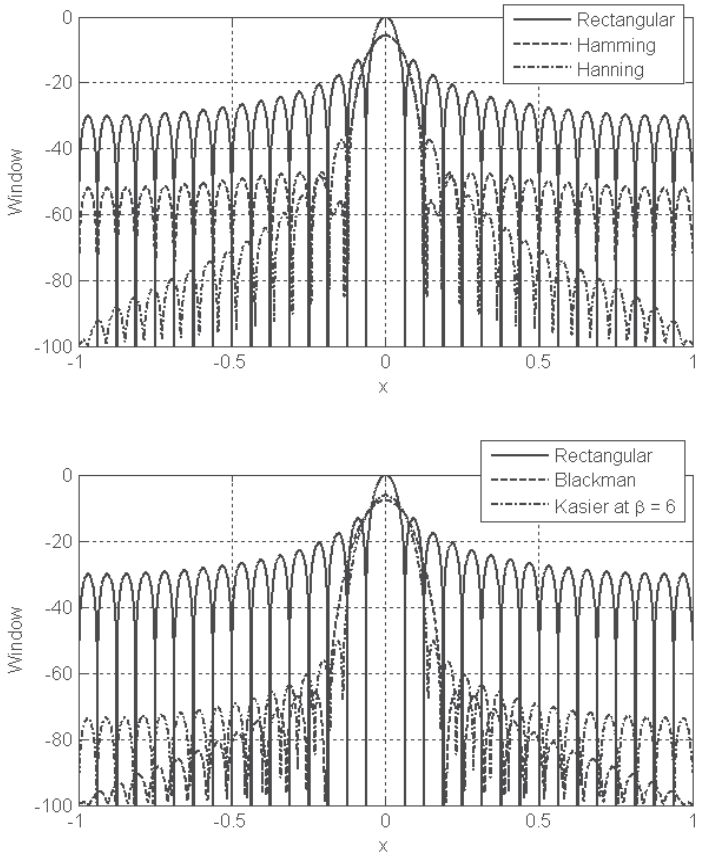


Figure 15.7. Most common windows.

15.4.2. Computation of the Radiation Pattern via the DFT

Figure 15.8 shows a linear array of size N, element spacing d, and wavelength λ. The radiators are circular dishes of diameter d. Let $w(n)$ and $\Phi(n)$, respectively, denote the tapering and phase shifting sequences. The normalized electric field at a far field point in the direction-sine $\sin\theta$ is

$$E(\sin\theta) = \sum_{n=0}^{N-1} w(n) e^{j\Delta\theta\left(n-\left(\frac{N-1}{2}\right)\right)} \qquad \textbf{Eq. (15.45)}$$

where in this case the phase reference is taken as the physical center of the array, and

$$\Delta\theta = \frac{2\pi d}{\lambda}\sin\theta\,. \qquad \textbf{Eq. (15.46)}$$

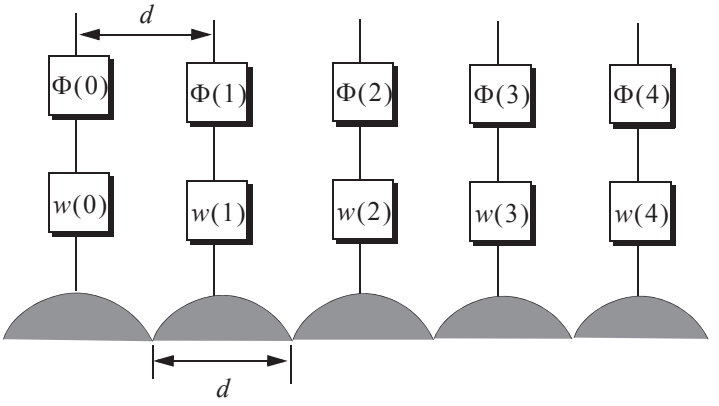


Figure 15.8. Linear array of size 5, with tapering and phase shifting hardware.

Expanding Eq. (15.45) and factoring the common phase term $\exp[j(N-1)\Delta\theta/2]$ yields

$$E(\sin\theta) = e^{j(N-1)\Delta\theta/2}\{w(0)e^{-j(N-1)\Delta\theta} + w(1)e^{-j(N-2)\Delta\theta} + \ldots + w(N-1)\}. \quad \textbf{Eq. (15.47)}$$

By using the symmetry property of a window sequence (remember that a window must be symmetrical about its central point), we can rewrite Eq. (15.47) as

$$E(\sin\theta) = e^{j\theta_0}\{w(N-1)e^{-j(N-1)\Delta\theta} + w(N-2)e^{-j(N-2)\Delta\theta} + \ldots + w(0)\} \quad \textbf{Eq. (15.48)}$$

where $\theta_0 = (N-1)\Delta\theta/2$.

Define $\{V_1^n = \exp(-jn\Delta\theta); n = 0, 1, \ldots, N-1\}$. It follows that

$$E(\sin\theta) = e^{j\theta_0}[w(0) + w(1)V_1^1 + \ldots + w(N-1)V_1^{N-1}] = e^{j\theta_0}\sum_{n=0}^{N-1} w(n)V_1^n. \quad \textbf{Eq. (15.49)}$$

The discrete Fourier transform of the sequence $w(n)$ is defined as

$$W(q) = \sum_{n=0}^{N-1} w(n)e^{-\frac{(j2\pi nq)}{N}} \quad ; \; q = 0, 1, \ldots, N-1. \quad \textbf{Eq. (15.50)}$$

The set $\{\sin\theta_q\}$ that makes V_1 equal to the DFT kernel is

$$\sin\theta_q = \frac{\lambda q}{Nd}; \; q = 0, 1, \ldots, N-1. \quad \textbf{Eq. (15.51)}$$

Then, by using Eq. (15.51) in Eq. (15.50) yields

$$E(\sin\theta) = e^{j\phi_0}W(q). \quad \textbf{Eq. (15.52)}$$

The one-way array pattern is computed as the modulus of Eq. (15.52). It follows that the one-way radiation pattern of a tapered linear array of circular dishes is

$$G(\sin\theta) = G_e \ |W(q)| \qquad \textbf{Eq. (15.53)}$$

where G_e is the element pattern. In practice, phase shifters are normally implemented as part of the Transmit/Receive (TR) modules, using a finite number of bits. Consequently, due to the quantization error (difference between desired phase and actual quantized phase) the sidelobe levels are affected.

MATLAB Function "linear_array.m"

The function *"linear_array.m"* computes and plots the linear array gain pattern as a function of real sine-space. The syntax is as follows:

[theta, patternr, patterng] = linear_array(Nr, dolr, theta0, winid, win, nbits)

where

Symbol	Description	Units	Status
Nr	*number of elements in array*	*none*	*input*
dolr	*element spacing in lambda units*	*wavelengths*	*input*
theta0	*steering angle*	*degrees*	*input*
winid	*-1== No weighting; 1== weighting = user specified window*	*none*	*input*
win	*window for sidelobe control*	*none*	*input*
nbits	*negative #: perfect quantization* *positive #: use* 2^{nbits} *quantization levels*	*none*	*input*
theta	*real angle available for steering*	*degrees*	*output*
patternr	*array pattern*	*dB*	*output*
patterng	*gain pattern*	*dB*	*output*

The MATLAB-based graphical user interface (GUI) in Fig. 15.9 implements this function. This GUI was used to produce Figs. 15.10 through 15.18 assuming the following cases:

[theta, patternr, patterng] = linear_array(19, 0.5, 0, -1, -1, -3);

[theta, patternr, patterng] = linear_array(19, 0.5, 0, 1, 'hamming', -3);

[theta, patternr, patterng] = linear_array(19, 0.5, 5, -1, -1, 3);

[theta, patternr, patterng] = linear_array(19, 0.5, 5, 1, 'hamming', 3);

[theta, patternr, patterng] = linear_array(19, 0.5, 25, 1, 'hamming', 3);

[theta, patternr, patterng] = linear_array(19, 1.5, 48, -1, -1, -3);

[theta, patternr, patterng] = linear_array(19, 1.5, 48, 1, 'hamming', -3);

[theta, patternr, patterng] = linear_array(19, 1.5, -48, -1, -1, 3);

[theta, patternr, patterng] = linear_array(19, 1.5, -38, 1, 'hamming', 3);

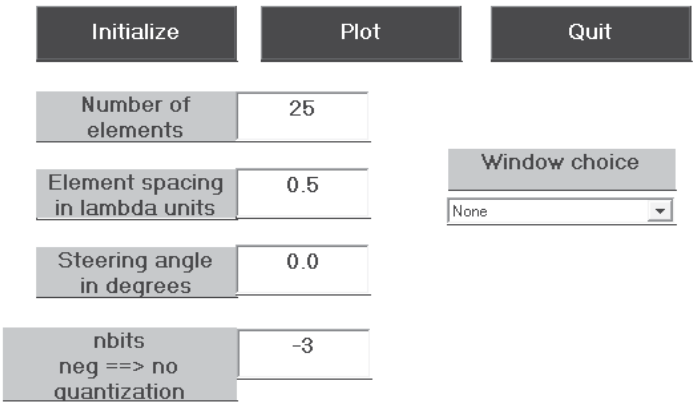


Figure 15.9. MATLAB GUI workspace associated with the function *"linear_array.m."*

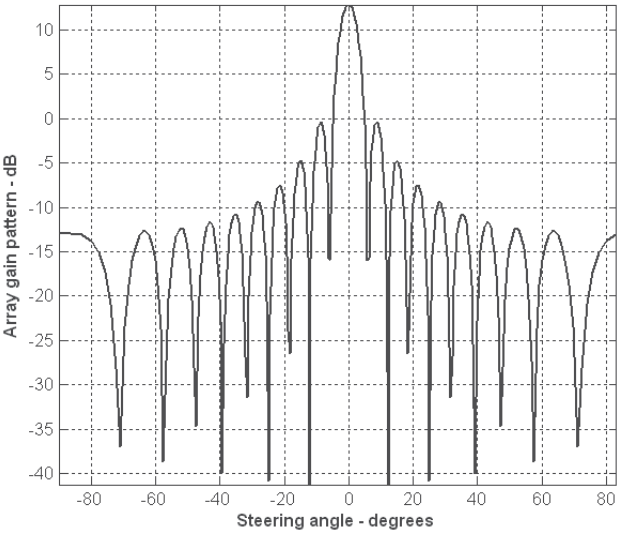


Figure 15.10. Array gain pattern: $Nr = 19;\ dolr = 0.5;\ \theta_0 = 0°;$ $win = none;\ nbits = -3.$

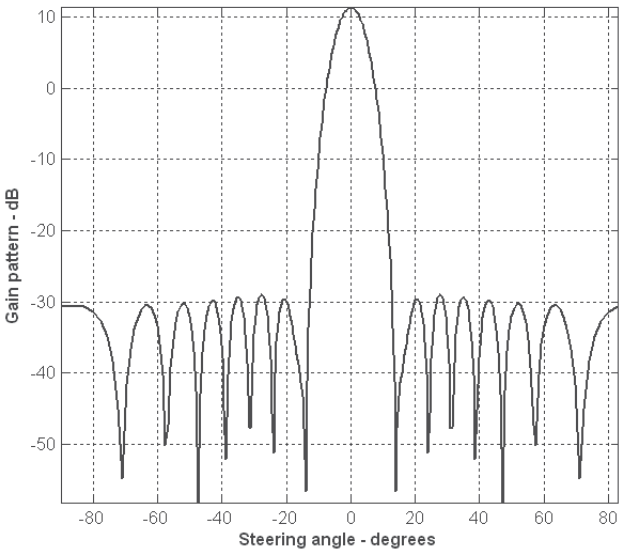


Figure 15.11. Array gain pattern: $Nr = 19;\ dolr = 0.5;\ \theta_0 = 0°$**;** $win = Hamming;\ nbits = -3$

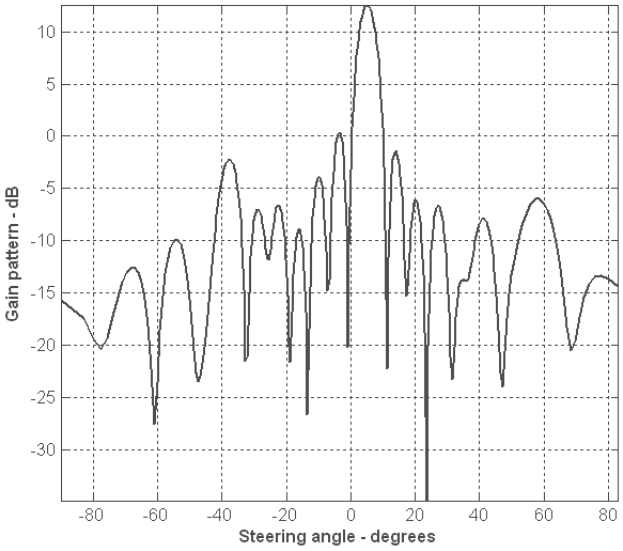


Figure 15.12. Array gain pattern: $Nr = 19;\ dolr = 0.5;\ \theta_0 = 5°$**;** $win = none;\ nbits = 3$

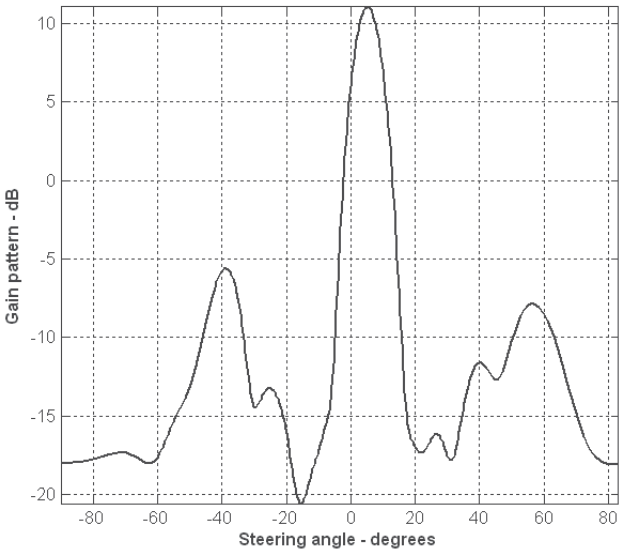


Figure 15.13. Array gain pattern: $Nr = 19;\ dolr = 0.5;\ \theta_0 = 5°$ **;** $win = Hamming;\ nbits = 3$

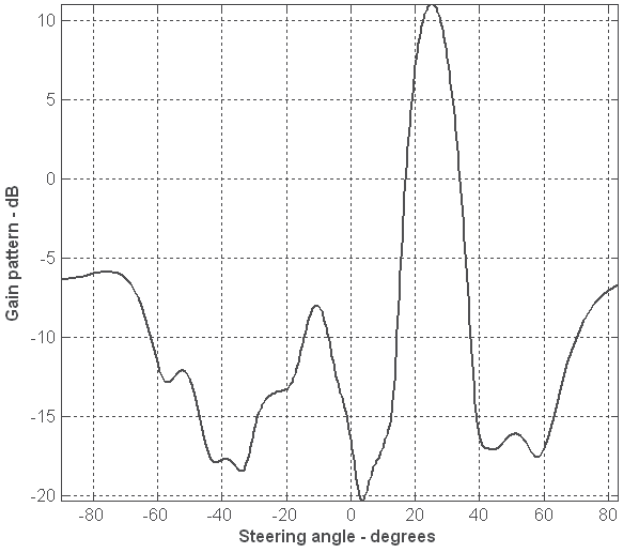


Figure 15.14. Array gain pattern: $Nr = 19;\ dolr = 0.5;\ \theta_0 = 25°$ **;** $win = Hamming;\ nbits = 3$

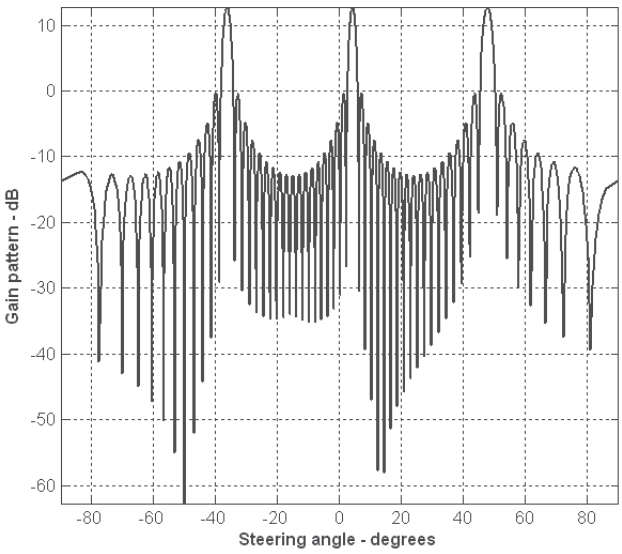


Figure 15.15. Array gain pattern: $Nr = 19;\ dolr = 1.5;\ \theta_0 = 48°$ **;** $win = none;\ nbits = -3$

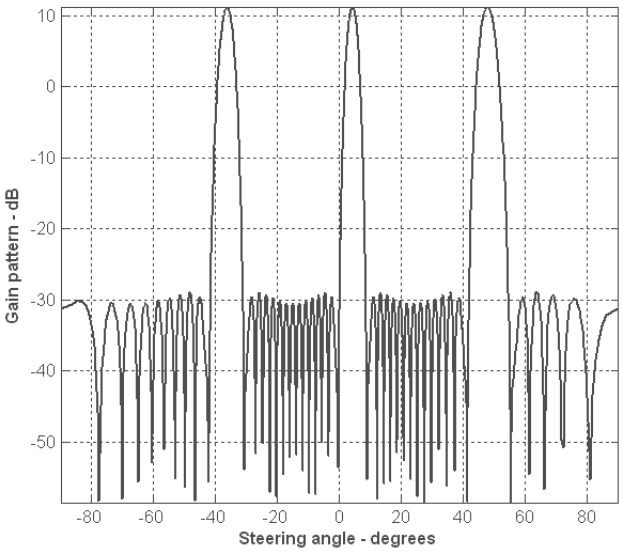


Figure 15.16. Array gain pattern: $Nr = 19;\ dolr = 1.5;\ \theta_0 = 48°$ **;** $win = Hamming;\ nbits = -3$

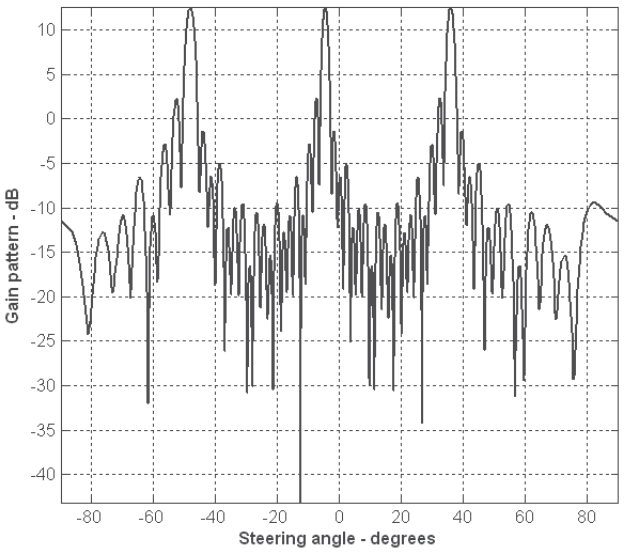


Figure 15.17. Array gain pattern: $Nr = 19;\ dolr = 1.5;\ \theta_0 = -48°$**;** *win = none*; *nbits* = 3

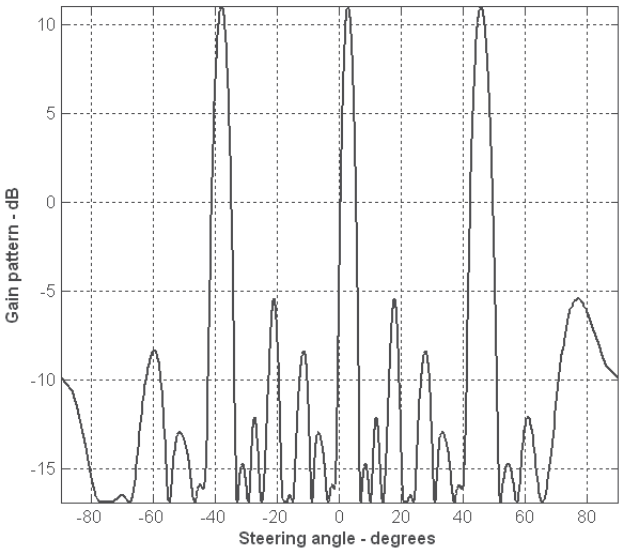


Figure 15.18. Array gain pattern: $Nr = 19;\ dolr = 1.5;\ \theta_0 = -38°$**;** *win = Hamming*; *nbits* = 3

15.5. Planar Arrays

Planar arrays are a natural extension of linear arrays. Planar arrays can take on many configurations, depending on the element spacing and distribution defined by a "grid." Examples include rectangular, rectangular with circular boundary, hexagonal with circular boundary, circular, and concentric circular grids, as illustrated in Fig. 15.19.

Planar arrays can be steered in elevation and azimuth $((\theta, \phi)$, as illustrated in Fig. 15.20 for a rectangular grid array. The element spacing along the x- and y-directions are respectively denoted by d_x and d_y. The total electric field at a far field observation point for any planar array can be computed using Eqs. (15.24) and (15.25).

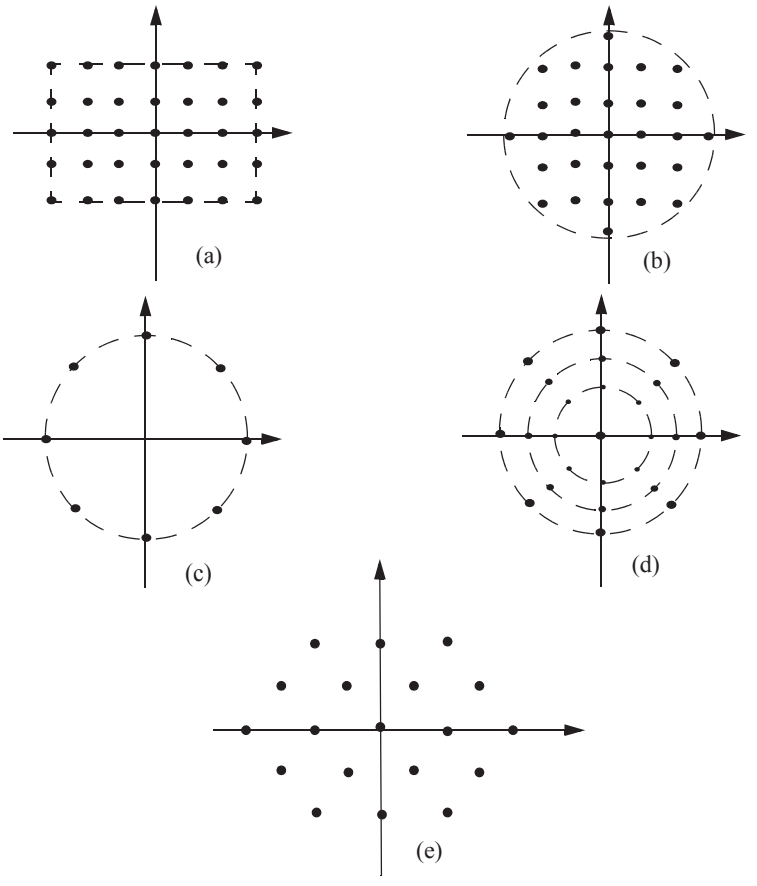


Figure 15.19. Planar array grids. (a) Rectangular, (b) rectangular with circular boundary, (c) circular, (d) concentric circular, and (e) hexagonal.

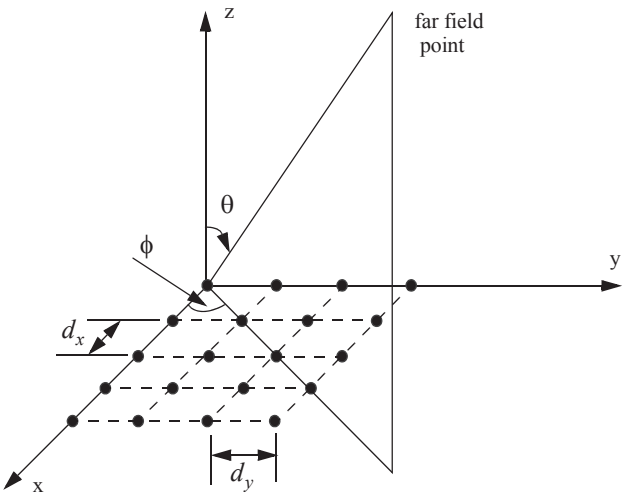


Figure 15.20. Rectangular array geometry.

15.5.1. Rectangular Grid Arrays

Consider the $N \times M$ rectangular grid as shown in Fig. 15.20. The dot product $\vec{r}_i \bullet \vec{r}_0$, where the vector $\vec{r}_i$ is the vector to the ith element in the array and $\vec{r}_0$ is the unit vector to the far field observation point, can be broken linearly into its *x*- and *y*-components. It follows that the electric field components due to the elements distributed along the x- and y-directions are, respectively,

$$E_x(\theta, \phi) = \sum_{n=1}^{N} I_{x_n} e^{j(n-1)kd_x \sin\theta \cos\phi} \qquad \textbf{Eq. (15.54)}$$

$$E_y(\theta, \phi) = \sum_{m=1}^{N} I_{y_m} e^{j(m-1)kd_y \sin\theta \sin\phi}. \qquad \textbf{Eq. (15.55)}$$

The total electric field at the far field observation point is then given by

$$E(\theta, \phi) = E_x(\theta, \phi) E_y(\theta, \phi) = \left(\sum_{m=1}^{N} I_{y_m} e^{j(m-1)kd_y \sin\theta \sin\phi} \right) \left(\sum_{n=1}^{N} I_{x_n} e^{j(n-1)kd_x \sin\theta \cos\phi} \right) \qquad \textbf{Eq. (15.56)}$$

Eq. (15.56) can be expressed in terms of the directional cosines

$$\begin{aligned} u &= \sin\theta \cos\phi \\ v &= \sin\theta \sin\phi \end{aligned} \qquad \textbf{Eq. (15.57)}$$

$$\phi = \operatorname{atan}\left(\frac{u}{v}\right)$$
$$\theta = \operatorname{asin}\sqrt{u^2 + v^2} \qquad \textbf{Eq. (15.58)}$$

the visible region is then defined by

$$\sqrt{u^2 + v^2} \le 1 \,. \qquad \textbf{Eq. (15.59)}$$

It is very common to express a planar array's ability to steer the beam in space in terms of the U, V space instead of the angles θ, ϕ. Figure 15.21 shows how a beam steered in a certain θ, ϕ direction is translated into U, V space.

The rectangular array one-way intensity pattern is then equal to the product of the individual patterns. More precisely for a uniform excitation ($I_{y_m} = I_{x_n} = const$),

$$E(\theta, \phi) = \left| \frac{\sin\left(\frac{Nkd_x \sin\theta\cos\phi}{2}\right)}{\sin\left(\frac{kd_x \sin\theta\cos\phi}{2}\right)} \right| \left| \frac{\sin\left(\frac{Nkd_y \sin\theta\sin\phi}{2}\right)}{\sin\left(\frac{kd_y \sin\theta\sin\phi}{2}\right)} \right| . \qquad \textbf{Eq. (15.60)}$$

The radiation pattern maxima, nulls, sidelobes, and grating lobes in both the *x*- and *y*-axes are computed in a similar fashion to the linear array case. Additionally, the same conditions for grating lobe control are applicable. Note the symmetry is about the angle ϕ.

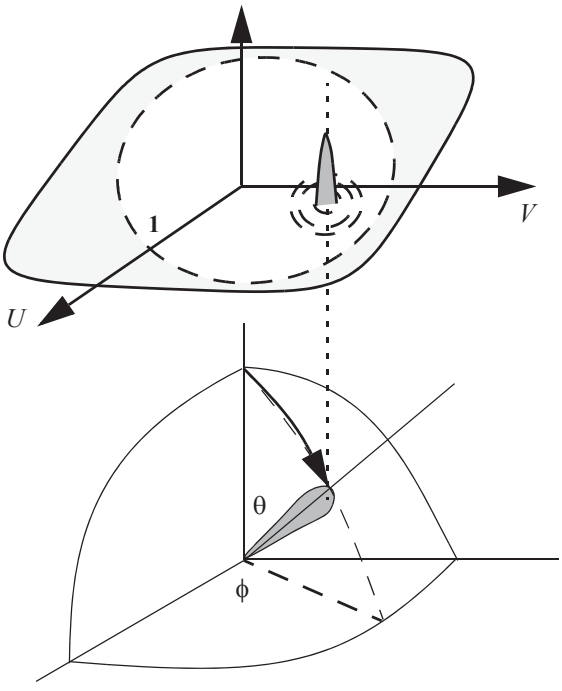


Figure 15.21. Translation from spherical coordinates into U,V space.

15.5.2. Circular Grid Arrays

The geometry of interest is shown in Fig. 15.19c. In this case, N elements are distributed equally on the outer circle whose radius is a. For this purpose, consider the geometry shown in Fig. 15.22. From the geometry

$$\Phi_n = \frac{2\pi}{N} n \quad ; \; n = 1, 2, \ldots, N. \qquad \textbf{Eq. (15.61)}$$

The coordinates of the nth element are

$$\begin{aligned} x_n &= a \cos\Phi_n \\ y_n &= a \sin\Phi_n . \\ z_n &= 0 \end{aligned} \qquad \textbf{Eq. (15.62)}$$

It follows that

$$k(\vec{r}_n \bullet \vec{r}_0) = \Psi_n = k(a\sin\theta\cos\phi\cos\Phi_n + a\sin\theta\sin\phi\sin\Phi_n + 0). \qquad \textbf{Eq. (15.63)}$$

Equation (15.63) can be rearranged as

$$\Psi_n = ak\sin\theta(\cos\phi\cos\Phi_n + \sin\phi\sin\Phi_n). \qquad \textbf{Eq. (15.64)}$$

Then, by using the identity $\cos(A-B) = \cos A\cos B + \sin A\sin B$, Eq.(15.63) collapses to

$$\Psi_n = ak\sin\theta\cos(\Phi_n - \phi). \qquad \textbf{Eq. (15.65)}$$

Finally, by using Eq. (15.25), the far field electric field is then given by

$$E(\theta, \phi; a) = \sum_{n=1}^{N} I_n \exp\left\{ j\frac{2\pi a}{\lambda}\sin\theta\cos(\Phi_n - \phi) \right\} \qquad \textbf{Eq. (15.66)}$$

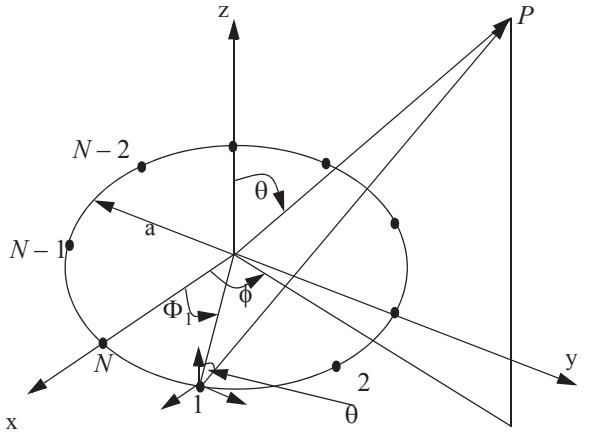


Figure 15.22. Geometry for a circular array.

where I_n represents the complex current distribution for the nth element. When the array main beam is directed in the (θ_0, ϕ_0), Eq. (15.65) takes on the following form

$$E(\theta, \phi; a) = \sum_{n=1}^{N} I_n \exp\left\{ j \frac{2\pi a}{\lambda} [\sin\theta \cos(\Phi_n - \phi) - \sin\theta_0 \cos(\Phi_n - \phi_0)] \right\}. \qquad \textbf{Eq. (15.67)}$$

MATLAB program "circular_array.m"

The MATLAB program *"circular_array.m"* calculates and plots the rectangular and polar array patterns for a circular array versus θ and ϕ constant planes. The input parameters to this program are:

Symbol	Description	Units
a	*Circular array radius*	λ
N	*number of elements*	*none*
theta0	*main direction in* θ	*degrees*
phi0	*main direction in* ϕ	*degrees*
Variations	*'Theta'; or 'Phi'*	*none*
phid	*constant* ϕ *plane*	*degrees*
thetad	*constant* θ *plane*	*degrees*

As an example, consider the following two cases with inputs defined in Table 15.2:

Table 15.2. Parameters to be used in Figs. 15.23 through 15.32

Parameter	Case I	Case II
a	*1.*	*1.5*
N	*10*	*10*
θ_0	*45*	*45*
ϕ_0	*60*	*60*
variation	*'Theta'*	*'Phi'*
ϕ_d	*60*	*60*
θ_d	*45*	*45*

Figures 15.23 and 15.24 respectively show the array patterns in relative amplitude and the power patterns versus the angle θ corresponding to Case I parameters. Figures 15.25 and 15.26 are similar to Figs. 15.23 and 15.24, except in this case the patterns are plotted in polar coordinates. Figure 15.27 shows a plot of the normalized single element pattern (upper left corner), the normalized array factor (upper right corner), the total array pattern (lower left corner). Figures 15.28 through 15.32 are similar to those in Figs. 15.23 through 15.27, except in this case the input parameters correspond to Case II.

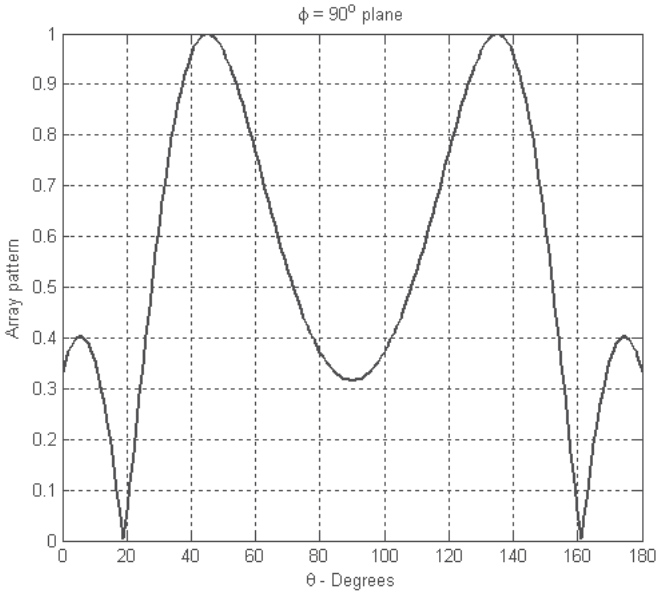


Figure 15.23. Array factor pattern for a circular array, using the parameters defined in Case I of Table 15.2.

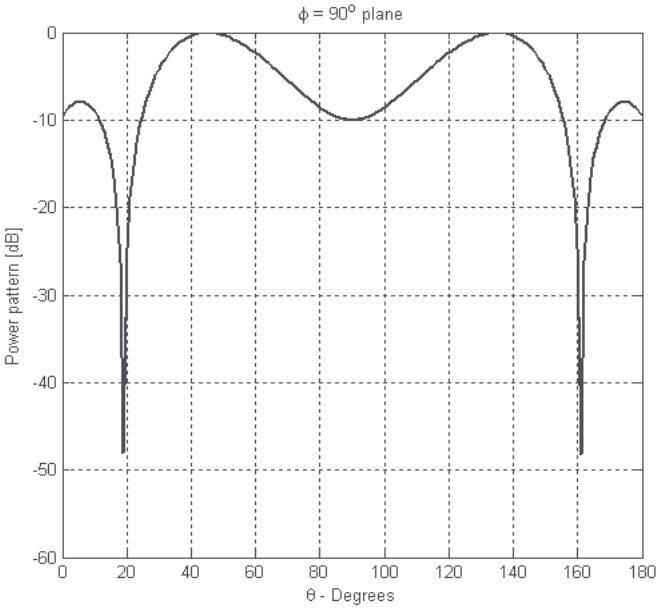


Figure 15.24. Same as Fig. 15.23 using dB scale.

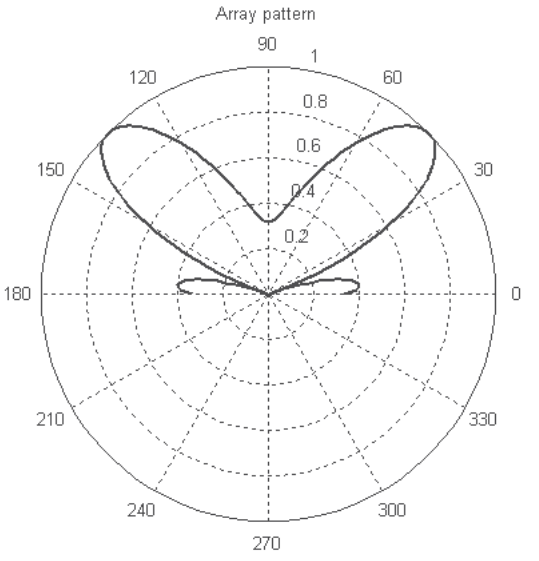


Figure 15.25. Array factor pattern for a circular array, using the parameters defined in Case I of Table 15.2.

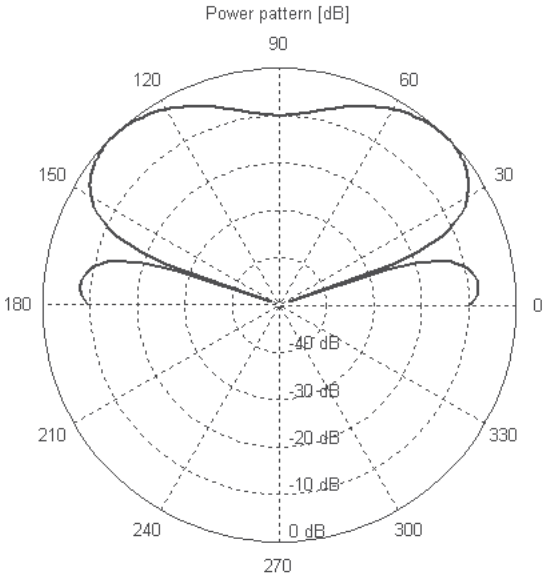


Figure 15.26. Same as Fig. 15.25 using dB scale.

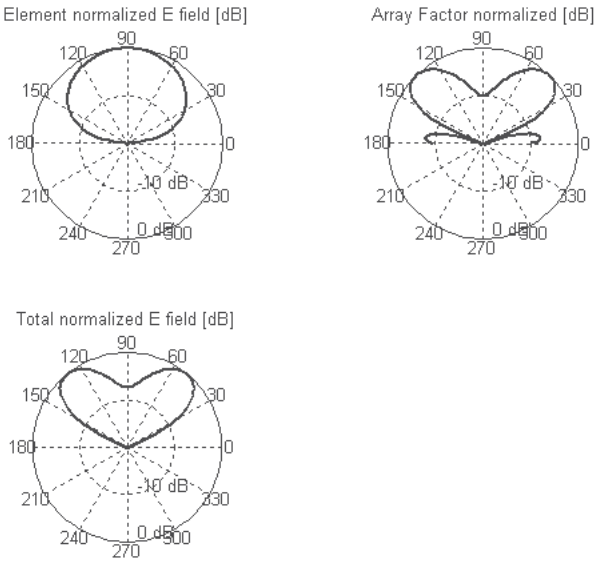


Figure 15.27. Element, array factor and total pattern using the parameters defined in Case I of Table 15.2.

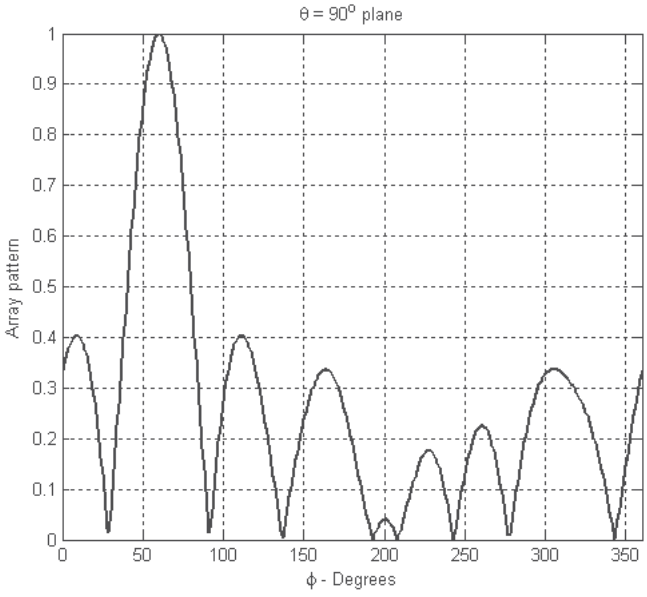


Figure 15.28. Array factor pattern for a circular array, using the parameters defined in Case II of Table 15.2.

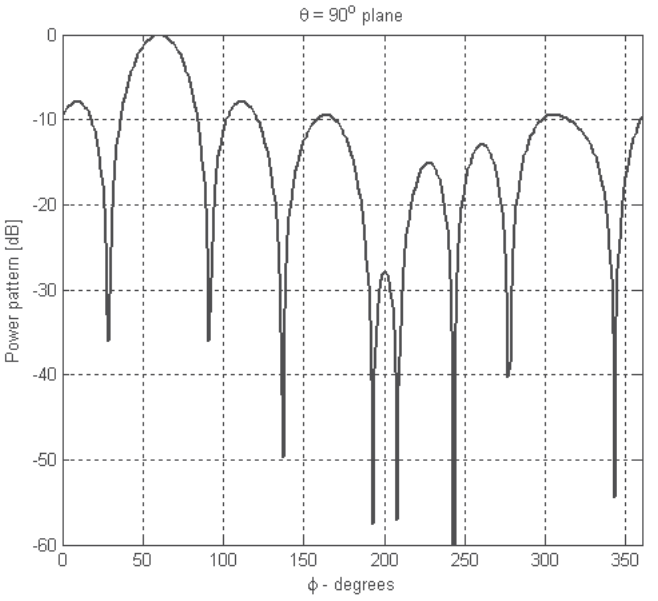


Figure 15.29. Same as Fig. 15.28 using dB scale.

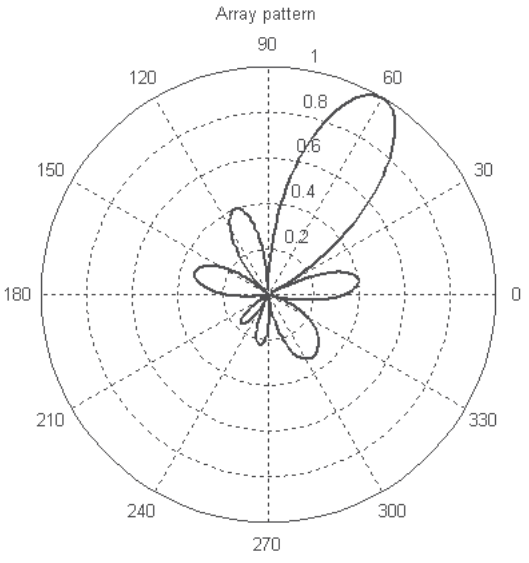


Figure 15.30. Array factor pattern for a circular array, using the parameters defined in Case II of Table 15.2.

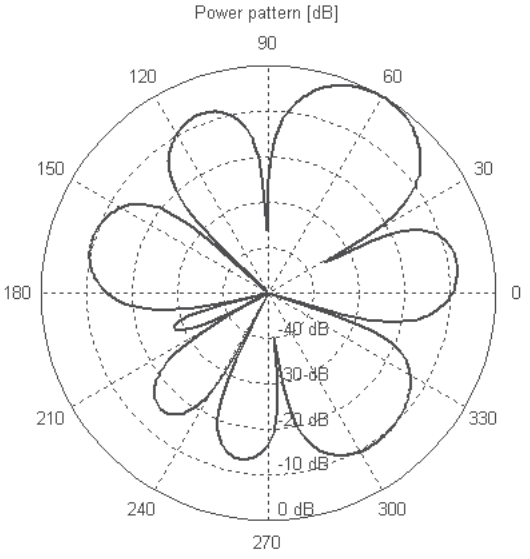


Figure 15.31. Same as Fig. 15.30 using dB scale.

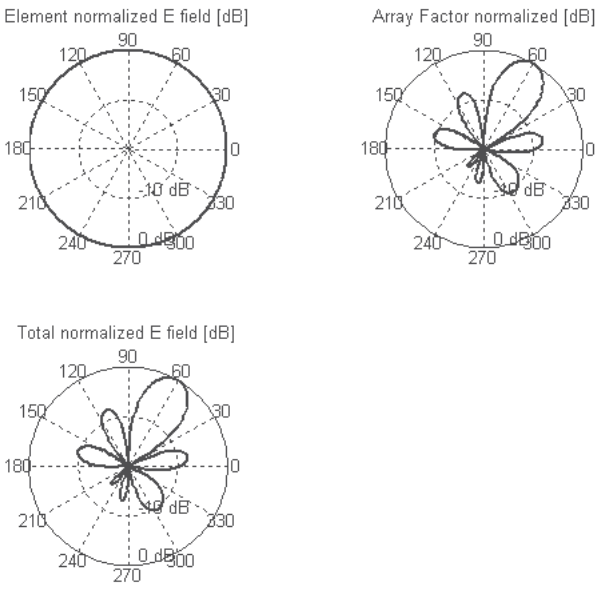


Figure 15.32. Element, array factor and total pattern for the circular using the parameters defined in Case II of Table 15.2.

15.5.3. Concentric Grid Circular Arrays

The geometry of interest is shown in Fig. 15.33. In this case, N_2 elements are distributed equally on the outer circle whose radius is a_2, while another N_1 elements are linearly distributed on the inner circle whose radius is a_1. The element located on the center of both circles is used as the phase reference. In this configuration, there are $N_1 + N_2 + 1$ total elements in the array.

The array factor is derived in two steps. First, the array factor corresponding to a linearly distributed circular array is computed. Second, the overall array factor corresponding to all elements will be the product of each individual circular array times the pattern of the central element. More precisely,

$$E(\theta, \phi) = E_1(\theta, \phi; a_1) E_2(\theta, \phi; a_2) E_0(\theta, \phi) \tag{8.68}$$

Fig. 15.34 shows a 3-D plot for a concentric circular array in the θ, ϕ space for the following parameters: $a_1 = 1\lambda$, $N_1 = 8 = N_2$, and $a_2 = 2\lambda$

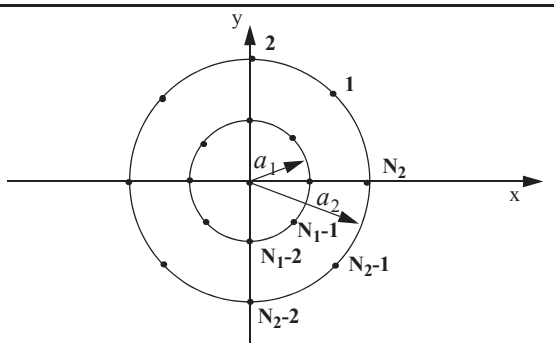


Figure 15.33. Concentric circular array geometry.

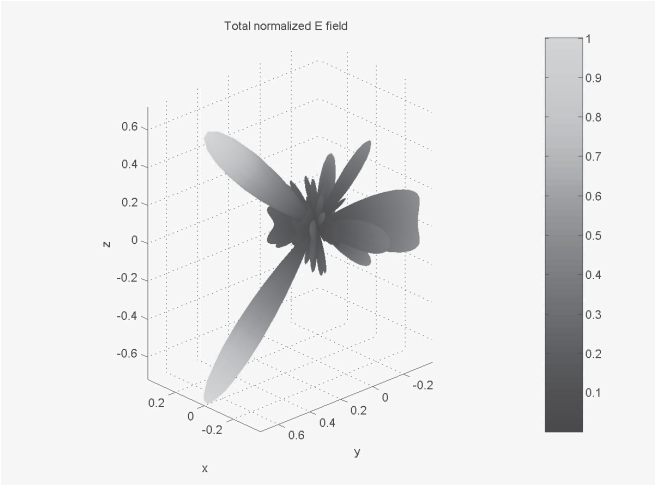


Figure 15.34. A 3-D array pattern - concentric circular array; $\theta = 45°$ and $\phi = 90°$.

15.5.4. Rectangular Grid with Circular Boundary Arrays

The far field electric field associated with this configuration can be easily obtained from that corresponding to a rectangular grid. In order to accomplish this task, follow these steps: First, select the desired maximum number of elements along the diameter of the circle and denote it by N_d. Also select the associated element spacings d_x, d_y. Define a rectangular array of size $N_d \times N_d$. Draw a circle centered at $(x, y) = (0, 0)$ with radius r_d where

$$r_d = \frac{N_d - 1}{2} + \Delta x \qquad \textbf{Eq. (15.69)}$$

and $\Delta x \le d_x/4$. Finally, modify the weighting function across the rectangular array by multiplying it with the two-dimensional sequence $a(m, n)$, where

$$a(m,n) = \begin{cases} 1 & , \ if \ \ dis \ \ to \ \ (m,n)th \ \ element < r_d \\ 0 & ; \ elsewhere \end{cases} \qquad \textbf{Eq. (15.70)}$$

where distance, dis, is measured from the center of the circle. This is illustrated in Fig. 15.35.

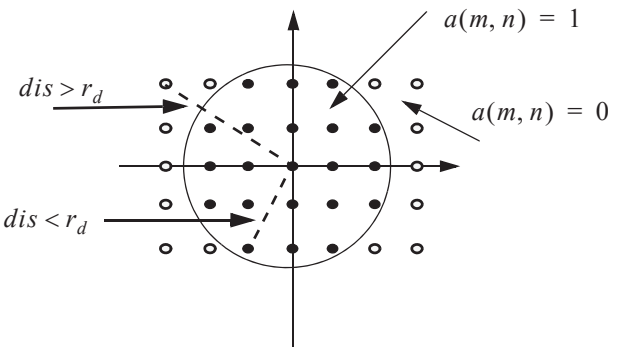


Figure 15.35. Elements with solid dots have $a(m, n) = 1$; other elements have $a(m, n) = 0$.

15.5.5. Hexagonal Grid Arrays

The analysis provided in this section is limited to hexagonal arrays with circular boundaries. The horizontal element spacing is denoted as d_x and the vertical element spacing is

$$d_y = \frac{\sqrt{3}}{2} \ d_x . \qquad \textbf{Eq. (15.71)}$$

The array is assumed to have the maximum number of identical elements along the x-axis ($y = 0$). This number is denoted by N_x, where N_x is an odd number, in order to obtain a symmetric array, where an element is present at $(x, y) = (0, 0)$. The number of rows in the array is denoted by M. The horizontal rows are indexed by m, which varies from $-(N_x - 1)/2$ to $(N_x - 1)/2$. The number of elements in the mth row is denoted by N_r and is defined by

$$N_r = N_x - |m| . \qquad \textbf{Eq. (15.72)}$$

The electric field at a far field observation point is computed using Eq. (15.24) and (15.25). The phase associated with $(m, n)th$ location is

$$\psi_{m,n} = \frac{2\pi d_x}{\lambda}\sin\theta\left[\left(m+\frac{n}{2}\right)\cos\phi + n\frac{\sqrt{3}}{2}\sin\phi\right]. \qquad \textbf{Eq. (15.73)}$$

MATLAB Function "rect_array.m"

The function *"rect_array.m"* computes and plots the rectangular antenna gain pattern in the visible *U,V* space. The syntax is as follows:

[pattern] = rect_array(Nxr, Nyr, dolxr, dolyr, theta0, phi0, winid, win, nbits)

where

Symbol	Description	Units	Status
Nxr	*number of elements along x*	*none*	*input*
Nyr	*number of elements along y*	*none*	*input*
dolxr	*element spacing in lambda units along x*	*wavelengths*	*input*
dolyr	*element spacing in lambda units along y*	*wavelengths*	*input*
theta0	*elevation steering angle*	*degrees*	*input*
phi0	*azimuth steering angle*	*degrees*	*input*
winid	*-1: No weighting is used* *1: Use weighting defined in win*	*none*	*input*
win	*window for sidelobe control*	*none*	*input*
nbits	*negative #: perfect quantization* *positive #: use* 2^{nbits} *quantization levels*	*none*	*input*
pattern	*gain pattern*	*dB*	*output*

A MATLAB-based GUI workspace called *"array.m"* was developed for this function. It is shown in Fig. 15.36. The user is advised to use this MATLAB GUI workspace to generate array gain patterns that match this requirement. Figures 15.37 through 15.42, respectively, show plots of the array gain pattern in the *U-V* space, for the following cases:

Case I: [pattern] = rect_array(15, 15, 0.5, 0.5, 0, 0, -1, -1, -3)

Case II: [pattern] = rect_array(15, 15, 0.5, 0.5, 20, 30, -1, -1, -3)

Case III: [pattern] = rect_array(15, 15, 0.5, 0.5, 45, 45, 1, 'Hamming', -3)

Case IV: [pattern] = rect_array(15, 15, 0.5, 0.5, 10, 20, -1, -1, 3)

Case V: [pattern] = rect_array(15, 15, 1, 0.5, 20, 25, -1, -1, -3)

Case VI: [pattern] = rect_array(15, 15, 1.25, 1.25, 0, 0, -1, -1, -3)

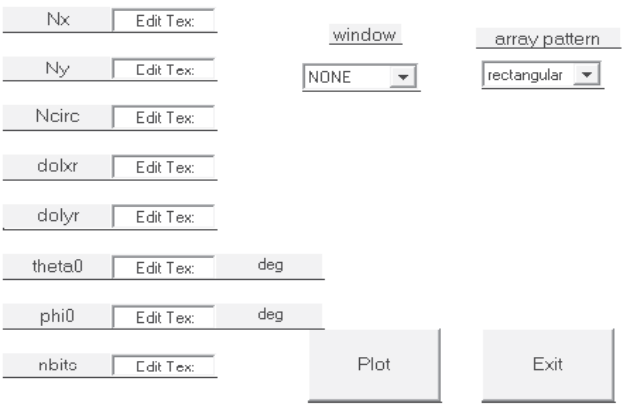


Figure 15.36. MATLAB GUI workspace *"array.m."*

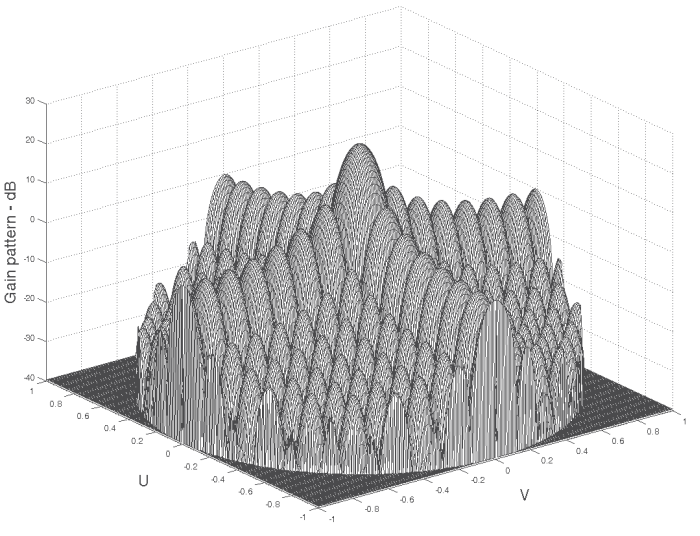


Figure 15.37a. 3-D gain pattern corresponding to Case I.

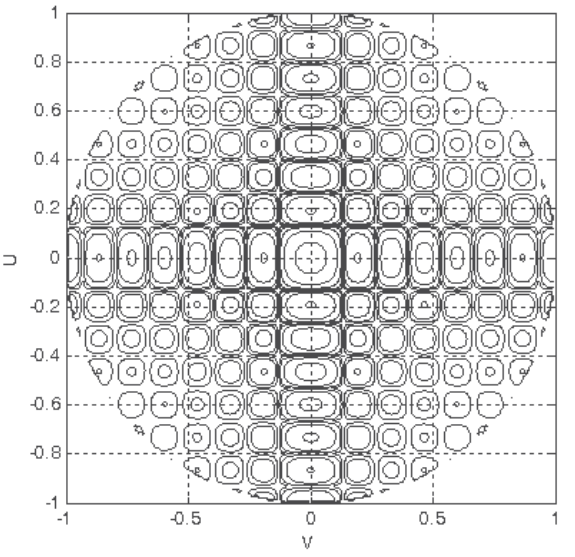


Figure 15.37b. Contour plot corresponding to Case I.

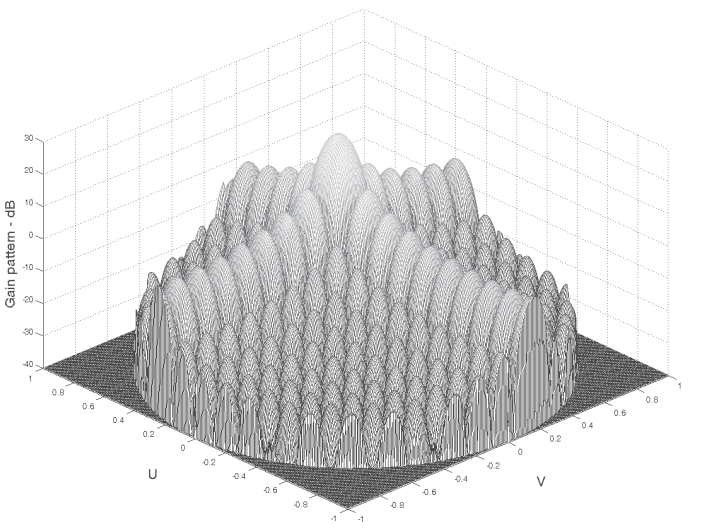


Figure 15.38a. 3-D gain pattern corresponding to Case II.

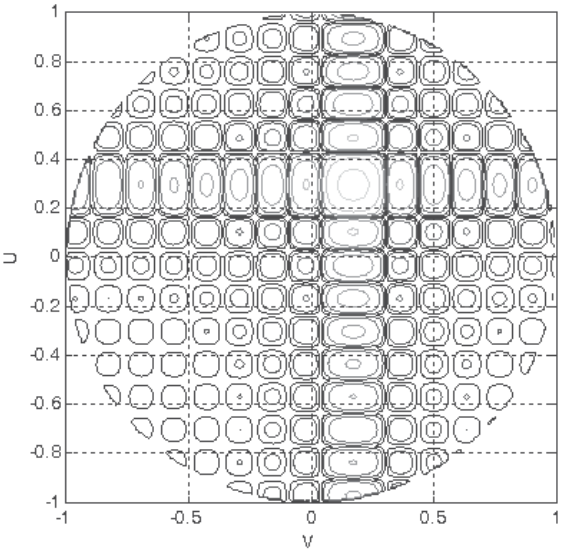


Figure 15.38b. Contour plot corresponding to Case II.

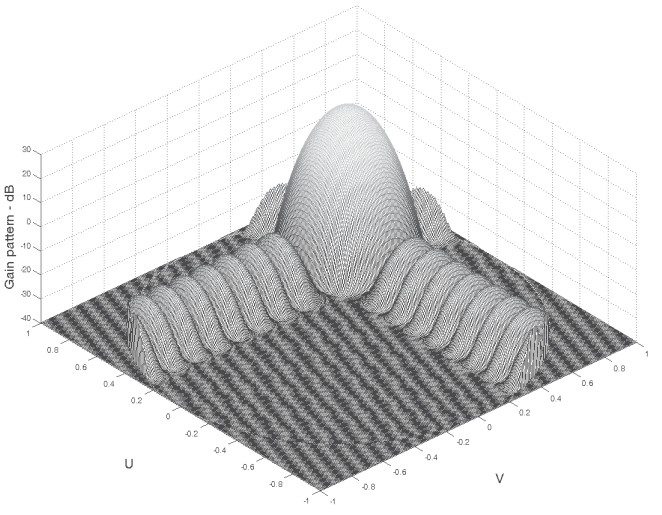


Figure 15.39a. 3-D gain pattern corresponding to Case III.

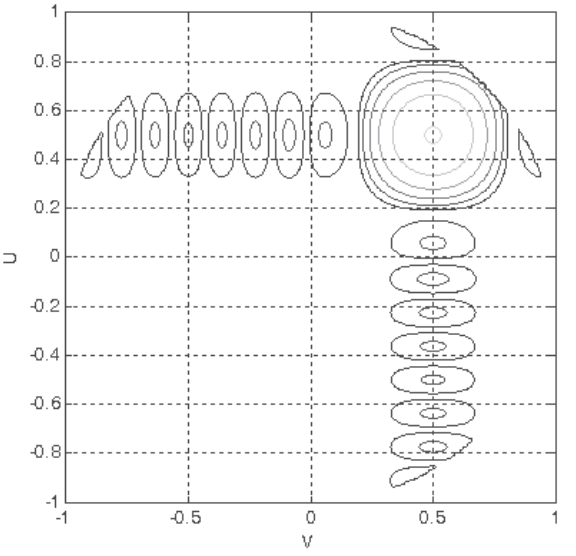


Figure 15.39b. Contour plot corresponding to Case III.

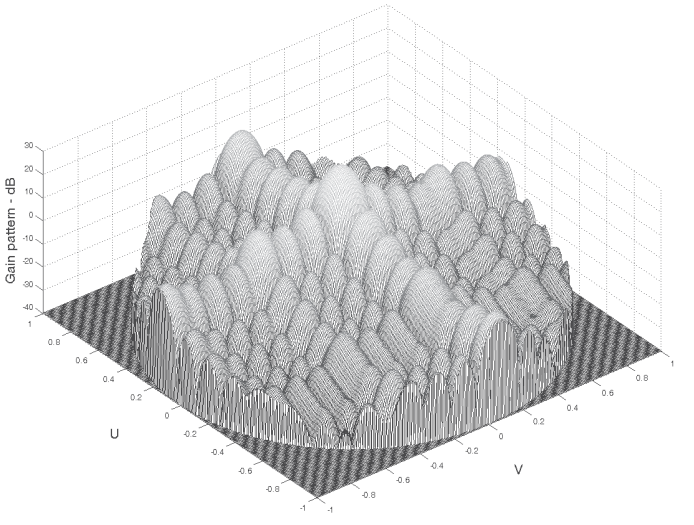


Figure 15.40a. 3-D gain pattern corresponding to Case IV.

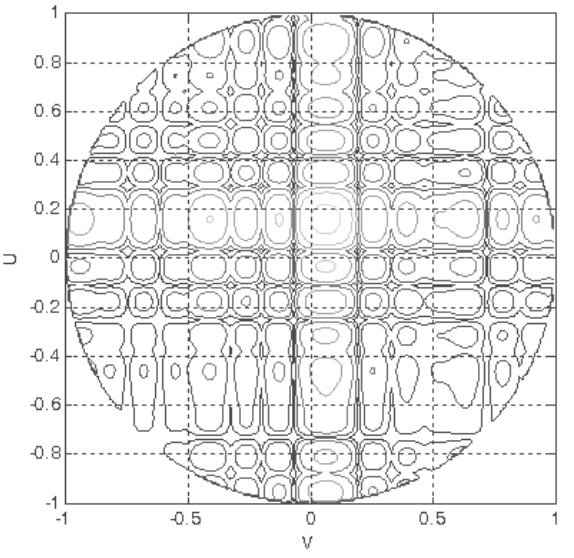


Figure 15.40b. Contour plot corresponding to Case IV.

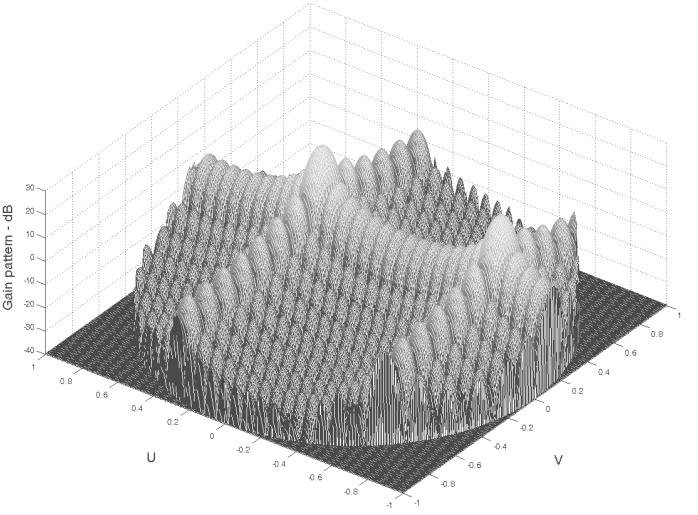


Figure 15.41a. 3-D gain pattern corresponding to Case V.

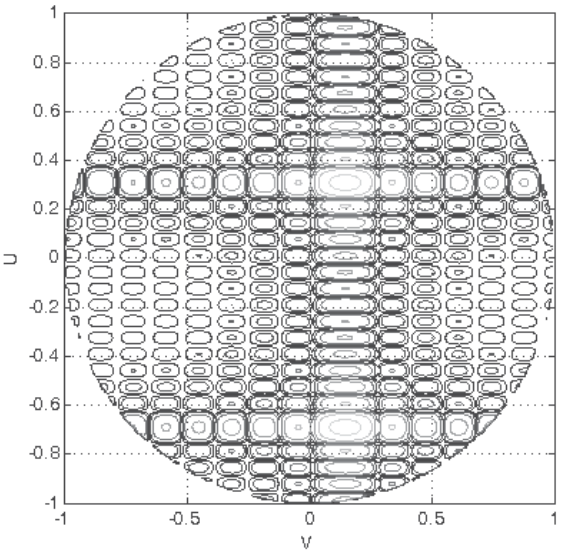


Figure 15.41b. Contour plot corresponding to Case V.

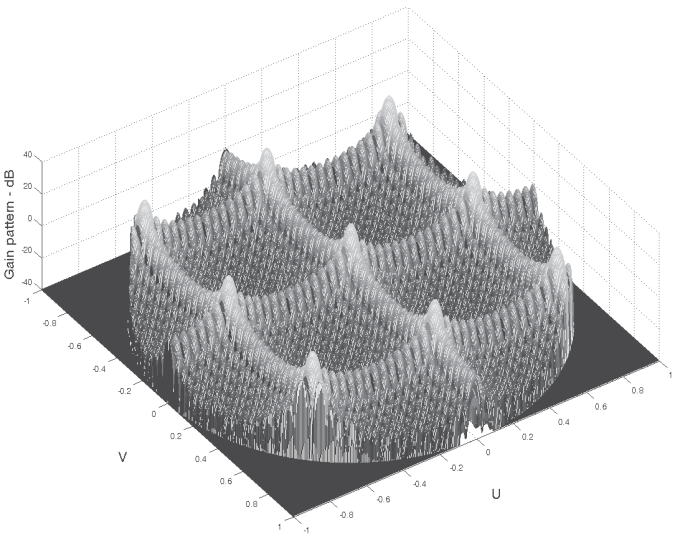


Figure 15.42a. 3-D gain pattern corresponding to Case VI.

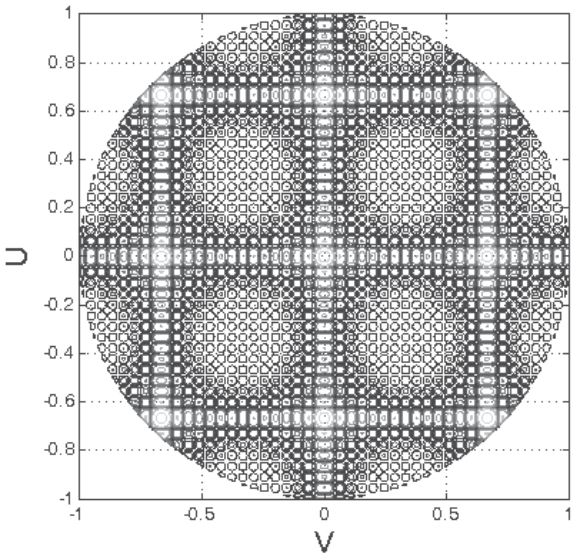


Figure 15.42b. Contour plot corresponding to Case VI.

MATLAB Function "circ_array.m"

The function *"circ_array.m"* computes and plots the rectangular grid with a circular array boundary antenna gain pattern in the visible *U,V* space. The syntax is as follows:

[pattern, amn] = circ_array(N, dolxr, dolyr, theta0, phi0, winid, win, nbits);

where

Symbol	Description	Units	Status
N	*number of elements along diameter*	*none*	*input*
dolxr	*element spacing in lambda units along x*	*wavelengths*	*input*
dolyr	*element spacing in lambda units along y*	*wavelengths*	*input*
theta0	*elevation steering angle*	*degrees*	*input*
phi0	*azimuth steering angle*	*degrees*	*input*
winid	*-1: No weighting is used* *1: Use weighting defined in win*	*none*	*input*
win	*window for sidelobe control*	*none*	*input*
nbits	*negative #: perfect quantization* *positive #: use* 2^{nbits} *quantization levels*	*none*	*input*

Symbol	Description	Units	Status
patterng	*gain pattern*	*dB*	*output*
amn	*a(m,n) sequence defined in Eq. (15.68)*	*none*	*output*

Figures 15.43 through 15.48, respectively, show plots of the array gain pattern versus steering for the following cases:

Case I: [pattern, amn] = circ_array(15, 0.5, 0.5, 0, 0, -1, -1, -3)

Case II: [pattern, amn] = circ_array(15, 0.5, 0.5, 20, 30, -1, -1, -3)

Case III: [pattern, amn] = circ_array(15, 0.5, 0.5, 30, 30, 1, 'Hamming', -3)

Case IV: [pattern, amn] = circ_array(15, 0.5, 0.5, 30, 30, -1, -1, 3)

Case V: [pattern, amn] = circ_array(15, 1, 0.5, 30, 30, -1, -1, -3)

Case VI: [pattern, amn] = circ_array(15, 1, 1, 0, 0, -1, -1, -3)

Note that the function *"circ_array.m"* uses the function *"rec_to_circ.m,"* which computes the array $a(m,n)$. Also note that the GUI workspace shown in Fig. 15.36 can be used in this case by applying the *"Ncirc"* option on the GUI, where *"Ncirc"* refers to the number of array elements along the diameter.

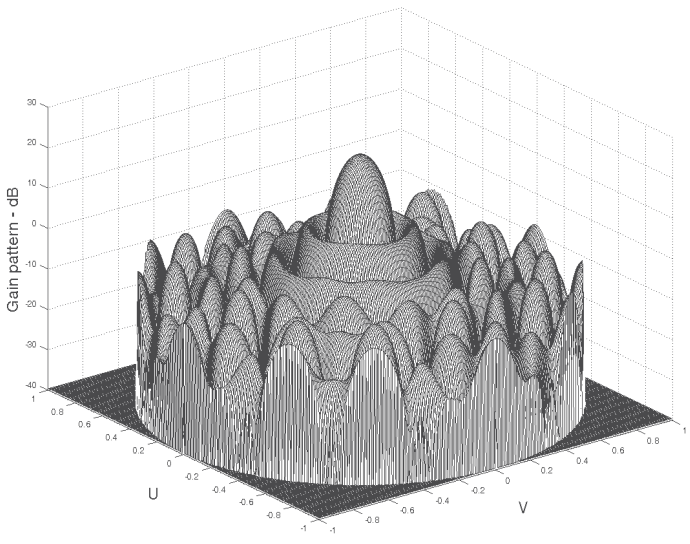


Figure 15.43a. 3-D gain pattern corresponding to Case I.

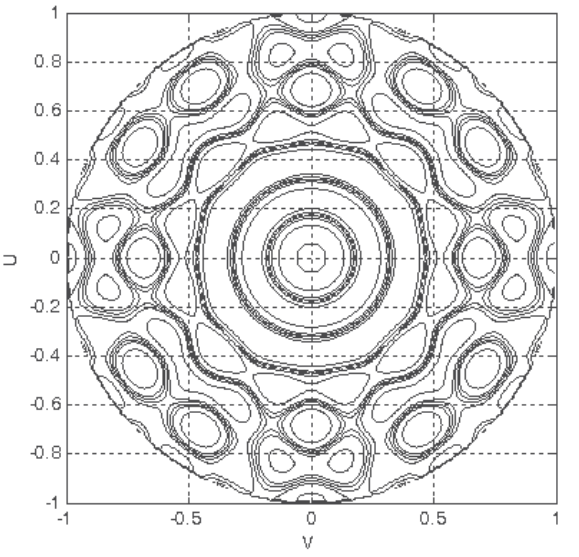


Figure 15.43b. Contour plot corresponding to Case I.

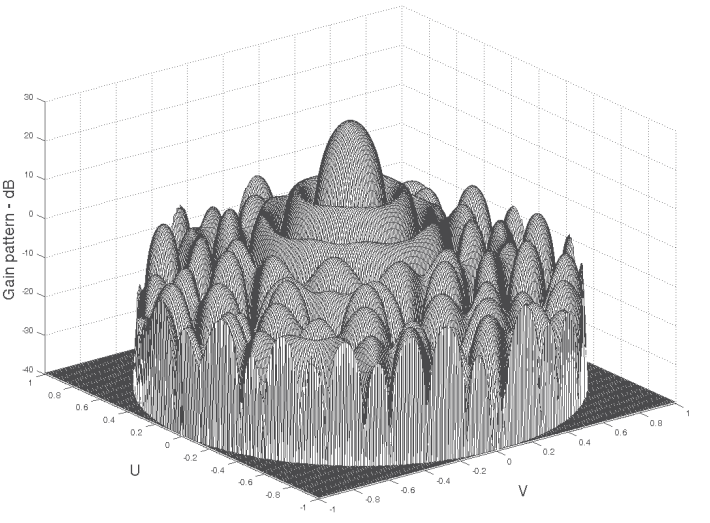


Figure 15.44a. 3-D gain pattern corresponding to Case II.

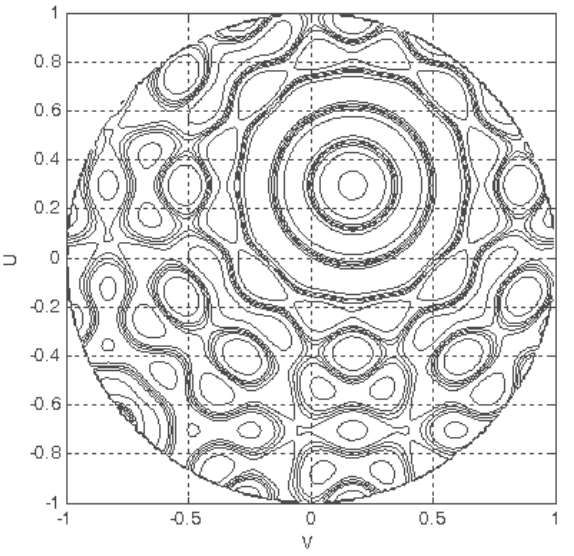


Figure 15.44b. Contour plot corresponding to Case II.

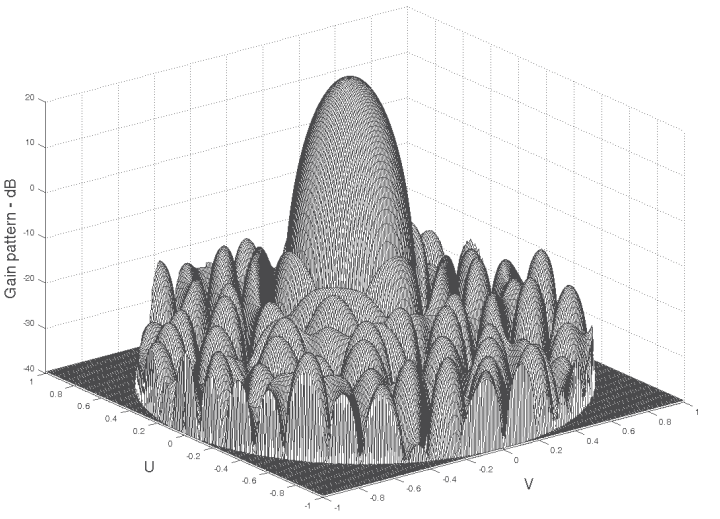


Figure 15.45a. 3-D gain pattern corresponding to Case III.

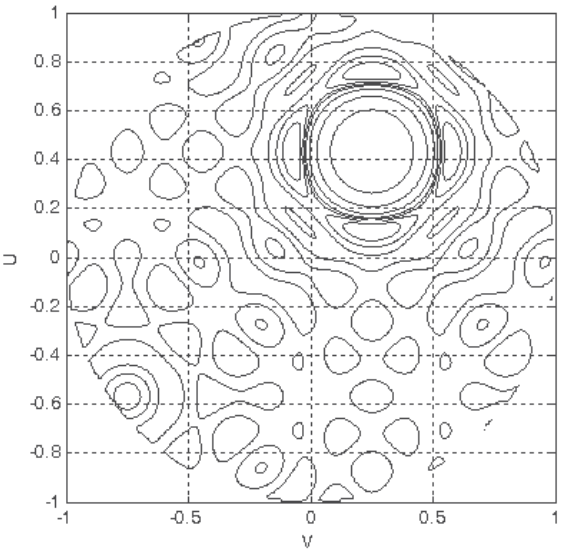


Figure 15.45b. Contour plot corresponding to Case III.

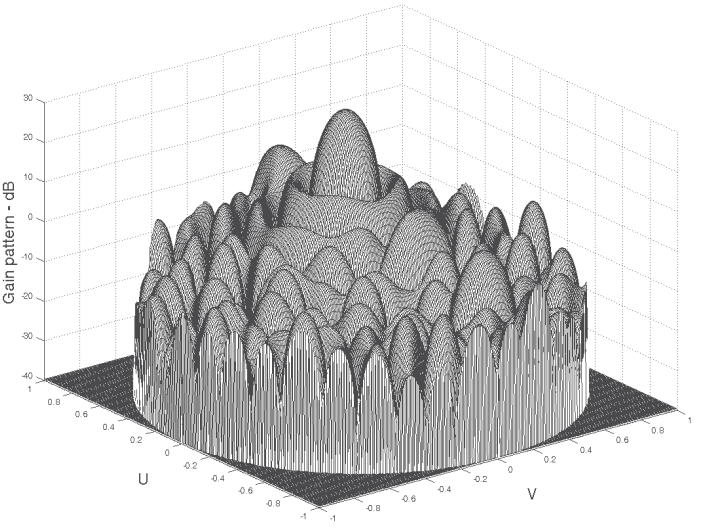


Figure 15.46a. 3-D gain pattern corresponding to Case IV.

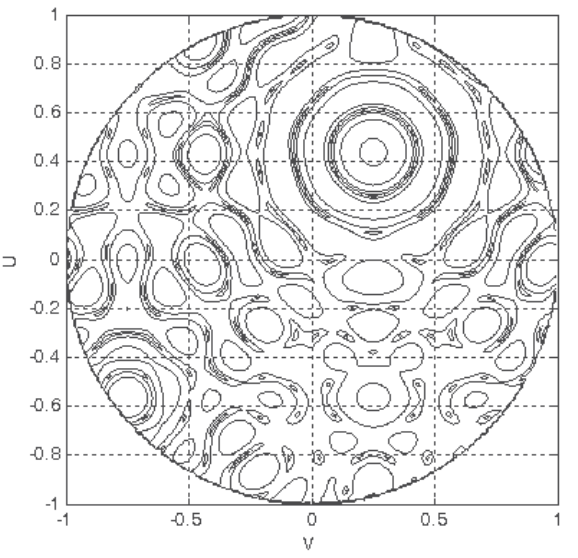


Figure 15.46b. Contour plot corresponding to Case IV.

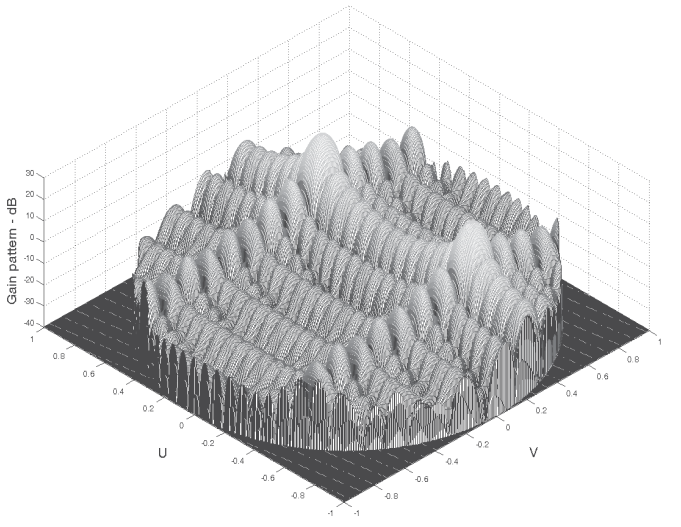


Figure 15.47a. 3-D gain pattern corresponding to Case V.

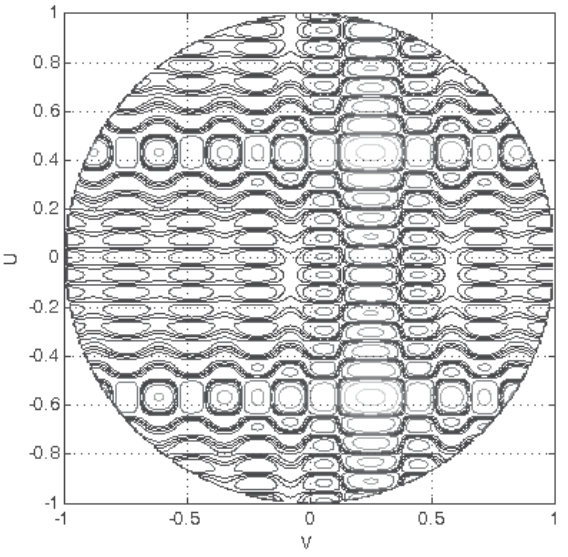


Figure 15.467b. Contour plot corresponding to Case V.

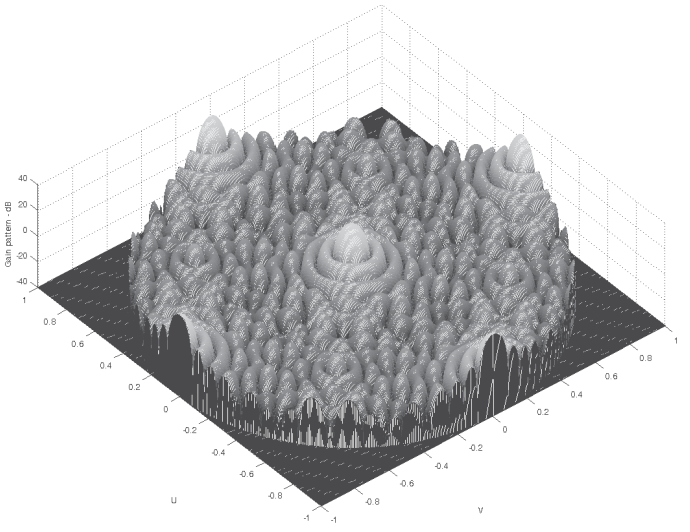


Figure 15.48a. 3-D gain pattern corresponding to Case VI.

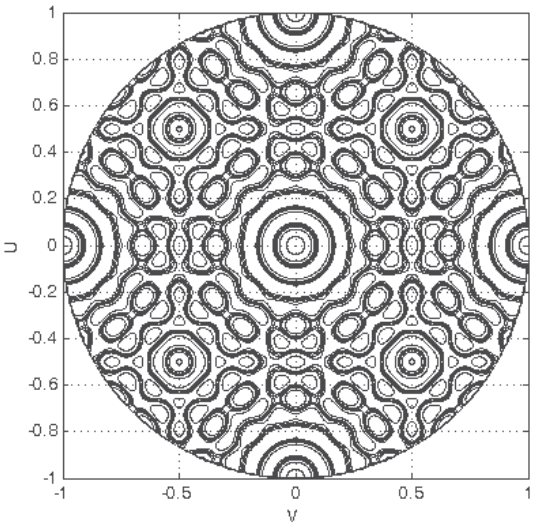


Figure 15.48b. Contour plot corresponding to Case VI.

The program *"array.m"* also plots the array's element spacing pattern. Figures 15.49a and 15.49b show two examples. The ***"x's"*** indicate the location of actual active array elements, while the ***"o's"*** indicate the location of dummy or virtual elements created merely for computational purposes. More precisely, Figure 15.49a shows a rectangular grid with circular boundary as defined in Eqs. (15.67) and (15.68) with $d_x = d_y = 0.5\lambda$ and $a = 0.35\lambda$. Figure 15.49b is similar, except for an element spacing $d_x = 1.5\lambda$ and $d_y = 0.5\lambda$.

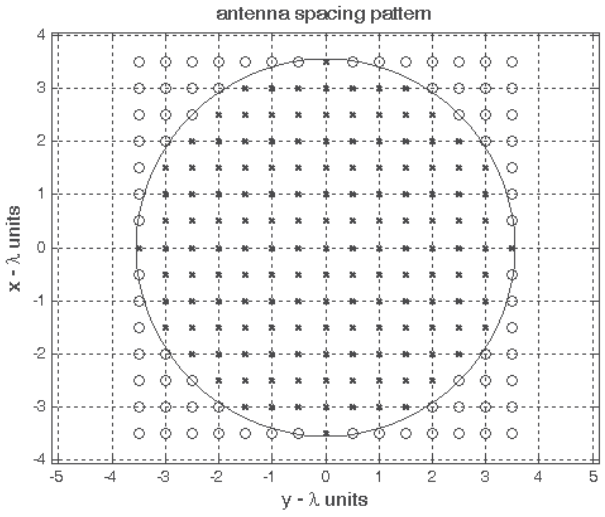


Figure 15.49a. A 15-element circular array made from a rectangular array with circular boundary. Element spacing $d_x = 0.5\lambda = d_y$.

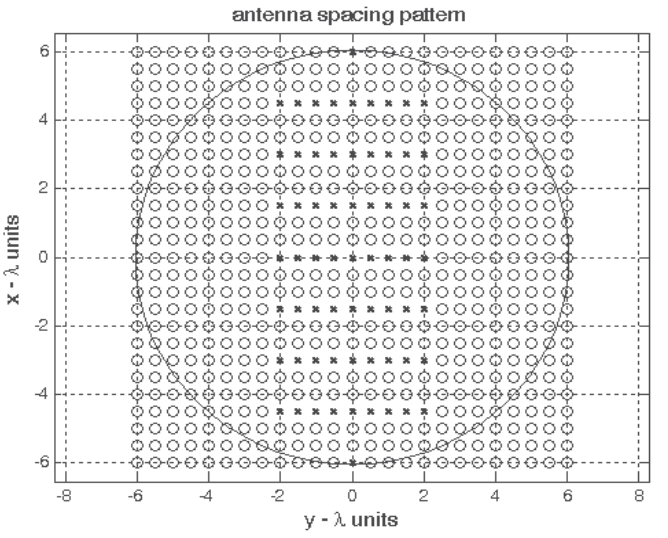


Figure 15.49b. A 15-element circular array made from a rectangular array with circular boundary. Element spacing $d_y = 0.5\lambda$ and $d_x = 1.5\lambda$.

15.6. Array Scan Loss

Phased arrays experience gain loss when the beam is steered away from the array boresight, or zenith (normal to the face of the array). This loss is due to the fact that the array effective aperture becomes smaller, and consequently the array beamwidth is broadened, as illustrated in Fig. 15.50. This loss in antenna gain is called scan loss, L_{scan}, where

$$L_{scan} = \left(\frac{A}{A_\theta}\right)^2 = \left(\frac{G}{G_\theta}\right)^2. \qquad \textbf{Eq. (15.74)}$$

A_θ is the effective aperture area at scan angle θ, and G_θ is the effective array gain at the same angle.

The beamwidth at scan angle θ is

$$\Theta_\theta = \frac{\Theta_{broadside}}{\cos\theta} \qquad \textbf{Eq. (15.75)}$$

due to the increased scan loss at large scanning angles. In order to limit the scan loss to under some acceptable practical values, most arrays do not scan electronically beyond about $\theta = 60°$. Such arrays are called Full Field of View (FFOV) arrays. FFOV arrays employ element spacing of 0.6λ or less to avoid grating lobes. FFOV array scan loss is approximated by

$$L_{scan} \approx (\cos\theta)^{2.5}. \qquad \textbf{Eq. (15.76)}$$

Arrays that limit electronic scanning to under $\theta = 60°$ are referred to as Limited Field of View (LFOV) arrays. In this case the scan loss is

$$L_{scan} = \left[\frac{\sin\left(\frac{\pi d}{\lambda}\sin\theta\right)}{\frac{\pi d}{\lambda}\sin\theta} \right]^{-4} . \qquad \textbf{Eq. (15.77)}$$

Figure 15.51 shows a plot for scan loss versus scan angle. This figure can be reproduced using MATLAB program *"Fig15_50.m,"* listed in Appendix 15-A.

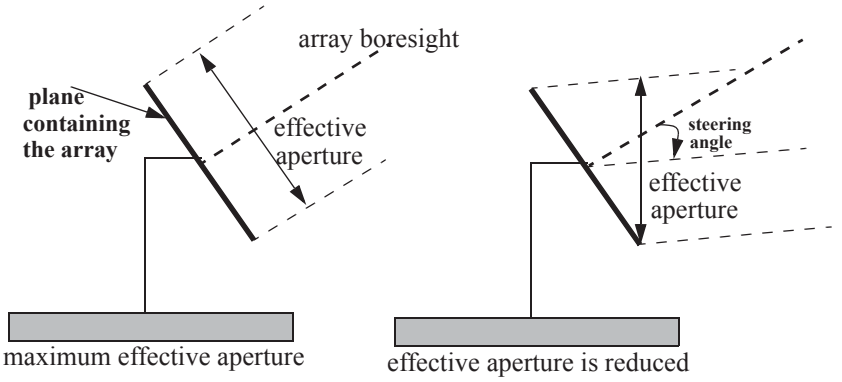


Figure 15.50. Reduction in array effective aperture due to electronic scanning.

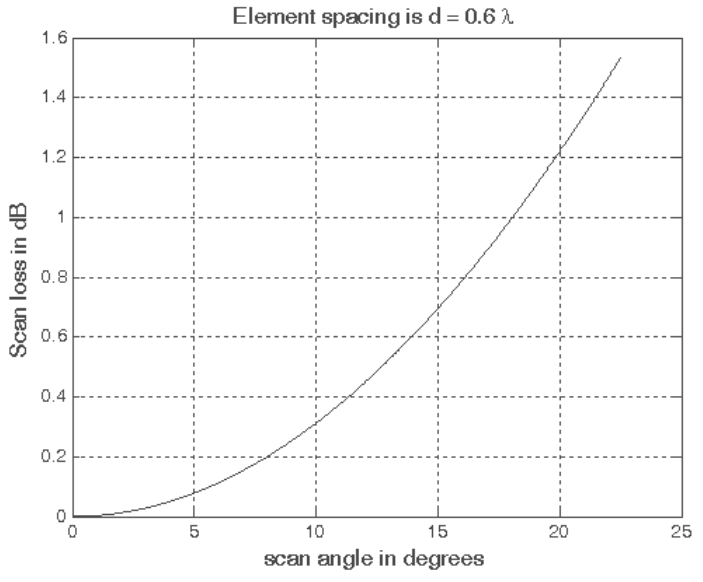


Figure 15.51. Scan loss versus scan angle, based on Eq. (15.77).

15.7. Multiple Input Multiple Output (MIMO) - Linear Array

In this section, a multiple input multiple output (MIMO) target detection technique is presented. This section is based on Mahafza[1] et al (1996). In this approach, each array element (or subarray; super-element) has its own receive channel as described in Fig. 15.52. The radar is assumed to transmit a burst of N pulses, where N is equal to the number of elements in the array. The phase reference of transmitted pulses is assumed to move linearly from the first element in the array when the first pulse in the burst is transmitted, to the last element in the array for the last pulse in the burst. In this fashion, a total of N^2 complex returns are collected and stored in memory. As will be explained later, there are a total of $(2N-1)$ distinct returns of equal two-way phase. It follows that an array twice as large as the actual one is synthesized; hence the term *synthetic* in the title. This synthetic array effectively doubles the angular resolution as compared to the standard operation and the SNR is greatly improved.

Consider the array shown in Fig. 15.52. A burst of N pulses is transmitted where the phase reference for n^{th} is the physical center of element n. The echo signals are collected and stored coherently on the basis of equal two-way geometric phase. A complex information sequence $\{b(m); m = 0, 2N-2\}$ is synthesized. The two-way array pattern is computed as the amplitude spectrum of $\{b(m)\}$. The synthesized sequence has natural triangular windowing, and the sidelobes are about -27dB, thus extra tapering may not be required.

For each transmitted pulse there are a total of N echo signals. The two-way phase corresponding to the echo signal for the i^{th} element transmitting and the j^{th} element receiving is computed as

$$b(n) = e^{jk(\mathbf{r}_i \cdot \mathbf{r}_0 + \mathbf{r}_j \cdot \mathbf{r}_0)} = e^{j\phi_i} e^{j\phi_j} \qquad \text{Eq. (15.78)}$$

$$m = i + j;\ (i,j) = 0, N-1 \qquad \text{Eq. (15.79)}$$

$$b(m) = p(i,j)\exp(j\phi_i)\exp(j\phi_j) \qquad \text{Eq. (15.80)}$$

$$\phi_i = \left(-\left(\frac{N-1}{2}\right) + i\right)\Delta\theta \qquad \text{Eq. (15.81)}$$

$$\phi_j = \left(-\left(\frac{N-1}{2}\right) + j\right)\Delta\theta \qquad \text{Eq. (15.82)}$$

$$\Delta\theta = \frac{2\pi d}{\lambda}\sin\theta \qquad \text{Eq. (15.83)}$$

where $\mathbf{r}_i \cdot \mathbf{r}_0$ and $\mathbf{r}_j \cdot \mathbf{r}_0$ are the dot products between the vectors $\mathbf{r}_0$ and $\mathbf{r}_i, \mathbf{r}_j$, respectively. The vector $\mathbf{r}_0$ is the ray between the phase reference point of the array, taken as the physical center of the array, in this analysis, and the far field target; the vectors $\mathbf{r}_i, \mathbf{r}_j$ are the rays between i^{th}, j^{th} elements of the array and the far field target, respectively. Note that $p(i,j) = 1$ if the path: i^{th} element transmitting and j^{th} element receiving exists, otherwise it is equal to zero, and $\sin\theta$ is the direction-sine toward which the radiation pattern is steered. The information sequence has $N_a = 2N-1$ distinct entries. The components of the information sequence have linear phase, and the phase increment between any two adjacent terms is equal to $\Delta\phi$. The sequence $\{b(m)\}$ also has triangular shape weighting, defined by

1. Mahafza, B. R., Heifner, L.A., and Gracchi, V. C., Multitarget Detection Using Synthetic Sampled Aperture Radars (SSAMAR), *IEEE - AES Trans.*, Vol. 31, No. 3, July 1995, pp. 1127-1132.

$$\{c(m);\ m = 0, 2N-2\} = \begin{Bmatrix} m+1;\ m = 0, N-2 \\ N;\ m = N-1 \\ 2N-1-m;\ m = N, 2N-2 \end{Bmatrix}. \qquad \textbf{Eq. (15.84)}$$

Through zero padding, the sequence $\{b(m)\}$ is extended to the next power of 2. The two-way pattern in the direction $\sin\theta$, is computed as the modulus of the Discrete Fourier Transform (DFT) of the extended sequence $\{b(m)\}$.

Assume an incident plane wave defined by amplitude A_1 and direction-sine $\sin\theta_1$, and zero-mean, white additive noise w_n with variance σ^2. Then, the m^{th} sample of the information sequence is

$$b(m) = A_1 s(m) + w_n(m);\ m = 0, N_a - 1 \qquad \textbf{Eq. (15.85)}$$

$$A_1 = G_e^2(\sin\theta_1)\left(\frac{R_o}{R}\right)^4 \rho_1 \qquad \textbf{Eq. (15.86)}$$

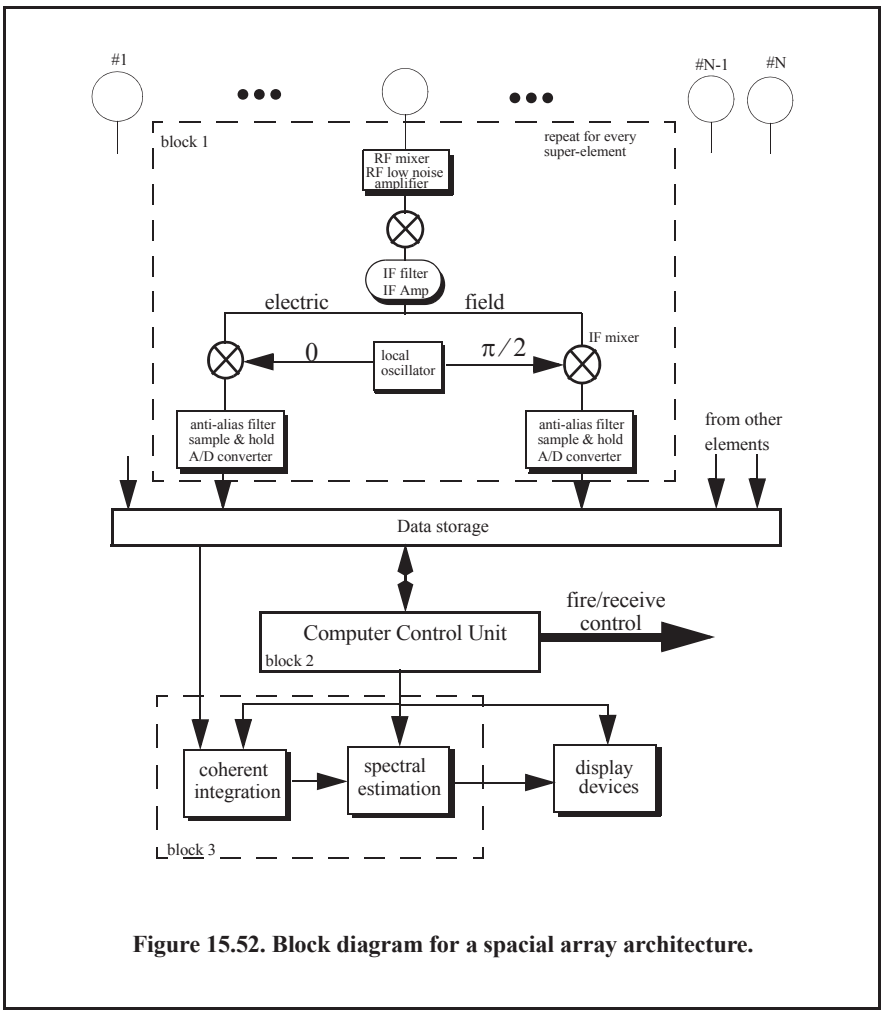


Figure 15.52. Block diagram for a spacial array architecture.

$$s(m) = c(m)\exp\{j[m-(N-1)]k_1\} \quad \text{Eq. (15.87)}$$

where, G_e^2 represents the two-way element gain, R_0 is the reference range, and ρ_1 is the wave amplitude. It follows that if there are L incident plane waves defined by $\{\sin\theta_i;\ i=1,L\}$, then the composite information sequence is

$$b(m) = \sum_{i=1}^{L} A_i s_i(m) + w_n(m);\ m = 0, N_a - 1, \quad \text{Eq. (15.88)}$$

which can be written in vector notation as

$$\mathbf{b} = \sum_{i=1}^{L} A_i \mathbf{s}_i + \mathbf{w}_n. \quad \text{Eq. (15.89)}$$

Assuming that the noise is spatially incoherent and is uncorrelated with the signal samples, the autocorrelation matrix for the field sensed by the array is

$$\Re = E[\mathbf{b}\mathbf{b}^\dagger] = \sigma^2\mathbf{I} + \sum_{i=1}^{L} P_i \mathbf{s}\mathbf{s}^\dagger \quad \text{Eq. (15.90)}$$

where $\mathbf{I}$ is the identity matrix. The discrete Fourier transformation of the sequence $\{b(m)\}$ yields,

$$B(q) = \sum_{m=0}^{N_a-1} b(m)\exp\left(-j\frac{2\pi qm}{N_a}\right);\ q = 0, N_a - 1, \quad \text{Eq. (15.91)}$$

which can be expressed as the dot product

$$B(q) = \mathbf{a}^\dagger(q)\cdot\mathbf{b} \quad \text{Eq. (15.92)}$$

where

$$a(q;m) = \exp\left(j\frac{2\pi qm}{N_a}\right). \quad \text{Eq. (15.93)}$$

The power at the output of the signal processor at frequency bin q is

$$P(q) = E[|B(q)|^2] = E[\{\mathbf{a}^\dagger(q)\mathbf{b}\}\{\mathbf{a}^\dagger(q)\mathbf{b}\}^\dagger] = \mathbf{a}^\dagger(q)\Re\mathbf{a}(q) \quad \text{Eq. (15.94)}$$

where $\Re$ is defined in Eq. (15.90). Thus,

$$P(q) = \sigma_\upsilon^2\mathbf{a}^\dagger(q)\mathbf{I}\mathbf{a}(q) + \sum_{i=1}^{L} P_1\mathbf{a}^\dagger(q)\mathbf{s}_i\mathbf{s}_i^\dagger\mathbf{a}(q). \quad \text{Eq. (15.95)}$$

After compensation for range attenuation and antenna gain, spectral peaks will be proportional to amplitudes of incident waves. For example, a peak at an arbitrary bin q_j will correspond to a plane wave defined by direction-sine $\sin\theta_j$. It follows that

$$P(q_j) = 2N\sigma^2 + N^4 P_j + \sum_{\substack{i=1 \\ i \neq j}}^{L} P_i \mathbf{a}^{\dagger}(q_j)\mathbf{s}_i\mathbf{s}_i^{\dagger}\mathbf{a}(q_j). \qquad \textbf{Eq. (15.96)}$$

The first term of the right-hand side of Eq. (15.96) represents the noise power at q_j. The last term corresponds to spectral leakage, while the signal power at q_j is given by the middle term. Note that the sequence $\{c(m)\}$ is the reason for the N^4 factor. More precisely,

$$N^4 = \sum_{m=0}^{2N-2} [c(m)]^2. \qquad \textbf{Eq. (15.97)}$$

Thus, the SNR is

$$SNR|_{q_j} = \left(\frac{N^3}{2}\right)\left(\frac{P_j}{\sigma_{\upsilon}^2}\right). \qquad \textbf{Eq. (15.98)}$$

Recall that in conventional phased array radars the SNR is improved by a factor of *N*, where *N* is the size of the array. Examination of Eq. (15.98) indicates that the SNR improvement factor using this MIMO mode of operation over conventional array operation and signal processing is

$$I_{SNR} = (20\log N - 3)dB. \qquad \textbf{Eq. (15.99)}$$

Problems

15.1. Consider an antenna whose diameter is $d = 3m$. What is the far field requirement for an X-band or an L-band radar that is using this antenna?

15.2. Consider an antenna with electric field intensity in the *xy*-plane $E(\varsigma)$. This electric field is generated by a current distribution $D(y)$ in the *yz*-plane. The electric field intensity is computed using the integral

$$E(\varsigma) = \int_{-r/2}^{r/2} D(y)\exp\left(2\pi j\frac{y}{\lambda}\sin\varsigma\right)dy$$

where λ is the wavelength and r is the aperture. (a) Write an expression for $E(\varsigma)$ when $D(Y) = d_0$ (a constant). (b) Write an expression for the normalized power radiation pattern and plot it in dB.

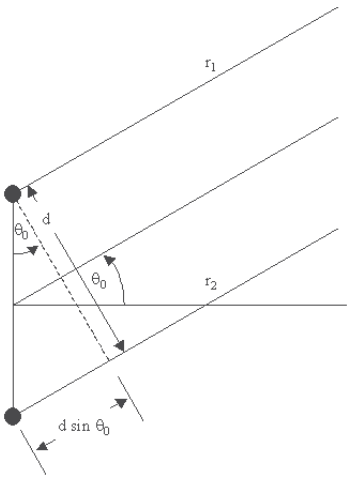
15.3. A linear phased array consists of 50 elements with $\lambda/2$ element spacing. (a) Compute the $3dB$ beamwidth when the main-beam steering angle is $0°$ and $45°$. (b) Compute the electronic phase difference for any two consecutive elements for steering angle $60°$.

15.4. A linear phased array antenna consists of eight elements spaced with $d = \lambda$ element spacing. (a) Give an expression for the antenna gain pattern (assume no steering and uniform aperture weighting). (b) Sketch the gain pattern versus the sine of the off-boresight angle β. What problems do you see is using $d = \lambda$ rather than $d = \lambda/2$?

15.5. In Section 15.4.2 we showed how a DFT can be used to compute the radiation pattern of a linear phased array. Consider a linear phased array of 64 elements at half wavelength spacing, where an FFT of size 512 is used to compute the pattern. What are the FFT bins that correspond to steering angles $\beta = 30°, 45°$?

15.6. Derive Eq. (15.73).

15.7. Consider the two-element array shown in the figure below. If the composite array electric field is $E(\theta) = a_1 E_1(\theta) + a_2 E_2(\theta)$, where a_1, a_2 are constants (can be complex) and E_1, E_2 are the individual elements fields. Determine a_1, a_2 so that the electric field is maximum at θ_o. Plot the resulting array pattern.



15.8. Use the FFT to compute the radiation pattern for an array of size 21 elements and element spacing (a) $d = 0.5\lambda$, and (b) $d = 0.8\lambda$. In each case, compute and plot the array pattern with and without using Hamming weights.

15.9. Modify the FFT routine developed in the previous problem to compute and plot the power gain pattern.

15.10. Repeat Problems 15.8 and 15.9, where in this case the array pattern can be steered in any off-boresight direction.

15.11. Why do the grating lobes appear when the array beam is steered to angles other than the boresight? Include reasonable plots to back up your argument.

Appendix 15-A: Chapter 15 MATLAB Code Listings

The MATLAB code provided in this chapter was designed as an academic standalone tool and is not adequate for other purposes. The code was written in a way to assist the reader in gaining a better understanding of the theory. The code was not developed, nor is it intended to be used as part of an open-loop or a closed-loop simulation of any kind. The MATLAB code found in this textbook can be downloaded from this book's web page on the CRC Press website. Simply use your favorite web browser, go to www.crcpress.com, and search for keyword *"Mahafza"* to locate this book's web page.

MATLAB Program "Fig15_5.m" Listing

```
% Use this code to produce Fig. 15.5a and 15.5b based on Eq.(15.35)
clc
clear all
close all
eps = 0.00001;
k = 2*pi;
theta = -pi : pi / 10791 : pi;
var = sin(theta);
nelements = 8;
d = 1;        %  d = 1;
num = sin((nelements * k * d * 0.5) .* var);
%
if(abs(num) <= eps)
  num = eps;
end
den = sin((k* d * 0.5) .* var);
if(abs(den) <= eps)
  den = eps;
end
%
pattern = abs(num ./ den);
maxval = max(pattern);
pattern = pattern ./ maxval;
 %
figure(1)
plot(var,pattern,'linewidth', 1.5)
xlabel('\bfsine angle - dimensionless')
ylabel('\bfArray pattern')
grid
%
figure(2)
plot(var,20*log10(pattern),'linewidth', 1.5)
axis ([-1 1 -60 0])
xlabel('\bfsine angle - dimensionless')
ylabel('\bfPower pattern [dB]')
grid;
%
figure(3)
theta = theta +pi/2;
polar(theta,pattern)
title ('\bfArray pattern')
```

MATLAB Program "Fig15_7.m" Listing

```
% Use this program to reproduce Fig. 15.7 of text
clc
clear all
close all
eps =0.00001;
N = 32;
rect(1:32) = 1;
ham = hamming(32);
han = hanning(32);
blk = blackman(32);
k3 = kaiser(32,3);
k6 = kaiser(32,6);
RECT = 20*log10(abs(fftshift(fft(rect, 1024)))./32 +eps);
HAM =  20*log10(abs(fftshift(fft(ham, 1024)))./32 +eps);
HAN =  20*log10(abs(fftshift(fft(han, 1024)))./32+eps);
BLK = 20*log10(abs(fftshift(fft(blk, 1024)))./32+eps);
K6 = 20*log10(abs(fftshift(fft(k6, 1024)))./32+eps);
x = linspace(-1,1,1024);
figure
subplot(2,1,1)
plot(x,RECT,'k',x,HAM,'k--',x,HAN,'k-.','linewidth',1.5);
xlabel('x')
ylabel('Window')
grid
axis tight
legend('Rectangular','Hamming','Hanning')
subplot(2,1,2)
plot(x,RECT,'k',x,BLK,'k--',x,K6,'K-.','linewidth',1.5)
xlabel('x')
ylabel('Window')
legend('Rectangular','Blackman','Kasier at \beta = 6')
grid
axis tight
```

MATLAB Function "linear_array.m" Listing

```
function [theta,patternr,patterng] = linear_array(Nr,dolr,theta0,winid,win,nbits);
% This function computes and returns the gain radiation pattern for a linear array
% It uses the FFT to computes the pattern
%%%% *INPUTS ********** %%%%%%%%%%%%%%%%%%%%
% Nr ==> number of elements; dolr ==> element spacing (d) in lambda units divided by lambda
% theta0 ==> steering angle in degrees; winid ==> use winid negative for no window, winid positive to
enter your window of size(Nr)
% win is input window, NOTE that win must be an NrX1 row vector; nbits ==> number of bits used in the
pahse shifters
% negative nbits mean no quantization is used
%%%% *OUTPUTS ********** %%%%%%%%%%%%%%%%%%%%
% theta ==> real-space angle; patternr ==> array radiation pattern in dBs
% patterng ==> array directive gain pattern in dBs
%%%%%%%% ********************** %%%%%%%%%%%%%%%%%%%%
eps = 0.00001;
n = 0:Nr-1;
i = sqrt(-1);
```

```
%if dolr is > 0.5 then; choose dol = 0.25 and compute new N
if(dolr <=0.5)
  dol = dolr;
  N = Nr;
else
  ratio = ceil(dolr/.25);
  N = Nr * ratio;
  dol = 0.25;
end
% choose proper size fft, for minimum value choose 256
Nrx = 10 * N;
nfft = 2^(ceil(log(Nrx)/log(2)));
if nfft < 256
   nfft = 256;
end
% convert steering angle into radians; and compute the sine of angle
theta0 = theta0 *pi /180.;
sintheta0 = sin(theta0);
% detrmine and comupte quantized steering angle
if nbits < 0
  phase0 = exp(i*2.0*pi .* n * dolr * sintheta0);
else
   % compute and add the phase shift terms (WITH nbits quantization)
   % Use formula thetal = (2*pi*n*dol) * sin(theta0) divided into 2^nbits
   % and rounded to the nearest qunatization level
   levels = 2^nbits;
   qlevels = 2.0 * pi / levels; % compute quantization levels
% compute the phase level and round it to the closest quantizatin level
   angleq = round(dolr .* n * sintheta0 * levels) .* qlevels; % vector of possible angles
   phase0 = exp(i*angleq);
end
% generate array of elements with or without window
if winid < 0
   wr(1:Nr) = 1;
else
   wr = win';
end
% add the phase shift terms
 wr =  wr .* phase0;
 % determine if interpolation is needed (i.e N > Nr)
if N > Nr
   w(1:N) = 0;
   w(1:ratio:N) = wr(1:Nr);
else
   w = wr;
end
% compute the sine(theta) in real space sthat correspond to the FFT index
arg = [-nfft/2:(nfft/2)-1] ./ (nfft*dol);
idx = find(abs(arg) <= 1);
sinetheta = arg(idx);
theta = asin(sinetheta);
% convert angle into degrees
theta = theta .* (180.0 / pi);
% Compute fft of w (radiation pattern)
```

```
patternv = (abs(fftshift(fft(w,nfft)))).^2;
% convert raditiona pattern to dBs
patternr = 10*log10(patternv(idx) ./Nr +  eps);
% Compute directive gain pattern
rbarr  = 0.5 *sum(patternv(idx)) ./ (nfft * dol);
patterng = 10*log10(patternv(idx) + eps) - 10*log10(rbarr + eps);
return
```

MATLAB Program "circular_array.m" Listing

```
%Circular Array in the x-y plane
% Element is a short dipole antenna parallel to the z axis
% 2D Radiation Patterns for fixed phi or fixed theta
% dB polar plots uses the polardb.m file
%
%%%% Element expression needs to be modified if different
%%%% than a short dipole antenna along the z axis
%
clear all
clf
close all
%
%  ====  Input Parameters  ====
a = 1.;         % radius of the circle
N = 10;          % number of Elements of the circular array
theta0 = 45;    % main beam Theta direction
phi0 = 60;      % main beam Phi direction
% Theta or Phi variations for the calculations of the far field pattern
Variations = 'Phi';  % Correct selections are  'Theta' or 'Phi'
phid = 60;       % constant phi plane for theta variations
thetad = 45;     % constant theta plane for phi variations
%  ====  End of Input parameters section  ====
%
dtr = pi/180;          % conversion factors
rtd = 180/pi;
phi0r = phi0*dtr;
theta0r = theta0*dtr;
lambda = 1;
k = 2*pi/lambda;
ka = k*a;              % Wavenumber times the radius
jka = j*ka;
I(1:N) = 1;            % Elements excitation Amplitude and Phase
alpha(1:N) =0;
for n = 1:N            % Element positions Uniformly distributed along the circle
   phin(n) = 2*pi*n/N;
end
%
switch Variations
case 'Theta'
   phir = phid*dtr;    % Pattern in a constant Phi plane
   i = 0;
   for theta = 0.001:1:181
      i = i+1;
      thetar(i) = theta*dtr;
```

```
      angled(i) = theta;  angler(i) = thetar(i);
      Arrayfactor(i) = 0;
      for n = 1:N
         Arrayfactor(i) = Arrayfactor(i) + I(n)*exp(j*alpha(n)) ...
                  * exp( jka*(sin(thetar(i))*cos(phir -phin(n))) ...
                      -jka*(sin(theta0r  )*cos(phi0r-phin(n)))  );
      end
      Arrayfactor(i) = abs(Arrayfactor(i));
      Element(i) = abs(sin(thetar(i)+0*dtr));  % use the abs function to avoid
   end
case 'Phi'
   thetar = thetad*dtr;  % Pattern in a constant Theta plane
   i = 0;
   for phi = 0.001:1:361
      i = i+1;
      phir(i)   = phi*dtr;
      angled(i) = phi;  angler(i) = phir(i);
      Arrayfactor(i) = 0;
      for n = 1:N
         Arrayfactor(i) = Arrayfactor(i) +  I(n)*exp(j*alpha(n)) ...
                  * exp( jka*(sin(thetar )*cos(phir(i)-phin(n))) ...
                      -jka*(sin(theta0r)*cos(phi0r  -phin(n)))  );
      end
      Arrayfactor(i) = abs(Arrayfactor(i));
      Element(i) = abs(sin(thetar+0*dtr));  % use the abs function to avoid
   end
end
angler = angled*dtr;
Element = Element/max(Element);
Array = Arrayfactor/max(Arrayfactor);
ArraydB = 20*log10(Array);
EtotalR =(Element.*Arrayfactor)/max(Element.*Arrayfactor);
%
figure(1)
plot(angled,Array,'linewidth',1.5)
ylabel('Array pattern')
grid
switch Variations
case 'Theta'
  axis ([0 180 0 1 ])
%  theta = theta +pi/2;
   xlabel('\theta - Degrees')
   title ( '\phi = 90^o plane')
case 'Phi'
axis ([0 360 0 1 ])
   xlabel('\phi - Degrees')
    title ( '\theta = 90^o plane')
end
%
figure(2)
plot(angled,ArraydB,'linewidth',1.5)
%axis ([-1 1 -60 0])
ylabel('Power pattern [dB]')
grid;
```

```
switch Variations
case 'Theta'
  axis ([0 180 -60 0 ])
   xlabel('Theta [Degrees]')
      title ( '\phi = 90^o plane')
case 'Phi'
axis ([0 360 -60 0 ])
   xlabel('\phi - degrees')
       title ( '\theta = 90^o plane')
end
%
figure(3)
polar(angler,Array)
title ('Array pattern')
%
figure(4)
polardb(angler,Array)
title ('Power pattern [dB]')
 %
% the plots provided above are for the array factor based on the circular
% array plots for other patterns such as those for the antenna element
% (Element)or the total pattern (Etotal based on Element*Arrayfactor) can
% also be displayed by the user as all these patterns are already computed
% above.
 %
figure(10)
subplot(2,2,1)
polardb (angler,Element,'b-'); % rectangular plot of element pattern
title('Element normalized E field [dB]')
subplot(2,2,2)
polardb(angler,Array,'b-')
title(' Array Factor normalized [dB]')
subplot(2,2,3)
polardb(angler,EtotalR,'b-');  % polar plot
title('Total normalized E field [dB]')
```

MATLAB Function "rect_array.m" Listing

```
function [pattern] = rect_array(Nxr,Nyr,dolxr,dolyr,theta0,phi0,winid,win,nbits);
%%%%% ********************** %%%%%%%%%%%%%%
% This function computes the 3-D directive gain patterns for a planar array
% This function uses the fft2 to compute its output
%%%%%%%%%************  INPUTS ************ %%%%%%%%%%%%
% Nxr ==> number of along x-aixs; Nyr ==> number of elemnts along y-axis
% dolxr ==> element spacing in x-direction; dolyr ==> element spacing in y-direction Both are in
lambda units
% theta0 ==> elevation steering angle in degrees, phi0 ==> azimuth steering angle in degrees
% winid ==> window identifier; winid negative ==> no window ; winid positive ==> use window given
by win
% win ==> input window function (2-D window) MUST be of size (Nxr X Nyr)
% nbits is the number of nbits used in phase quantization; nbits negative ==> NO quantization
%%%%%%%% ********** OUTPUTS ************ %%%%%%%%%%%%%
% pattern ==> directive gain pattern
%%%%%%%%%%%%%%%% ********************** %%%%%%%%%%%
```

```
eps = 0.0001;
nx = 0:Nxr-1;
ny = 0:Nyr-1;
i = sqrt(-1);
% check that window size is the same as the array size
[nw,mw] = size(win);
if winid >0
   if nw ~= Nxr
   fprintf('STOP == Window size must be the same as the array')
   return
end
if mw ~= Nyr
   fprintf('STOP == Window size must be the same as the array')
   return
end
end
%if dol is > 0.5 then; choose dol = 0.5 and compute new N
if(dolxr <=0.5)
  ratiox = 1  ;
  dolx = dolxr ;
  Nx = Nxr ;
else
  ratiox = ceil(dolxr/.5) ;
  Nx = (Nxr -1 ) * ratiox + 1 ;
  dolx = 0.5 ;
end
if(dolyr <=0.5)
  ratioy = 1  ;
  doly = dolyr ;
  Ny = Nyr ;
else
  ratioy = ceil(dolyr/.5) ;
  Ny = (Nyr -1) * ratioy + 1 ;
  doly = 0.5 ;
end
% choose proper size fft, for minimum value choose 256X256
Nrx = 10 * Nx;
Nry = 10 * Ny;
nfftx = 2^(ceil(log(Nrx)/log(2)));
nffty = 2^(ceil(log(Nry)/log(2)));
if nfftx < 256
  nfftx = 256;
end
if nffty < 256
  nffty = 256;
end
% generate array of elements with or without window
if winid < 0
  array = ones(Nxr,Nyr);
else
  array = win;
end
% convert steering angles (theta0, phi0) to radians
theta0 = theta0 * pi / 180;
```

```
phi0 = phi0 * pi / 180;
% convert steering angles (theta0, phi0) to U-V sine-space
u0 = sin(theta0) * cos(phi0);
v0 = sin(theta0) * sin(phi0);
% Use formula thetal = (2*pi*n*dol) * sin(theta0) divided into 2^m levels
% and rounded to the nearest qunatization level
if nbits < 0
   phasem = exp(i*2*pi*dolx*u0 .* nx *ratiox);
   phasen = exp(i*2*pi*doly*v0 .* ny *ratioy);
else
   levels = 2^nbits;
   qlevels = 2.0*pi / levels; % compute quantization levels
   sinthetaq = round(dolx .* nx * u0 * levels * ratiox) .* qlevels; % vector of possible angles
   sinphiq = round(doly .* ny * v0 * levels *ratioy) .* qlevels; % vector of possible angles
   phasem = exp(i*sinthetaq);
   phasen = exp(i*sinphiq);
end
 % add the phase shift terms
array = array .* (transpose(phasem) * phasen);
  % determine if interpolation is needed (i.e. N > Nr)
if (Nx > Nxr )| (Ny > Nyr)
  for xloop = 1 : Nxr
     temprow = array(xloop, :) ;
     w( (xloop-1)*ratiox+1, 1:ratioy:Ny) =  temprow ;
  end
  array = w;
else
   w = array ;
%    w(1:Nx, :) = array(1:N,:);
end
% Compute array pattern
arrayfft = abs(fftshift(fft2(w,nfftx,nffty))).^2 ;
%compute [su,sv] matrix
U = [-nfftx/2:(nfftx/2)-1] ./(dolx*nfftx);
indexx = find(abs(U) <= 1);
U = U(indexx);
V = [-nffty/2:(nffty/2)-1] ./(doly*nffty);
indexy = find(abs(V) <= 1);
V = V(indexy);
%Normalize to generate gain pattern
rbar=sum(sum(arrayfft(indexx,indexy))) / dolx/doly/4./nfftx/nffty;
arrayfft = arrayfft(indexx,indexy) ./rbar;
[SU,SV] = meshgrid(V,U);
indx = find((SU.^2 + SV.^2) >1);
arrayfft(indx) = eps/10;
pattern = 10*log10(arrayfft +eps);
figure(1)
mesh(V,U,pattern);
xlabel('V')
ylabel('U');
zlabel('Gain pattern - dB')
figure(2)
contour(V,U,pattern)
grid
```

```
axis image
xlabel('V')
ylabel('U');
axis([-1 1 -1 1])
figure(3)
x0 = (Nx+1)/2 ;
y0 = (Ny+1)/2 ;
radiusx = dolx*((Nx-1)/2) ;
radiusy = doly*((Ny-1)/2) ;
[xxx, yyy]=find(abs(array)>eps);
xxx = xxx-x0 ;
yyy = yyy-y0 ;
plot(yyy*doly, xxx*dolx,'rx')
hold on
axis([-radiusy-0.5 radiusy+0.5 -radiusx-0.5  radiusx+0.5]);
grid
title('antenna spacing pattern');
xlabel('y - \lambda units')
ylabel('x - \lambda units')
[xxx0, yyy0]=find(abs(array)<=eps);
xxx0 = xxx0-x0 ;
yyy0 = yyy0-y0 ;
plot(yyy0*doly, xxx0*dolx,'co')
axis([-radiusy-0.5 radiusy+0.5 -radiusx-0.5  radiusx+0.5]);
hold off
return
```

MATLAB Function "circ_array.m" Listing

```
function [pattern,amn] = circ_array(N,dolxr,dolyr,theta0,phi0,winid,win,nbits);
%%%%%% ********************** %%%%%%%%%%%%%%
% This function computes the 3-D directive gain patterns for a circular planar array
% This function uses the fft2 to compute its output. It assumes that there are the same number
% of elements along the major x- and y-axes
%%%%%%%%% ************ INPUTS ************ %%%%%%%%%% N ==> number of elements along x-aixs or y-axis
% dolxr ==> element spacing in x-direction; dolyr ==> element spacing in y-direction. Both are in lambda units
% theta0 ==> elevation steering angle in degrees, phi0 ==> azimuth steering angle in degrees
% This function uses the function (rec_to_circ) which computes the circular array from a square
% array (of size NXN) using the notation developed by ALLEN,J.L.,"The Theory of Array Antennas
% (with Emphasis on Radar Application)" MIT-LL Technical Report No. 323,July, 25 1965.
% winid ==> window identifier; winid negative ==> no window ; winid positive ==> use window given by win
% win ==> input window function (2-D window) MUST be of size (Nxr X Nyr)
% nbits is the number of nbits used in phase quantization; nbits negative ==> NO quantization
%%%%%%%%%% *********** OUTPUTS ************* %%%%%%%%%
% amn ==> array of ones and zeros; ones indicates true element location on the grid
% zeros mean no elements at that location; pattern ==> directive gain pattern
%%%%%%%%%%%% ********************** %%%%%%%%%%
eps = 0.0001;
nx = 0:N-1;
ny = 0:N-1;
i = sqrt(-1);
```

```
% check that window size is the same as the array size
[nw,mw] = size(win);
if winid >0
  if mw ~= N
    fprintf('STOP == Window size must be the same as the array')
     return
  end
  if nw ~= N
    fprintf('STOP == Window size must be the same as the array')
     return
  end
end
%if dol is > 0.5 then; choose dol = 0.5 and compute new N
if(dolxr <=0.5)
  ratiox = 1 ;
  dolx = dolxr ;
  Nx = N ;
else
  ratiox = ceil(dolxr/.5) ;
  Nx = (N-1) * ratiox + 1 ;
  dolx = 0.5 ;
end
if(dolyr <=0.5)
  ratioy = 1 ;
  doly = dolyr ;
  Ny = N ;
else
  ratioy = ceil(dolyr/.5);
  Ny = (N-1)*ratioy + 1 ;
  doly = 0.5 ;
end
% choose proper size fft, for minimum value choose 256X256
Nrx = 10 * Nx;
Nry = 10 * Ny;
nfftx = 2^(ceil(log(Nrx)/log(2)));
nffty = 2^(ceil(log(Nry)/log(2)));
if nfftx < 256
   nfftx = 256;
end
if nffty < 256
   nffty = 256;
end
% generate array of elements with or without window
if winid < 0
  array = ones(N,N);
else
   array = win;
end
% convert steering angles (theta0, phi0) to radians
theta0 = theta0 * pi / 180;
phi0 = phi0 * pi / 180;
% convert steering angles (theta0, phi0) to U-V sine-space
u0 = sin(theta0) * cos(phi0);
v0 = sin(theta0) * sin(phi0);
```

```
% Use formula thetal = (2*pi*n*dol) * sin(theta0) divided into 2^m levels
% and rounded to the nearest quantization level
if nbits < 0
   phasem = exp(i*2*pi*dolx*u0 .* nx * ratiox);
   phasen = exp(i*2*pi*doly*v0 .* ny * ratioy);
else
   levels = 2^nbits;
   qlevels = 2.0*pi / levels; % compute quantization levels
   sinthetaq = round(dolx .* nx * u0 * levels * ratiox) .* qlevels; % vector of possible angles
   sinphiq = round(doly .* ny * v0 * levels *ratioy) .* qlevels; % vector of possible angles
   phasem = exp(i*sinthetaq);
   phasen = exp(i*sinphiq);
end
% add the phase shift terms
array = array .* (transpose(phasem) * phasen) ;
% determine if interpolation is needed (i.e N > Nr)
if (Nx > N )| (Ny > N)
  for xloop = 1 : N
     temprow = array(xloop, :) ;
     w( (xloop-1)*ratiox+1, 1:ratioy:Ny) =  temprow ;
  end
  array = w;
else
   w(1:Nx, :) = array(1:N,:);
end
% Convert rectangular array into circular using function rec_to_circ
[m,n] = size(w) ;
NC = max(m,n);  % Use Allens algorithm
if Nx == Ny
   temp_array = w;
else
   midpoint = (NC-1)/2 +1 ;
   midwm = (m-1)/2 ;
   midwn = (n-1)/2 ;
   temp_array = zeros(NC,NC);
   temp_array(midpoint-midwm:midpoint+midwm, midpoint-midwn:midpoint+midwn) = w ;
end
amn = rec_to_circ(NC);  % must be rectangular array (Nx=Ny)
amn = temp_array .* amn ;
% Compute array pattern
arrayfft = abs(fftshift(fft2(amn,nfftx,nffty))).^2 ;
%compute [su,sv] matrix
U = [-nfftx/2:(nfftx/2)-1] ./(dolx*nfftx);
indexx = find(abs(U) <= 1);
U = U(indexx);
V = [-nffty/2:(nffty/2)-1] ./(doly*nffty);
indexy = find(abs(V) <= 1);
V = V(indexy);
[SU,SV] = meshgrid(V,U);
indx = find((SU.^2 + SV.^2) >1);
arrayfft(indx) = eps/10;
%Normalize to generate gain patern
rbar=sum(sum(arrayfft(indexx,indexy))) / dolx/doly/4./nfftx/nffty;
```

```
arrayfft = arrayfft(indexx,indexy) ./rbar;
[SU,SV] = meshgrid(V,U);
indx = find((SU.^2 + SV.^2) >1);
arrayfft(indx) = eps/10;
pattern = 10*log10(arrayfft +eps);
figure(1)
mesh(V,U,pattern);
xlabel('V')
ylabel('U');
zlabel('Gain pattern - dB')
figure(2)
contour(V,U,pattern)
axis image
grid
xlabel('V')
ylabel('U');
axis([-1 1 -1 1])
figure(3)
x0 = (NC+1)/2 ;
y0 = (NC+1)/2 ;
radiusx = dolx*((NC-1)/2 + 0.05/dolx) ;
radiusy = doly*((NC-1)/2 + 0.05/dolx) ;
theta = 5  ;
[xxx, yyy]=find(abs(amn)>0);
xxx = xxx-x0 ;
yyy = yyy-y0 ;
plot(yyy*doly, xxx*dolx,'rx')
axis equal
hold on
axis([-radiusy-0.5 radiusy+0.5 -radiusx-0.5  radiusx+0.5]);
grid
title('antenna spacing pattern');
xlabel('y - \lambda units')
ylabel('x - \lambda units')
[x, y]= makeellip( 0, 0, radiusx, radiusy, theta) ;
plot(y, x) ;
axis([-radiusy-0.5 radiusy+0.5 -radiusx-0.5  radiusx+0.5]);
[xxx0, yyy0]=find(abs(amn)<=0);
xxx0 = xxx0-x0 ;
yyy0 = yyy0-y0 ;
plot(yyy0*doly, xxx0*dolx,'co')
axis([-radiusy-0.5 radiusy+0.5 -radiusx-0.5  radiusx+0.5]);
axis equal
hold off ;
return
```

MATLAB Function "rect_to_circ.m" Listing

```
function amn = rec_to_circ(N)
midpoint = (N-1)/2 + 1;
amn = zeros(N);
array1(midpoint,midpoint) = N;
x0 = midpoint;
y0 = x0;
```

```
for i = 1:N
   for j = 1:N
      distance(i,j) = sqrt((x0-i)^2 + (y0-j)^2);
   end
end
idx = find(distance < (N-1)/2 + .025);
amn (idx) = 1;
return
```

MATLAB Program "Fig15_51.m" Listing

```
%Use this program to reproduce Fig. 15.51 of text
clear all
close all
d = 0.6; % element spacing in lambda units
betadeg = linspace(0,22.5,1000);
beta = betadeg .*pi ./180;
den = pi*d .* sin(beta);
numarg = den;
num = sin(numarg);
lscan = (num./den).^-4;
LSCAN = 10*log10(lscan+eps);
figure (1)
plot(betadeg,LSCAN,'linewidth',1.5)
xlabel('\bfscan angle in degrees')
ylabel('\bfScan loss in dB')
grid
title('Element spacing is d = 0.6 \lambda ')
```

Chapter 16

Adaptive Signal Processing

The emphasis in this chapter is on adaptive signal processing to include adaptive array processing and Space Time Adaptive processing (STAP). Adaptive arrays employ phased array antennas to adaptively sense and eliminate unwanted signals entering the radar's Field of View (FOV) while enhancing reception about the desired target returns. For this purpose, adaptive arrays utilize a rather complicated combination of hardware and require demanding levels of software implementation. Through feedback networks, a proper set of complex weights is computed and applied to each channel of the array.

STAP processing refers to the ability to simultaneously process spatial sensor and temporal (time dependent) input data. For this purpose, phased arrays (spatial component) along with time delay units (temporal component) are used to optimally detect targets in the presence of high clutter or interference environment.

16.1. Nonadaptive Beamforming

In adaptive beamforming the beam of interest is formed (generated) by continuously changing a set of weights through feedback circuits to minimize an output error signal. Nonadaptive or conventional beamformers do the same thing in the sense that the beam of interest is generated using a set of unique weights. Except in this case, these weights are determined a priori so that interference from a specific angle of arrival is minimized or eliminated. Different sets of weights will produce nulls in different directions in the array's field of view.

Consider a linear array of N equally spaced elements, and a plane wave $\exp(j2\pi f_0 t)$) incident on the aperture with direction-sine $\sin\theta$, as shown in Fig. 16.1. The weights $w_i,\ \ i = 0, 1, \ldots N-1$ are, in general, complex constants. The output of the beamformer is

$$y(t) = \sum_{n=0}^{N-1} w_n x_n(t-\tau_n) \qquad \textbf{Eq. (16.1)}$$

$$\tau_n = n\frac{d}{c}\sin\theta;\ \ n = 0, 1, \ldots, (N-1) \qquad \textbf{Eq. (16.2)}$$

where d is the element spacing and c is the speed of light. Fourier transformation of Eq. (16.1) yields

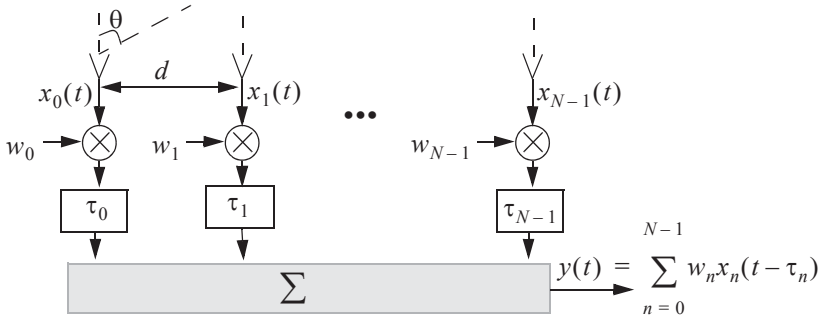


Figure 16.1. A linear array of size N, element spacing d, and an incident plane wave defined by $\sin\theta$.

$$Y(\omega) = \sum_{n=0}^{N-1} w_n X_n(\omega) exp(-j\omega\tau_n) = \sum_{n=0}^{N-1} w_n X_n(\omega) e^{-jn\Delta\theta}. \quad \textbf{Eq. (16.3)}$$

The phase term $\Delta\theta$ is defined as

$$\Delta\theta = 2\pi f_0 \frac{d}{c}\sin\theta = \frac{2\pi}{\lambda} d\sin\theta, \quad \textbf{Eq. (16.4)}$$

$\omega = 2\pi f_0$ and $f_0/c = 1/\lambda$. Eq. (16.3) can be written in vector form as

$$\mathbf{Y} = \mathbf{s}^{\dagger}\mathbf{x} \quad \textbf{Eq. (16.5)}$$

$$\mathbf{s}^{\dagger} = \left[1 \quad e^{j\Delta\theta} \quad \ldots \quad e^{j(N-1)\Delta\theta}\right] \quad \textbf{Eq. (16.6)}$$

$$\mathbf{x}^{\dagger} = \left[w_0 X_o \; w_1 X_1 \; \ldots \; \ldots w_{N-1} X_{N-1}\right]^{*} \quad \textbf{Eq. (16.7)}$$

where the superscripts * and †, respectively, indicate complex conjugate and complex conjugate transpose.

Let A_1 be the amplitude of the wavefront defined by $\sin\theta_1$; it follows that the vector $\mathbf{x}$ is given by

$$\mathbf{x} = A_1 \mathbf{s}_1^{*} \quad \textbf{Eq. (16.8)}$$

where $\mathbf{s}_1$ is a steering vector and can be written as,

$$\mathbf{s}^{\dagger}_1 = \left[w_0 \quad w_1 e^{-j\Delta\theta_1} \ldots w_{N-1} e^{-j(N-1)\Delta\theta_1}\right] \; ; \; \Delta\theta_1 = \frac{2\pi d}{\lambda} \cdot \sin\theta_1. \quad \textbf{Eq. (16.9)}$$

Using this notation, Eq. (16.5) can be expressed in the form

$$\mathbf{Y} = \mathbf{s}^{\dagger}\mathbf{x} = A_1\mathbf{s}^{\dagger}\mathbf{s}^{*}{}_1 . \qquad \textbf{Eq. (16.10)}$$

The array pattern of the beam steered at θ_1 is computed as the expected value of $\mathbf{Y}$. In other words, the power spectrum density for the beamformer output is given by

$$S(k) = E[\mathbf{Y}\mathbf{Y}^{\dagger}] = P_1\mathbf{s}^{\dagger}\Re\mathbf{s} \qquad \textbf{Eq. (16.11)}$$

where $P_1 = E[|A_1|^2]$ and $\Re$ is the correlation matrix given by

$$\Re = E\{\mathbf{s}_1\mathbf{s}^{\dagger}{}_1\} . \qquad \textbf{Eq. (16.12)}$$

Consider L incident plane waves with directions of arrival defined by

$$\Delta\theta_i = \frac{2\pi d}{\lambda}\sin\theta_i ;\ i = 1, L . \qquad \textbf{Eq. (16.13)}$$

The nth sample at the output of the mth sensor is

$$y_m(n) = \upsilon(n) + \sum_{i=1}^{L} A_i(n)exp(-jm\Delta\theta_i);\ m = 0, N-1 \qquad \textbf{Eq. (16.14)}$$

where $A_i(n)$ is the amplitude of the ith plane wave and $\upsilon(n)$ is white, zero-mean noise with variance σ_{υ}^2, and it is assumed to be uncorrelated with the signals. Equation (16.44) can be written in vector notation as

$$\mathbf{y}(n) = \upsilon(n) + \sum_{i=1}^{L} A_i(n)\mathbf{s}_i^{*} . \qquad \textbf{Eq. (16.15)}$$

A set of L steering vectors is needed to simultaneously form L beams. Define the steering matrix $\aleph$ as

$$\aleph = \left[\mathbf{s}_1\ \mathbf{s}_2\ \ldots\ \mathbf{s}_L\right] . \qquad \textbf{Eq. (16.16)}$$

Then the autocorrelation matrix of the field measured by the array is

$$\Re = E\{\mathbf{y}_m(n)\mathbf{y}_m{}^{\dagger}(n)\} = \sigma_{\upsilon}^2\mathbf{I} + \aleph\mathbf{C}\aleph^{\dagger} \qquad \textbf{Eq. (16.17)}$$

where $\mathbf{C} = dig\left[P_1\ P_2\ \ldots\ P_L\right]$, and $\mathbf{I}$ is the identity matrix.

For example, consider the case depicted in Fig. 16.2, where an interfering signal is located at angle $\theta_i = \pi/6$ off the antenna boresight. The desired signal is at $\theta_t = 0°$. The desired output should contain only the signal $s(t)$. From Eq. (16.3) and Eq. (16.4), the desired output is

$$y_d(t) = \sum_{n=0}^{1} w_n x_n(t-\tau_{n_t}) = w_0x_0 + w_1x_1e^{-j\frac{2\pi}{\lambda}d\sin\theta_t} . \qquad \textbf{Eq. (16.18)}$$

Since the angle $\theta_t = 0°$, it follows that

$$y_d(t) = \{Ae^{j2\pi f_0 t}\}\{(w_{0R}+jw_{0I}) + (w_{1R}+jw_{1I})\} \qquad \textbf{Eq. (16.19)}$$

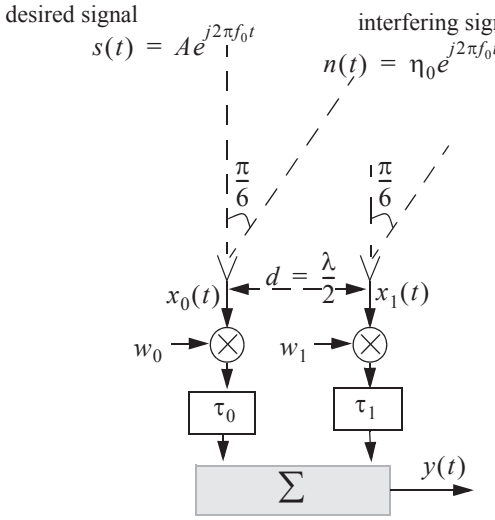


Figure 16.2. Two-element array with an interfering signal at $\theta_i = \pi/6$.

$$\begin{aligned} w_0 &= w_{0R} + jw_{0I} \\ w_1 &= w_{1R} + jw_{1I} \end{aligned} \quad \text{Eq. (16.20)}$$

Thus, in order to produce the desired signal, $s(t)$, at the output of the beamformer, it is required that

$$\begin{aligned} w_{0R} + w_{1R} = 1 &\Rightarrow w_{0R} = 1 - w_{1R} \\ w_{0I} + w_{1I} = 0 &\Rightarrow w_{0I} = -w_{1I} \end{aligned} \quad \text{Eq. (16.21)}$$

Next, the output due to the interfering signal is

$$y_i(t) = \sum_{n=0}^{1} w_n x_n(t - \tau_{n_t}) = w_0 x_0 + w_1 x_1 e^{-j\frac{2\pi}{\lambda} d \sin\theta_i} \quad \text{Eq. (16.22)}$$

Since the angle $\theta_i = \pi/6$, it follows that

$$y_i(t) = \{\eta_0 e^{j2\pi f_0 t}\}\{(w_{0R} + jw_{0I}) - j(w_{1R} + jw_{1I})\}, \quad \text{Eq. (16.23)}$$

and in order to eliminate the interference signal from the output of the beamformer, it is required that

$$\begin{aligned} w_{0R} + w_{1I} = 0 &\Rightarrow w_{0R} = -w_{1I} \\ w_{0I} - w_{1R} = 0 &\Rightarrow w_{0I} = w_{1R} \end{aligned} \quad \text{Eq. (16.24)}$$

Solving Eq. (16.21) and Eq. (16.24) yields

$$w_{0R} = \frac{1}{2}; w_{0I} = \frac{1}{2}; w_{1R} = \frac{1}{2}; w_{1I} = \frac{-1}{2}. \quad \text{Eq. (16.25)}$$

Using the weights given in Eq. (16.25) will allow the desired signal to get through the beamformer unaffected; however, the interference signal will be completely eliminated from the output.

16.2. Adaptive Signal Processing Using Least Mean Square (LMS)

Adaptive signal processing evolved as a natural evolution from adaptive control techniques of time-varying systems. Advances in digital processing computation techniques and associated hardware have facilitated maturing adaptive processing techniques and algorithms.

Consider the basic adaptive digital system shown in Fig. 16.3. The system input is the sequence $x[k]$ and its output is the sequence $y[k]$. What differentiates adaptive from non-adaptive systems is that in adaptive systems the transfer function $H_k(z)$ is now time varying. The arrow through the transfer function box is used to indicate adaptive processing (or time varying transfer function). The sequence $d[k]$ is referred to as the *desired* response sequence. The error sequence is the difference between the desired response and the actual response. Remember that the desired sequence is not completely known; otherwise, if it were completely known, one would not need any adaptive processing to compute it. The definition of this desired response is dependent on the system-specific requirements.

Many different techniques and algorithms have been developed to minimize the error sequence. Using one technique over another depends heavily on the operating environment under consideration. For example, if the input sequence is a stationary random process, then minimizing the error signal is nothing more than solving the least mean squares problem. However, in most adaptive processing systems, the input signal is a non-stationary process. In this section, the least mean squares technique is examined.

The least mean squares (LMS) algorithm is the most commonly utilized algorithm in adaptive processing, primary because of its simplicity. The time-varying transfer function of order L can be written as a Finite Impulse Response (FIR) filter defined by

$$H_k(z) = b_0 + b_1 z^{-1} + \ldots + b_L z^{-L}. \qquad \textbf{Eq. (16.26)}$$

The input output relationship is given by the discrete convolution

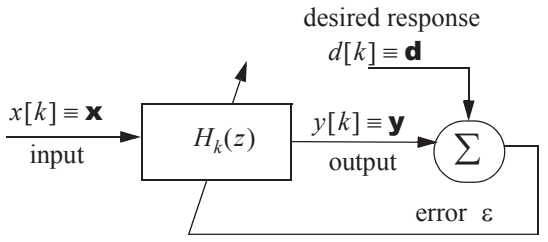


Figure 16.3. Basic adaptive system.

$$y(k) = \sum_{n=0}^{L} b_n(k)x(k-n)\,. \qquad \text{Eq. (16.27)}$$

The goal of the adaptive LMS process is to adjust the filter coefficients toward an optimum minimum mean square error (MMSE). The most common approach to achieving this MMSE utilizes the method of steepest descent. For this purpose, define the filter coefficients in vector notation as

$$\mathbf{b}_k = \left[b_o(k)\ b_1(k)\ldots\quad b_L(k)\right]^{\dagger}, \qquad \text{Eq. (16.28)}$$

then

$$\mathbf{b}_{k+1} = \mathbf{b}_k - \mu\nabla_k \qquad \text{Eq. (16.29)}$$

where μ is a parameter that controls how fast the error converges to the desired MMSE value, and the gradient vector ∇_k is defined by

$$\nabla_k = \frac{\partial}{\partial \mathbf{b}_k}E[\varepsilon_k^2] = \left[\frac{\partial}{\partial \mathbf{b}_0(k)}E[\varepsilon_k^2]\ \ldots\ \frac{\partial}{\partial \mathbf{b}_L(k)}E[\varepsilon_k^2]\right]^{\dagger}. \qquad \text{Eq. (16.30)}$$

As clearly indicated by Eq. (16.29), the adaptive filter coefficients update rate is proportional to the negative gradient; thus, if the gradient is known at each step of the adaptive process, then better computation of the coefficient is obtained. In other words, the MMSE decreases from step k to step $k+1$. Of course, once the solution is found, the gradient becomes zero and the coefficient will not change any more.

When the gradient is not known, estimates of the gradient are used based only on the instantaneous squared error. These estimates are defined by

$$\hat{\nabla}_k = \frac{\partial}{\partial \mathbf{b}_k}[\varepsilon_k^2] = 2\varepsilon_k\frac{\partial}{\partial \mathbf{b}_k}(d_k - y_k)\,. \qquad \text{Eq. (16.31)}$$

Since the desired sequence $d[k]$ is independent from the output $y[k]$, Eq. (16.31) can be written as

$$\hat{\nabla}_k = -2\varepsilon_k\mathbf{x}_k \qquad \text{Eq. (16.32)}$$

where the vector $\mathbf{x}_k$ is the input signal sequence. Substituting Eq. (16.32) into Eq. (16.29) yields

$$\mathbf{b}_{k+1} = \mathbf{b}_k + 2\varepsilon_k\mu\mathbf{x}_k\,. \qquad \text{Eq. (16.33)}$$

The choice of the convergence parameter μ plays a significant role in determining the system performance. This is clear because as indicated by Eq. (16.33), a successful implementation of the LMS algorithm depends on the input signal, the choice of the desired signal, and the convergence parameter. Much research and effort has been devoted to selecting the optimal value for μ. Nonetheless, no universal value has been found. However, a range for this parameter has been determined to be $0 < \mu < 1$.

Often, a normalized value for the convergence parameter μ_N can be used instead of its absolute value. That is,

$$\mu_N = \frac{\mu}{(L+1)\sigma^2} \qquad \textbf{Eq. (16.34)}$$

where L is the order of the adaptive FIR filter and σ^2 is the variance (power) of the input signal. When the input signal is not stationary and its variance is varying with time, a time-varying estimate of σ^2 is used. That is

$$\hat{\sigma}_k^2 = \alpha x_k^2 + (1-\alpha)\hat{\sigma}_{k-1}^2 \qquad \textbf{Eq. (16.35)}$$

where α is a factor selected such that $0 < \alpha < 1$. Finally, Eq. (16.33) can be written as

$$\mathbf{b}_{k+1} = \mathbf{b}_k + \frac{2\varepsilon_k \mu \mathbf{x}_k}{(L+1)\hat{\sigma}_k^2} \qquad \textbf{Eq. (16.36)}$$

MATLAB Function "LMS.m"

The MATLAB function "LMS.m" implements Eq. (16.36). Its syntax is as follows:

```
Xout = LMS(Xin, D, B, mu, sigma, alpha)
```

where

Symbol	Description	Status
X	*input data sequence - corrupted*	*input*
D	*desired signal sequence*	*input*
B	*adaptive coefficient*	*input*
mu	*convergence parameters*	*input*
sigma	*input signal power estimate*	*input*
alpha	*forgetting factor, see Eq. (16.35)*	*input*
Xout	*predicted out sequence*	*output*

As an example and in reference to Fig. 16.3, let the input and desired signals be defined as

$$x[k] = \sqrt{2}\ \sin\left(\frac{2\pi k}{20}\right) + n[k] \qquad ;\ k = 0, 1, \ldots, 500 \qquad \textbf{Eq. (16.37)}$$

$$d[k] = \sqrt{2}\ \sin\left(\frac{2\pi k}{20}\right) \qquad ;\ k = 0, 1, \ldots, 500 \qquad \textbf{Eq. (16.38)}$$

where $n[k]$ is additive white noise with zero mean and variance $\sigma_n^2 = 2$. Figure 16.4 shows the output of the LMS algorithm defined in Eq. (16.36) when $\mu = 0.1$ and $\alpha = 0$. Figure 16.5 is similar to Fig. 16.4 except in this case, $\mu = 0.01$ and $\alpha = 0.1$.

Note that in Fig. 16.5, the rate of convergence is reduced since μ is smaller than that used in Fig. 16.4; however, the filter's output is less noise because α is greater than zero, which allows for more accurate updates of the noise variance as defined in Eq. (16.35). These plots can be reproduced using the MATLAB program *"Fig16_4_5.m,"* listed in Appendix 16-A.

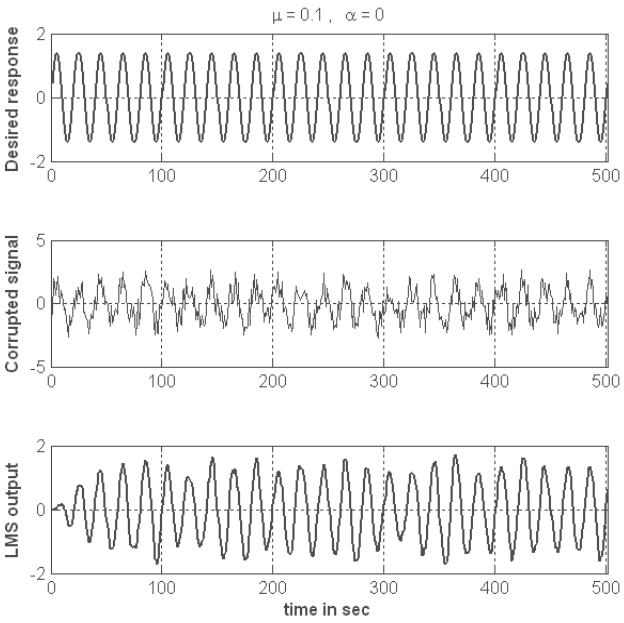


Figure 16.4. Input signal, desired response, and output response of an LMS filter.

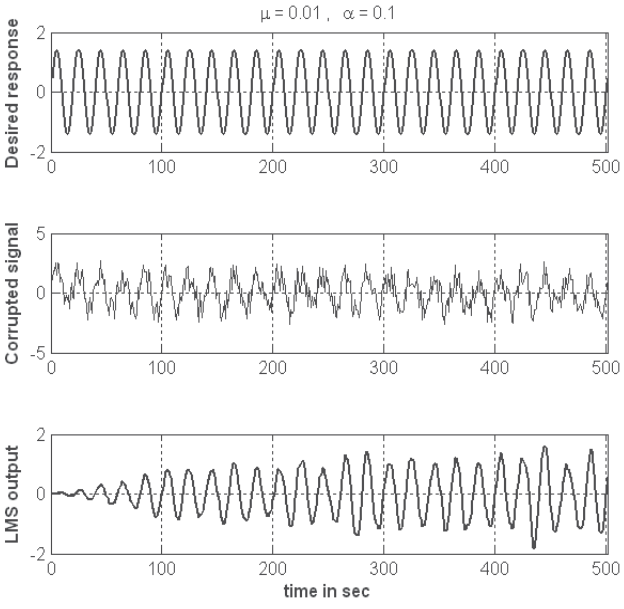


Figure 16.5. Input signal, desired response, and output response of an LMS filter.

16.3. The LMS Adaptive Array Processing

Consider the LMS adaptive array shown in Fig. 16.6. The difference between the reference signal and the array output constitutes an error signal. The error signal is then used to adaptively calculate the complex weights, using a predetermined convergence algorithm. The reference signal is assumed to be an accurate approximation of the desired signal (or desired array response). This reference signal can be computed using a training sequence or spreading code, which is supposed to be known at the radar receiver. The format of this reference signal will vary from one application to another. But in all cases, the reference signal is assumed to be correlated with the desired signal. An increased amount of this correlation significantly enhances the accuracy and speed of the convergence algorithm being used. In this section, the LMS algorithm is assumed.

In general, the complex envelope of a bandpass signal and its corresponding analytical (preenvelope) signal can be written using the quadrature components pair $(x_I(t), x_Q(t))$. Recall that the quadrature components are related using the Hilbert transform as follows:

$$x_Q(t) = \hat{x}_I(t) \text{ ; and } \hat{x}_Q = -x_I \qquad \textbf{Eq. (16.39)}$$

where $\hat{x}_I$ and $\hat{x}_Q$ are, respectively, the Hilbert transforms of x_I and x_Q. A bandpass signal $x(t)$ can be expressed as follows (visit Chapter 3 for a refresher):

$$x(t) = x_I(t)\cos 2\pi f_0 t - x_Q(t)\sin 2\pi f_0 t \qquad \textbf{Eq. (16.40)}$$

$$\psi_x(t) = x(t) + j\hat{x}(t) \equiv \tilde{x}(t)e^{j2\pi f_0 t} \qquad \textbf{Eq. (16.41)}$$

$$\tilde{x}(t) = x_I(t) + jx_Q(t) \qquad \textbf{Eq. (16.42)}$$

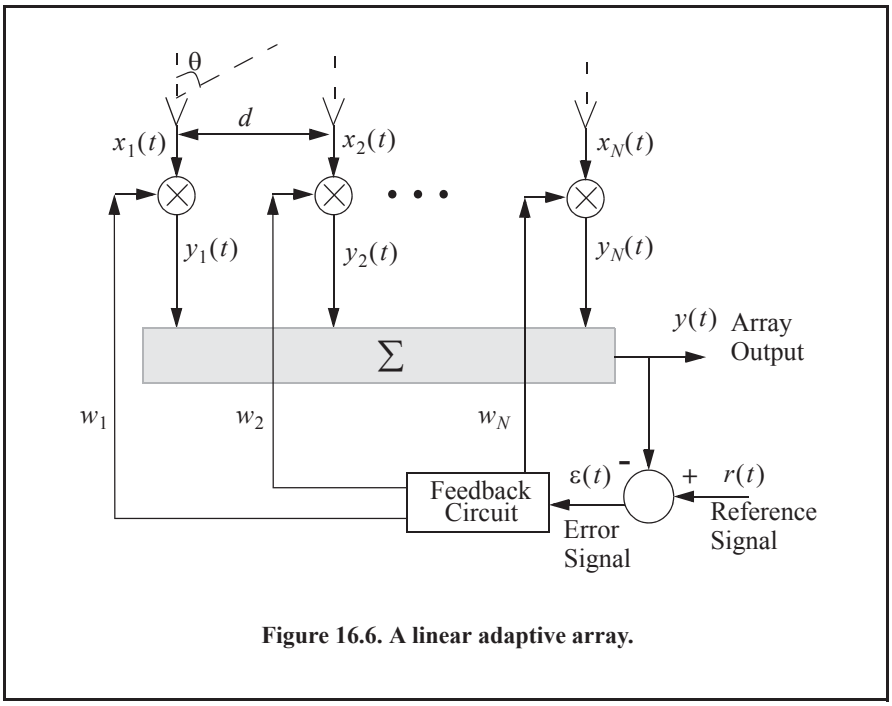


Figure 16.6. A linear adaptive array.

where $\psi(t)$ is the pre-envelope and $\tilde{x}(t)$ is the complex envelope. Equation (16.42) can be written using Eq. (16.39) as

$$\tilde{x}(t) = x_I(t) + jx_Q(t) = x_I(t) + j\hat{x}_I(t). \quad \textbf{Eq. (16.43)}$$

Using this notation, the adaptive array output signal, its reference signal, and the error signal can also be written using the same notation as

$$y(t) = y(t) + j\hat{y}(t) \quad \textbf{Eq. (16.44)}$$

$$\tilde{r}(t) = r(t) + j\hat{r}(t) \quad \textbf{Eq. (16.45)}$$

$$\tilde{\varepsilon}(t) = \varepsilon(t) + j\hat{\varepsilon}(t). \quad \textbf{Eq. (16.46)}$$

Referencing Fig. 16.6, denote the output of the n^{th} array input signal as $y_n(t)$ and assume complex weights given by

$$w_n = w_{nI} - jw_{nQ}. \quad \textbf{Eq. (16.47)}$$

It follows that

$$y_n(t) = w_{nI}\ x_{nI}(t) + w_{nQ}\ x_{nQ}(t). \quad \textbf{Eq. (16.48)}$$

Taking the Hilbert transform of Eq. (16.48) yields

$$\hat{y}_n(t) = w_{nI}\ \hat{x}_{nI}(t) + w_{nQ}\ \hat{x}_{nQ}(t). \quad \textbf{Eq. (16.49)}$$

By using Eq. (16.39) into Eq. (16.49), one gets

$$\hat{y}_n(t) = w_{nI}\ x_{nQ}(t) - w_{nQ}\ x_{nI}(t). \quad \textbf{Eq. (16.50)}$$

The n^{th} channel analytic signal is

$$\psi_{y_n}(t) = y_n(t) + j\hat{y}_n(t)\ . \quad \textbf{Eq. (16.51)}$$

Substituting Eq. (16.48) and Eq. (16.49) into Eq. (16.50) gives

$$\psi_{y_n}(t) = w_{nI}\ x_{nI}(t) + w_{nQ}\ x_{nQ}(t) + j[w_{nI}\ x_{nQ}(t) - w_{nQ}\ x_{nI}(t)]. \quad \textbf{Eq. (16.52)}$$

Collecting terms yields, using complex notation,

$$\psi_{y_n}(t) = w_n\tilde{x}_n(t). \quad \textbf{Eq. (16.53)}$$

Therefore, the output of the entire adaptive array is

$$\tilde{y}(t) = \sum_{n=1}^{N} \tilde{y}_n(t) = \sum_{n=1}^{N} w_n\tilde{x}_n(t), \quad \textbf{Eq. (16.54)}$$

which can be written using vector notation as

$$\tilde{\mathbf{y}} = \mathbf{w}^t\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^t\mathbf{w} \quad \textbf{Eq. (16.55)}$$

where the vectors $\mathbf{x}$ and $\mathbf{w}$ are given by

$$\tilde{\mathbf{x}} = \left[\tilde{x}_1(t)\ \tilde{x}_2(t)\ \ldots \tilde{x}_N(t)\right]^t \qquad \text{Eq. (16.56)}$$

$$\mathbf{w} = \left[w_1\ w_2\ \ldots w_N\right]^t. \qquad \text{Eq. (16.57)}$$

The superscript $\{\ ^t\}$ indicates the transpose operation.

As discussed earlier, one common technique to achieving the MMSE of an LMS algorithm is to use the steepest descent. Thus, the complex weights in the LMS adaptive array are related as defined in Eq. (16.29). That is,

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu\nabla_k \qquad \text{Eq. (16.58)}$$

where again, μ is the convergence parameter. The subscript k indicates time samples. In this case, the gradient vector ∇_k is defined by

$$\nabla_k = \frac{\partial}{\partial \mathbf{w}_k} E[\tilde{\varepsilon}_k^2] = \left[\frac{\partial}{\partial \mathbf{w}_0(k)} E[\tilde{\varepsilon}_k^2]\ \ldots\ \frac{\partial}{\partial \mathbf{w}_N(k)} E[\tilde{\varepsilon}_k^2]\right]^t. \qquad \text{Eq. (16.59)}$$

Rearranging Eq. (16.58) so that the rate of change between consecutive estimates of the complex weights is on one side of the equation yields

$$\mathbf{w}_{k+1} - \mathbf{w}_k = -\mu\frac{\partial}{\partial \mathbf{w}_k}(E[\tilde{\varepsilon}_k^2]) \qquad \text{Eq. (16.60)}$$

where the middle portion of Eq. (16.59) was also substituted for the gradient vector. In this format, the left-hand side of Eq. (16.60) represents the rate of change of the complex weights with respect to time (i.e., the derivative of the weights with respect to time). It follows that

$$\frac{d}{dt}\mathbf{w} = -\mu\frac{\partial}{\partial \mathbf{w}}(E[\tilde{\varepsilon}^2(t)]). \qquad \text{Eq. (16.61)}$$

However, see from Fig. 16.5, that the error signal complex envelope is

$$\tilde{\varepsilon}(t) = \tilde{r}(t) - \sum_{n=1}^{N} w_n\tilde{x}_n(t) \Rightarrow \tilde{\varepsilon}(t) = \tilde{r}(t) - \tilde{\mathbf{x}}^t\mathbf{w}. \qquad \text{Eq. (16.62)}$$

It can be shown (see Problem 16.1) that

$$\frac{\partial}{\partial \mathbf{w}} E[\tilde{\varepsilon}^2(t)] = -E[\tilde{\mathbf{x}}^*\tilde{\varepsilon}(t)]. \qquad \text{Eq. (16.63)}$$

Therefore, Eq. (16.61) can be written as

$$\frac{d}{dt}\mathbf{w} = \mu E[\tilde{\mathbf{x}}^*\tilde{\varepsilon}(t)]. \qquad \text{Eq. (16.64)}$$

Substituting Eq. (16.62) into Eq. (16.64) gives

$$\frac{d}{dt}\mathbf{w} = \mu E[\tilde{\mathbf{x}}^*(\tilde{r}(t) - \tilde{\mathbf{x}}^t\mathbf{w})]. \qquad \text{Eq. (16.65)}$$

Equivalently,

$$\frac{d}{dt}\mathbf{w} + \mu E[\tilde{\mathbf{x}}^*\tilde{\mathbf{x}}^t]\mathbf{w} = \mu E[\tilde{\mathbf{x}}^*\tilde{r}(t)]. \qquad \text{Eq. (16.66)}$$

The covariance matrix is by definition

$$\mathbf{C} = E[\tilde{\mathbf{x}}^*\tilde{\mathbf{x}}^t] = \begin{bmatrix} \tilde{x}_1^*\tilde{x}_1 & \tilde{x}_1^*\tilde{x}_2 & \cdots \\ \tilde{x}_2^*\tilde{x}_1 & \tilde{x}_2^*\tilde{x}_2 & \cdots \\ \cdots & \cdots & \end{bmatrix}, \qquad \text{Eq. (16.67)}$$

and the reference signal correlation vector $\mathbf{s}$ is

$$\mathbf{s} = E[\tilde{\mathbf{x}}^*\tilde{r}(t)] = E\left[\tilde{x}_1^*\tilde{r} \;\; \tilde{x}_2^*\tilde{r} \;\; \cdots\right]^t. \qquad \text{Eq. (16.68)}$$

Using Eq. (16.68) and Eq. (16.67), one can rewrite the differential equation (DE) given Eq. (16.66) as

$$\frac{d}{dt}\mathbf{w} + \mu\mathbf{C}\mathbf{w} = \mu\mathbf{s}. \qquad \text{Eq. (16.69)}$$

The steady state solution for the DE defined in Eq. (16.69) (provided that the covariance matrix is not singular) is

$$\mathbf{w} = \mathbf{C}^{-1}\,\mathbf{s}. \qquad \text{Eq. (16.70)}$$

As the size of the covariance matrix increases (i.e., number of channels in the adaptive array), so does the complexity associated with computing the adaptive weights in real time. This is true because computing the inverse of large matrices in real time can be extremely challenging and demands a significant amount of computing power. Consequently, the effectiveness of adaptive arrays has been limited to small-sized arrays, where only a few interfering signals can be eliminated (cancelled). Additionally, computing of a good estimate of the covariance matrix in real time is also difficult in practical applications. In order to mitigate that effect, a reasonable estimate for $E\{x_i x_j^*\}$ (the *i,j* element of the covariance matrix) is derived by averaging *m* independent samples of data from the same distribution. This approach can be extended to the entire covariance matrix by collecting *M* independent "snapshots" of data from N channels. Thus, the estimate of the covariance matrix can be given as,

$$\tilde{\mathbf{C}} \approx (\tilde{\mathbf{x}}^\dagger\tilde{\mathbf{x}})/M. \qquad \text{Eq. (16.71)}$$

The transient solution of Eq. (16.69) (see Problem 16.2) is

$$\mathbf{w}(t) = \sum_{n=1}^{N} \mathbf{p}_i e^{-\mu\lambda_i t} \qquad \text{Eq. (16.72)}$$

where the vectors $\mathbf{p}_i$ are constants that depend on the initial value of $\mathbf{w}(t)$, and λ_i are the eigenvalues of the matrix $\mathbf{C}$. It follows that the complete solution of Eq. (16.69) is

$$\mathbf{w}(t) = \sum_{n=1}^{N} \mathbf{p}_i e^{-\mu\lambda_i t} + \mathbf{C}^{-1}\,\mathbf{s}. \qquad \text{Eq. (16.73)}$$

A very common measure of effectiveness of an adaptive array is the ratio of the total output interference power, S_o, to the internal noise power, S_n.

Example:

Consider the two-element array in Section 16.2. Assume the desired signal is at directional-sine $\sin(\theta_t)$ *and the interference signal is at* $\sin(\theta_i)$. *Calculate the adaptive weights so that the interference signal is cancelled.*

Solution:

From Fig. 16.6

$$\tilde{x}_1(t) = \tilde{d}_1(t) + \tilde{n}_1(t) + \tilde{I}_1(t)$$

$$\tilde{x}_2(t) = \tilde{d}_2(t) + \tilde{n}_2(t) + \tilde{I}_2(t)$$

where d *is the desired response,* n *is the noise, signal, and* I *is the interference signal. The noise signal is spatially incoherent, more specifically*

$$E[\tilde{n}_i^*(t)\tilde{n}_j(t)] = \begin{cases} 0 & i \neq j \\ \sigma_n^2 & i = j \end{cases}.$$

Also

$$E[\tilde{d}_i^*(t)\tilde{n}_i(t)] = 0 \qquad for\ all\ (i,j)\ .$$

The desired signal is

$$\tilde{d}(t) = \tilde{d}_1(t) + \tilde{d}_2(t) = A_d e^{j2\pi f_0 t} e^{j\Theta_d} + A_d e^{j2\pi f_0 t} e^{j\Theta_d} e^{-j\pi \sin\theta_d}$$

where Θ_d *is a uniform random variable. The interference signal is*

$$\tilde{I}(t) = \tilde{I}_1(t) + \tilde{I}_2(t) = A_i e^{j2\pi f_0 t} e^{j\Theta_i} + A_i e^{j2\pi f_0 t} e^{j\Theta_i} e^{-j\pi \sin\theta_i}$$

where Θ_i *is a uniform random variable. Of course the random variables* Θ_d *and* Θ_i *are assumed to be statistically independent. In vector format,*

$$\tilde{\mathbf{x}}_d = A_d e^{j2\pi f_0 t} e^{j\Theta_d} \begin{bmatrix} 1 \\ e^{-j\pi \sin\theta_d} \end{bmatrix}$$

$$\tilde{\mathbf{x}}_i = A_i e^{j2\pi f_0 t} e^{j\Theta_i} \begin{bmatrix} 1 \\ e^{-j\pi \sin\theta_i} \end{bmatrix}\ .$$

Of course, the noise vector is

$$\tilde{\mathbf{x}}_n = \begin{bmatrix} \tilde{n}_1(t) \\ \tilde{n}_2(t) \end{bmatrix},$$

and the reference signal is (this is an assumption so that the desired and reference signal are correlated)

$$\tilde{r}(t) = A_r e^{j2\pi f_0 t} e^{j\Theta_d}.$$

Note that the input SNR is

$$SNR_d = A_d^2/\sigma_n^2$$

and the interference to noise ratio is

$$SNR_i = A_i^2/\sigma_n^2.$$

The input signal can be written using vector notation as

$$\tilde{\mathbf{x}} = \tilde{\mathbf{x}}_d + \tilde{\mathbf{x}}_i + \tilde{\mathbf{x}}_n.$$

The covariance matrix is computed from Eq. (16.67) as

$$\mathbf{C} = E[\tilde{\mathbf{x}}_d{}^* \tilde{\mathbf{x}}_d{}^t] = \begin{bmatrix} A_d^2 + A_i^2 + \sigma_n^2 & A_d^2 e^{-j\pi \sin\theta_d} + A_i^2 e^{-j\pi \sin\theta_i} \\ A_d^2 e^{j\pi \sin\theta_d} + A_i^2 e^{j\pi \sin\theta_i} & A_d^2 + A_i^2 + \sigma_n^2 \end{bmatrix}.$$

In order to compute the covariance matrix eigenvalue, one needs to compute the determinant first

$$|\mathbf{C}| = 4A_d^2 A_i^2 \left(\sin\left(\frac{\theta_d + \theta_i}{s}\right)\right)^2 + 2A_d^2\sigma_n^2 + 2A_i^2\sigma_n^2 + \sigma_n^4.$$

Thus,

$$\mathbf{C}^{-1} = \frac{1}{|\mathbf{C}|}\begin{bmatrix} A_d^2 + A_i^2 + \sigma_n^2 & -A_d^2 e^{-j\pi \sin\theta_d} - A_i^2 e^{-j\pi \sin\theta_i} \\ -A_d^2 e^{j\pi \sin\theta_d} - A_i^2 e^{j\pi \sin\theta_i} & A_d^2 + A_i^2 + \sigma_n^2 \end{bmatrix}.$$

The reference correlation vector is

$$\mathbf{s} = E[\tilde{\mathbf{x}}^* \tilde{r}(t)] = A_d A_r \begin{bmatrix} 1 \\ e^{j\pi \sin\theta_d} \end{bmatrix}.$$

It follows that the weights are

$$\mathbf{w} = \frac{A_d A_r}{|\mathbf{C}|}\begin{bmatrix} A_i^2 + \sigma_n^2 - A_i^2 e^{j\pi(\sin\theta_d - \sin\theta_i)} \\ e^{j\pi \sin\theta_d}\{A_i^2 + \sigma_n^2 - A_i^2 e^{j\pi(\sin\theta_i - \sin\theta_d)}\} \end{bmatrix}.$$

MATLAB Function "adaptive_array_lms.m"

The MATLAB function *"adaptive_array_lms.m"* implements the LMS adaptive array processing described in this section. Its syntax is as follows:

```
adaptive_array_lms(N, dol,tagt_angle, jam_angle)
```

where

Symbol	Description	Units	Status
N	*array size*	*none*	*input*
dol	*array element spacing*	*lambda*	*input*
tagt_angle	*desired beam spatial location*	*degrees*	*input*
jam_angle	*jammer spatial location*	*degrees*	*input*

The output of this function is a plot of the normalized array response in dB versus scan before and after adaptive processing is applied. Figure 16.7 shows an example using the following MATLAB call:

adaptive_array_lms(19, 0.5, 0, 35)

Note that the quality of the null (how deep and how narrow) heavily depends on the accuracy of the covariance matrix. In the *"adaptive_array_lms.m"* code, the MATLAB function *"mvnrnd"* was employed to estimate the noise vector used in computing the covariance matrix. It follows that each time the code is executed, a different covariance matrix is calculated, and hence it is very likely that the adaptive null will differ in appearance from one run to another. This is illustrated in Figs. 16.8a and 16.8b. In this case, the main beam is steered to $\theta = -10°$, while the jammer is located at $\theta = 25°$.

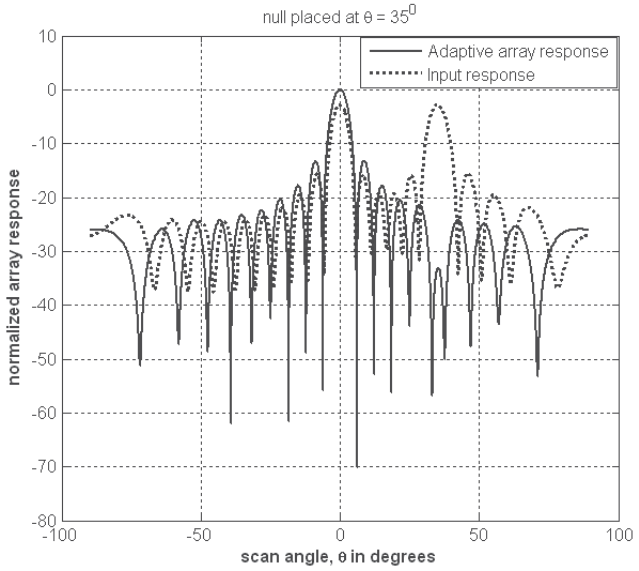


Figure 16.7. Demonstrating the LMS adaptive array processing. Note that the gain patterns have been purposefully normalized to slightly different maxima in order to make a clear visual distinction between the two patterns.

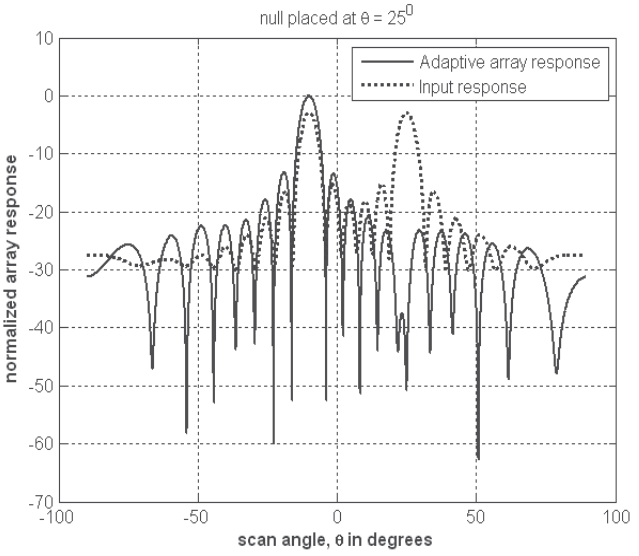


Figure 16.8a. Demonstrating the LMS adaptive array processing. Note that the gain patterns have been purposefully normalized to slightly different maxima in order to make a clear visual distinction between the two patterns.

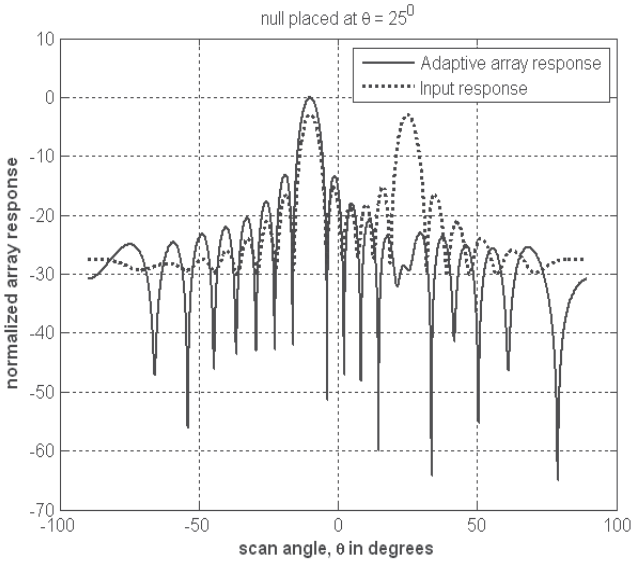


Figure 16.8b. Demonstrating the LMS adaptive array processing. Note that the gain patterns have been purposefully normalized to slightly different maxima in order to make a clear visual distinction between the two patterns.

16.4. Sidelobe Cancelers (SLC)

Sidelobe cancelers typically consist of a main antenna (which can be a phased array or a single element) and one or more auxiliary antennas. The main antenna is referred to as the main channel; it is assumed to be highly directional and is pointed toward the desired signal angular location. The interfering signal is assumed to be located somewhere off the main antenna boresight (in the sidelobes). Because of this configuration, the main channel receives returns from both the desired and the interfering signals. However, returns from the interfering signal in the main channel are weak because of the low main antenna sidelobe gain in the direction of the interfering signal. Also the auxiliary antenna returns are primarily from the interfering signal. This is illustrated in Fig. 16.9.

Referring to Fig. 16.9, $\tilde{s}(t)$ is the desired signal, $\tilde{n}(t)$ is the main channel noise signal, which is primarily from the interfering signal, while $\tilde{n}'(t)$ is the interfering signal in the auxiliary array. It is assumed that the signals $\tilde{s}(t)$ and $\tilde{n}(t)$ are uncorrelated. It is also assumed that the interfering signal is highly correlated with the noise signal in the main channel. The basic idea behind the SLC is to have the adaptive auxiliary channel produce an accurate estimate of the noise signal first, then to subtract that estimate from the main channel signal so that the output signal is mainly the desired signal.

The error signal is

$$\tilde{\varepsilon} = \tilde{\mathbf{d}} - \mathbf{w}^t \tilde{\mathbf{x}} \qquad \text{Eq. (16.74)}$$

where $\mathbf{x}$ is the vector of the auxiliary array signal, and $\mathbf{w}$ is the adapted weights. The vector $\mathbf{d}$ of size M. The residual power is

$$P_{res} = E[\tilde{\varepsilon}\tilde{\varepsilon}^{\dagger}] \qquad \text{Eq. (16.75)}$$

$$P_{res} = E[(\tilde{\mathbf{d}} - \mathbf{w}^t \tilde{\mathbf{x}})(\tilde{\mathbf{d}}^* - \tilde{\mathbf{x}}^{\dagger}\mathbf{w}^*)]\,. \qquad \text{Eq. (16.76)}$$

It follows that

$$P_{res} = E[|\tilde{\mathbf{d}}|^2] - E[\tilde{\mathbf{d}}\tilde{\mathbf{x}}^{\dagger}\mathbf{w}^*] - E[\tilde{\mathbf{d}}^*\mathbf{w}^t\tilde{\mathbf{x}}] - \mathbf{w}^t E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{\dagger}]\mathbf{w}^*\,. \qquad \text{Eq. (16.77)}$$

Differentiate the residual power with respect to $\mathbf{w}$ and setting the answer equal to zero (to compute the optimal weights that minimize the power residual) yields

$$\frac{\partial P_{res}}{\partial \mathbf{w}} = \mathbf{0} = -\tilde{\mathbf{x}}\tilde{\mathbf{d}} + \mathbf{C}_a\mathbf{w} \qquad \text{Eq. (16.78)}$$

where $\mathbf{C}_a$ is the covariance matrix of the auxiliary channel. Finally, the optimal weights are given by

$$\mathbf{w} = \mathbf{C}_a^{-1}\,\tilde{\mathbf{x}}\tilde{\mathbf{d}}\,. \qquad \text{Eq. (16.79)}$$

Note that the vector $\tilde{\mathbf{x}}\tilde{\mathbf{d}}$ represents the components that are common to both main and auxiliary channels. Note that Eq. (16.79) makes intuitive sense where the objective is to isolate the components in the data which are common to the main and auxiliary channels, and we then wish to give them some heavy attenuation (which comes from inverting $\mathbf{C}_a$).

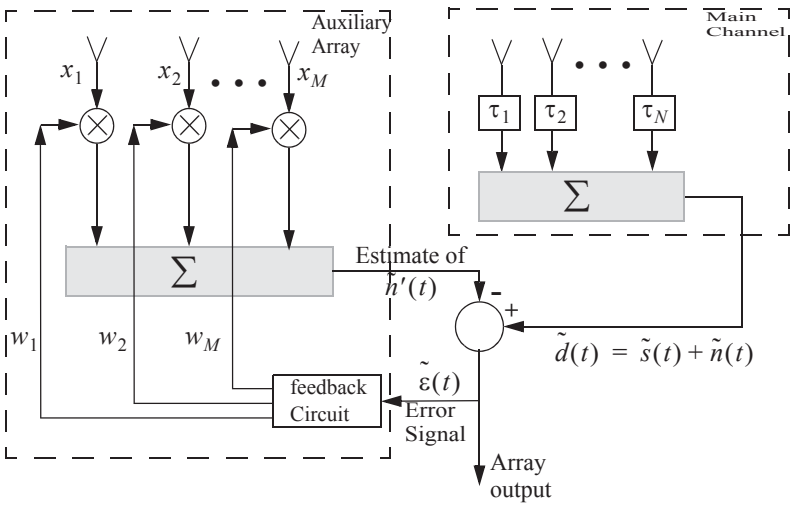


Figure 16.9. Sidelobe canceler array.

16.5. Space Time Adaptive Processing (STAP)

Space time adaptive processing (STAP) is the term used to describe adaptive arrays that simultaneously process spatial and temporal data. The spatial components of the signal are collected using the array sensors (same as in any array operation) while the temporal components of the signal are generated using time-delay units of equal intervals behind each array sensor. For this purpose, an array of size N will have N sub-channels (one behind each senor); within each sub-channel the signal from the j^{th} range bin comprises M pulses interleaved by the radar pulse repetition interval ($T = 1/f_r$) where f_r is the PRF. The outputs from all M delayed responses are then summed coherently, then all N channels are coherently summed to generate the composite array response. The array input is assumed to be made of target returns, clutter returns, and interfering signals (e.g., jammers) returns.

The material in this section is presented in the following sequence: First, the concept of space time beamforming is introduced; then the analysis is extended to encompass space time adaptive processing.

16.5.1. Space Time Processing

The configuration of a space time beamformer is illustrated in Fig. 16.10. In this case, an array of N sensors and M pulses (interleaved by the radar PRI) comprise the beamformer output for each range bin. The signal output of the n^{th} array sensor corresponding to the m^{th} pulse and j^{th} range bin is

$$y_i(t_j - mT) \quad ;\begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, J \\ m = 0, \ldots, M-1 \end{cases} \qquad \textbf{Eq. (16.80)}$$

where N is the number of sensors in the array, M is the number of pulses, and J is the number of range bins being processed.

Using this notation, the j^{th} range bin return signal from all pulses is given by

$$\mathbf{Y}_j = \begin{bmatrix} \mathbf{y}_{1j} \\ \mathbf{y}_{2j} \\ \\ \mathbf{y}_{Nj} \end{bmatrix} = \begin{bmatrix} y_1(t_j) & y_1(t_j - T) & y_1(t_j - 2T) & \dots & y_1(t_j - (M-1)T) \\ y_2(t_j) & y_2(t_j - T) & y_2(t_j - 2T) & & y_2(t_j - (M-1)T) \\ & & & \vdots & \\ y_N(t_j) & y_N(t_j - T) & y_N(t_j - 2T) & \dots & y_N(t_j - (M-1)T) \end{bmatrix} . \qquad \textbf{Eq. (16.81)}$$

In this manner, the space time beamformer receives a series of M pulses from each of the N array elements for each of the J range bins. Hence, a data cube of returns is generated, as illustrated in Fig. 16.11. For this purpose, the data received from the j^{th} range bin is made of $MN \times 1$ space (or time snapshots).

By taking element-1 the array phase reference, then the signal received by the n^{th} array element (or sensor) at time t_j from a far field target whose angle of arrival is θ can be computed with the help of Eq. (16.1) as

$$y_n(t_j) = x(t_j) e^{-j\Delta\theta_n} \qquad \textbf{Eq. (16.82)}$$

where

$$\Delta\theta_n = \frac{2\pi d}{\lambda}(n-1)\sin\theta \qquad ;n = 1, 2, \dots, N \qquad \textbf{Eq. (16.83)}$$

where, in general, the signal $x(t)$ is

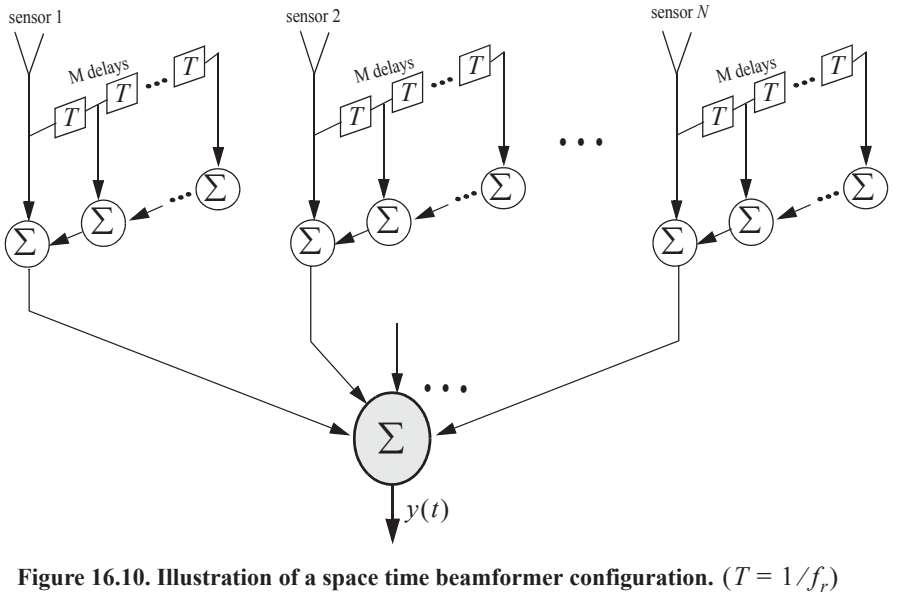


Figure 16.10. Illustration of a space time beamformer configuration. ($T = 1/f_r$)

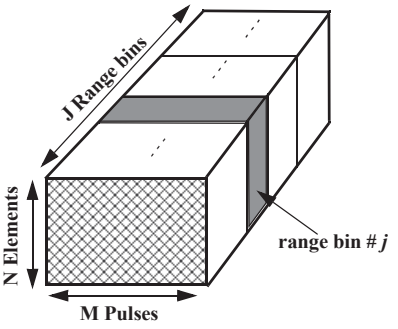


Figure 16.11. Space time data cube.

$$x(t) = e^{j2\pi f_0 t}. \qquad \textbf{Eq. (16.84)}$$

f_0 is the radar operating frequency. It follows that

$$\mathbf{Y}_j = \begin{bmatrix} \mathbf{y}_{1j} \\ \mathbf{y}_{2j} \\ \\ \\ \mathbf{y}_{Nj} \end{bmatrix} = x(t_j) \begin{bmatrix} 1 \\ e^{\left(-j\frac{2\pi d}{\lambda}\right)\sin\theta} \\ e^{\left(-j\frac{2\pi d}{\lambda}\right)2\sin\theta} \\ \\ \\ e^{\left(-j\frac{2\pi d}{\lambda}\right)(N-1)\sin\theta} \end{bmatrix} = x(t_j)\mathbf{s}_s(\theta) \qquad \textbf{Eq. (16.85)}$$

where the $\mathbf{s}_s(\theta)$ is the spatial steering vector associated with the arrival angle θ. In this notation, the subscript s is used to differentiate the spatial steering vector from the temporal steering vector, which will be defined later.

Next, consider the Doppler effects due to the target relative motion to the radar line of sight. In this case, the returned signal at sensor-1 due to M pulses is given by

$$\begin{bmatrix} y_1(t) \\ y_1(t-T) \\ \\ \\ y_1(t-(M-1)T) \end{bmatrix} = x(t) \begin{bmatrix} 1 \\ e^{-j2\pi(f_d)} \\ e^{-j2\pi(2f_d)} \\ \\ \\ e^{-j2\pi((M-1)f_d)} \end{bmatrix} = x(t)\mathbf{s}_t(f_d) \qquad \textbf{Eq. (16.86)}$$

where $\mathbf{s}_t(f_d)$ is the temporal steering vector for the Doppler shift f_d. Therefore, the composite return signal from the j^{th} range bin (i.e., time t_j) for a target whose Doppler frequency is f_d and is located at angle θ off the array boresight is

$$\mathbf{Y}_j = x(t_j)\{\mathbf{s}_s(\theta) \otimes \mathbf{s}_t(f_d)\} = x(t_j)\mathbf{s}_t \qquad \textbf{Eq. (16.87)}$$

where the symbol $\otimes$ indicates the Kronecker product and $\mathbf{s}_t = \mathbf{s}_s(\theta) \otimes \mathbf{s}_t(f_d)$.

16.5.2. Space Time Adaptive Processing

The space time adaptive beamformer is shown in Fig. 16.12. The output of the STAP beamformer is now given by,

$$\mathbf{Z}_j = \mathbf{W}\mathbf{Y}_j \qquad \textbf{Eq. (16.88)}$$

where $\mathbf{Y}_j$ was defined in the previous section and $\mathbf{W}$ is the adaptive weights matrix (see Fig. 16.11). As before, the input signal to the array is assumed to be made of the target returned signal, clutter and interference retuned signal, and thermal noise.

The total power output of the STAP beamformer is

$$P_{out} = E[\mathbf{Z}_j^* \mathbf{Z}_j] = P_{tgt}\left|\mathbf{W}^* \mathbf{s}_t\right|^2 + \mathbf{W}^* \mathbf{C} \mathbf{W} \qquad \textbf{Eq. (16.89)}$$

where $\mathbf{C}$ is the composite input signal covariance matrix and P_{tgt} is the desired target signal power. Note that the covariance matrix represents the combined target, clutter, noise, and interference signals; it is of size $MN \times MN$. It follows that the signal-to-interference plus noise ratio is

$$SINR_{out} = \frac{P_{tgt}\left|\mathbf{W}^* \mathbf{s}_t\right|^2}{\mathbf{W}^* \mathbf{C} \mathbf{W}}. \qquad \textbf{Eq. (16.90)}$$

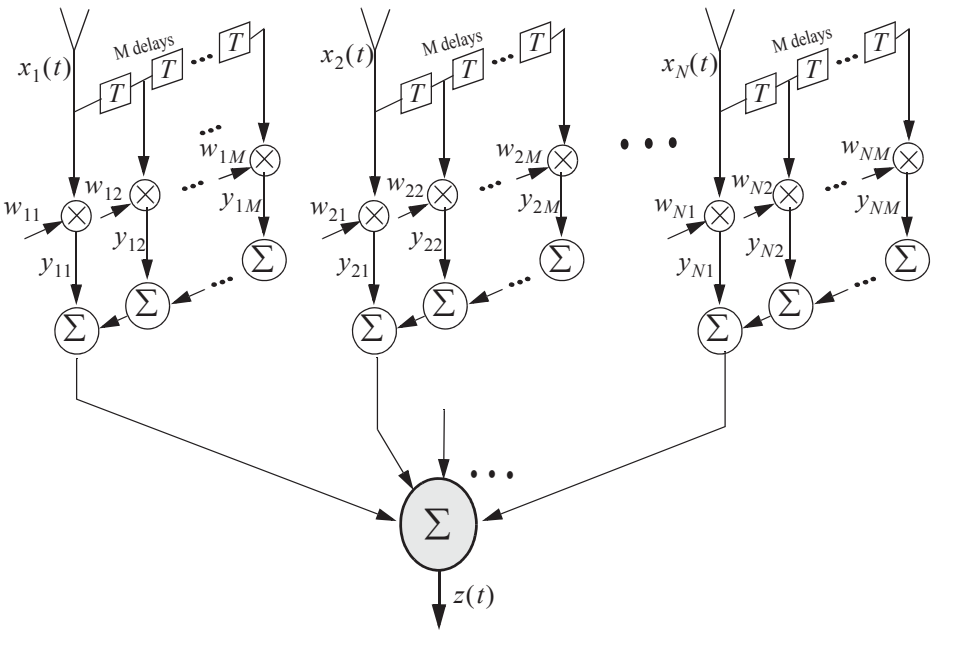


Figure 16.12. Illustration of a STAP array configuration. ($T = 1/f_r$)

Therefore, the optimal set of weights that maximize the ratio given in Eq. (16.90) is

$$\mathbf{W}_{opt} = \mathbf{C}^{-1}\mathbf{s}_t . \qquad \text{Eq. (16.91)}$$

Figures 16.13 through 16.18 demonstrate STAP processing. In these examples, one target and one jammer are present. Figures 16.13 and 16.14 show the combined target, jammer, and clutter returns. Figures 16.15 and 16.16 show the target and jammer return after removing the clutter ridge. Figures 16.17 and 16.18 show the target return after removing the jammer and clutter ridge returns. These figures can be reproduced using the MATLAB program *"run_stap.m"* and its associated MATLAB functions, which are listed in Appendix 16-A.

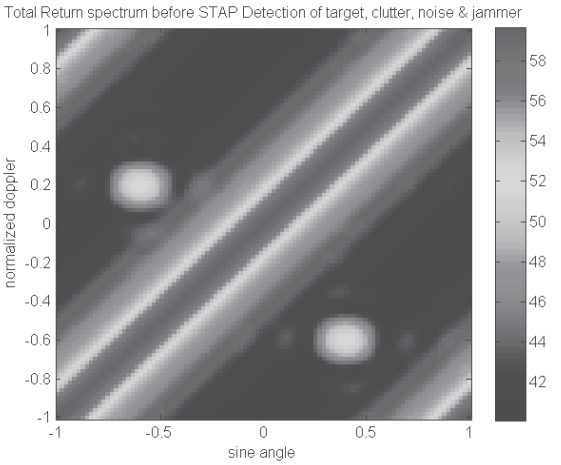


Figure 16.13. Output of STAP processor. The clutter dominates the scene.

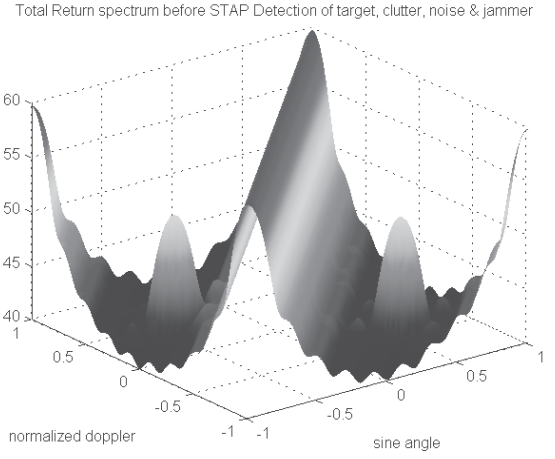


Figure 16.14. 3-D plot corresponding to Fig. 16.12.

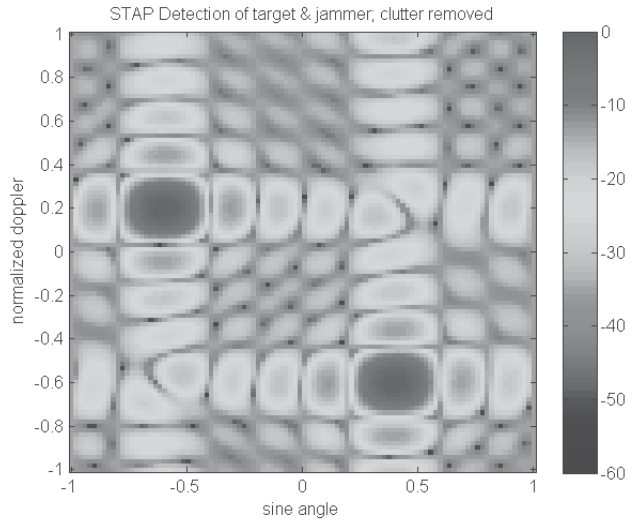


Figure 16.15. Output of STAP processor. Target and jammer returns; clutter ridge has been removed.

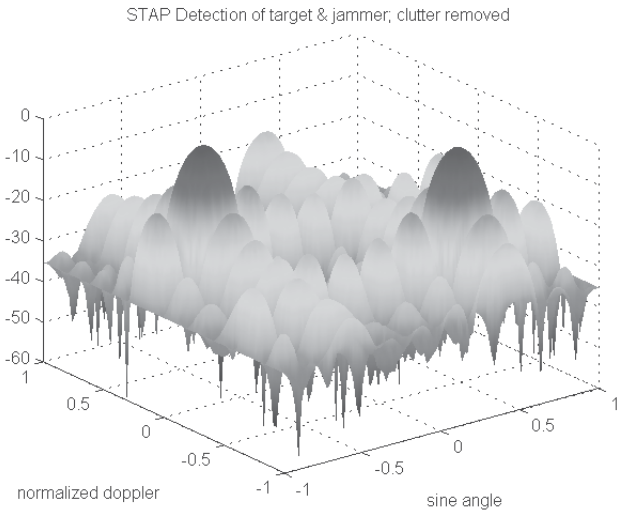


Figure 16.16. 3-D plot corresponding to Fig. 16.16.

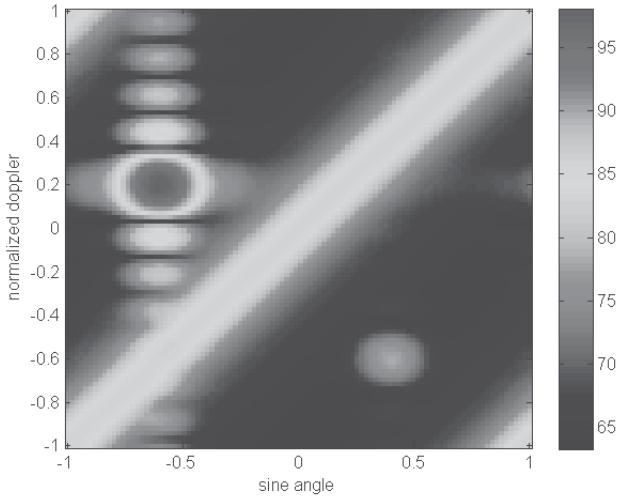


Figure 16.17. Output of STAP processor. Target only; jammer and clutter ridge returns have been removed.

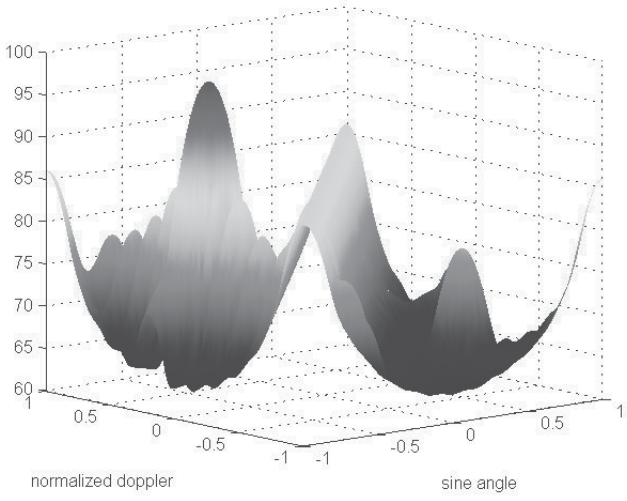


Figure 16.18. 3-D plot corresponding to Fig. 16.18.

Problems

16.1. Starting with Eq. (16.62), derive Eq. (16.63).

16.2. Compute the transient solution of the DE defined in Eq. (16.69).

16.3. Repeat the example in Section 16.1 for angle $\pi/4$ instead of $\pi/6$.

16.4. In Section 16.3, the MATLAB function *"adaptive_array_lms.m"* was developed to illustrate how linear arrays can adaptively place a null anywhere within the array's field of view. This code, however, assumed a single target (desired beam) and a single jammer (null). Extend this code (or develop your own) to account for multiple simultaneous desired beams (up to *N/2*) and multiple jammers (up to *N/2-1*) where *N* is the size of the array.

16.5. Building on the previous problem, in Chapter 15, the effect of having a limited number of bits to steer the main beam was demonstrated. Modify the code of the previous problem to include the effects of having a limited number of bits for phase shifting.

16.6. Figures 16.8a and 16.8b clearly demonstrate how the estimate of the covariance matrix impacts the quality of the adaptive null. In Section 16.3 (see Eq. (16.71)), a technique was described for estimating and improving the quality of the covariance matrix. Modify the MATLAB code *"adaptive_array_lms.m"* or develop your own code to implement Eq. (16.71). Briefly discuss how the quality of the adaptive null has been improved.

16.7. Develop a MATLAB code to implement the SLC canceler.

16.8. The MATLAB code *"run_stap.m"* used hard-coded (pre-determined) values for the SNR, CNR, and JSR power ratios. Modify this code, or write your own, to allow the user to change these values. Make a few runs with different combinations of these values and discuss your results.

16.9. Repeat the previous problem, where in this case, you will allow the number of elements in the array, *N*, to become a user-controlled variable. Run two cases one with low value for *N* (less than *10*) and one with a large value (more than *20*), briefly discuss your results.

Appendix 16-A: Chapter 16 MATLAB Code Listings

The MATLAB code provided in this chapter was designed as an academic standalone tool and is not adequate for other purposes. The code was written in a way to assist the reader in gaining a better understanding of the theory. The code was not developed, nor is it intended to be used as part of an open-loop or a closed-loop simulation of any kind. The MATLAB code found in this textbook can be downloaded from this book's web page on the CRC Press website. Simply use your favorite web browser, go to www.crcpress.com, and search for keyword *"Mahafza"* to locate this book's web page.

MATLAB Function "LMS.m" Listing

```
function Xout = LMS(Xin, D, B, mu, sigma, alpha)
%  This program was written by Stephen Robinson a senior radar
%  engineer at deciBel Research, Inc. in Huntsville, AL
%  Xin = data vector ; size = 1 x N
%  D = desired signal vector; size = 1 x N
%  N = number of data samples and of adaptive iterations
%  B = adaptive coefficients of Lht order fFIRfilter; size = 1 x L
%  L = order of adaptive system
%  mu = convergence parameter
%  sigma = input signal power estimate
%  alpha = exponential forgetting factor
N = size(Xin,2)
L = size(B,2)-1
px = B;
for k = 1:N
   px(1) = Xin(k);
   Xout(k) = sum(B.*px);
   E = D(k) - Xout(k);
   sigma = alpha*(px(1)^2) + (1 - alpha)*sigma;
   tmp = 2*mu/((L+1)*sigma);
   B = B + tmp*E*px;
   px(L+1:-1:2) = px(L:-1:1);
end; return
```

MATLAB Program "Fig16_4_5.m" Listing

```
% Figures 16.4 and 16.5
clc; close all;  clear all
N = 501;
mu = 0.1; % convergence parameter
Mu = num2str(mu);
L = 20; % FIR filter order
B = zeros(1,L+1); % FIR coefficients
sigma = 2; %Initial estimate for noise power
alpha = .100; % forgetting factor
Alpha =num2str(alpha);
k = 1:N;
noise = rand(1, length(k)) - .5; % Random noise
D = sqrt(2)*sin(2*pi*k/20);
X = D + sqrt(7)*noise;
Y = LMS(X, D, B, mu, sigma, alpha);
subplot(3,1,1)
```

```
plot(D,'linewidth',1.5);
xlim([0 501]); grid on;
ylabel('\bfDesired response');
title(['\mu = ',[Mu],  ' ,   \alpha = ',[Alpha]])
subplot(3,1,2)
plot(X,'linewidth',1);
xlim([0 501]); grid on;
ylabel('\bfCorrupted signal')
subplot(3,1,3)
plot(Y,'linewidth',1.5);
xlim([0 501]);
grid on; xlabel('\bftime in sec'); ylabel('\bfLMS output')
```

MATLAB Function "adaptive_array_lms.m" Listing

```
function adaptive_array_lms(N, dol,tagt_angle, jam_angle)
% This function implements the adaptive array LMS algorithm described in
% Section 16.6 of text.
% This function calls two other function
   % la_sampled_wave  and
   % linear_array_FFT
% Inputs
   % N        == size of linear array
   % dol      == array element spacing in lambda units
   % tgt_angle == targte angle (desired signal) in degrees
   % jam_angle == jammer angle (desired location of null) in degrees
% Outputs
   % This function will display the before and after normalized array
   % response in dB versus scan angle in degrees
clc; close all
mu = [0 0]; % noise mean value
sigma = [.21 .21; .21 .210]; % noise variance
% N = 19;
% dol = 0.5;
% tgt_angle = 0;
% jam_angle = 40;
sine_tgt_angle = sin(tagt_angle *pi/180);
sine_jam_angle = sin(jam_angle*pi/180);
al = la_sampled_wave(N, dol, sine_tgt_angle);
jl  = la_sampled_wave(N, dol, sine_jam_angle);
x = al + jl;
n = mvnrnd(mu,sigma,N);
jl = jl + complex(n(:,1),n(:,2));
Xl = jl * jl';
Cl = cov(Xl) + eye(N);
Wl = inv(Cl) * al;
[G, R, u, theta] = linear_array_FFT(Wl, dol);
[G1, R1, u, theta] = linear_array_FFT(x, dol);
u_deg = asin(u) *180/pi;
plot(u_deg, 10*log10((G/max(G)+eps))','linewidth', 1.5);
grid on
hold on
plot(u_deg, 10*log10((G1/max(G1)+eps)),'k:','linewidth', 2);
xlabel('\bfscan angle, \theta in degrees')
```

```
ylabel('\bf normalized array response')
legend('Adaptive array response','Input response')
JAM = num2str(jam_angle); title (['null placed at \theta = ',[JAM],'^0'])
```

MATLAB Function "la_sampled_wave.m" Listing

```
function s = la_sampled_wave(N, dol, sinbeta)
 k = 2*pi * dol * sinbeta;
for m = 1: N,
   s(m) = exp(j*(m-1)*k);
end
% Return a column vector, not a row vector
if size(s,1)==1,
   s = s.';
end
```

MATLAB Function "Linear_array_FFT.m" Listing

```
function [G, R, u, theta] = linear_array_FFT(a, dol);

Nelt = length(a);
ratio = 1;
if dol<=0.5,
   Nr = Nelt;
   dolr = dol;
else
   ratio = ceil(dol/0.5);
   Nr = Nelt * ratio;
   dolr = dol/ratio;
   atemp = a;
   a = zeros(1, Nr);
   a(1: ratio: Nr) = atemp(1: Nelt);
end
% use a value for NFFT that is at least 10 times that of N
% I borrowed this piece of code
nfft = 2^(ceil(log(10*Nr)/log(2)));
nfft = 65536;
A = fftshift(fft(a, nfft));
% Compute u = sin(theta)
u = [-nfft/2 : nfft/2-1] * (1/dolr/nfft);
% 'k' gives us the bounds of visible space
k = find(abs(u)<=1);
R = (abs(A(k))).^2;
u = u(k);
theta = asin(u);
% Gain patterns
Rbar = 0.5 * sum(R) / (nfft*dol);
G = R / Rbar;
```

MATLAB Program "run_stap.m" Listing

```
clear all; close all
sintheta_t1 = .4;
wd_t1 =-.6;
sintheta_t2 = -.6;
```

```
wd_t2 = .2;
[LL, sintheta, wd] = stap_std(sintheta_t1, wd_t1, sintheta_t2, wd_t2);
LL = LL / max(max(abs(LL)));
LL = max(LL, 1e-6);
figure (3)
imagesc(sintheta, wd, 10*log10(abs(LL)))
colorbar
title('STAP Detection of target & jammer; clutter removed');
set(gca,'ydir','normal'), xlabel('sine angle'), ylabel('normalized doppler')
figure (4)
surf(sintheta, wd, 10*log10(abs(LL)))
shading interp
title('STAP Detection of target & jammer; clutter removed');
set(gca,'ydir','normal'), xlabel('sine angle'), ylabel('normalized doppler')
[LL, sintheta, wd] = stap_smaa(sintheta_t1, wd_t1, sintheta_t2, wd_t2);
LL = LL / max(max(abs(LL)));
LL = max(LL, 1e-6);
figure
imagesc(sintheta, wd, 10*log10(abs(LL)))
colorbar
set(gca,'ydir','normal'), xlabel('sine angle'), ylabel('normalized doppler')
title('STAP Detection of target; jammer & clutter removed');
figure
surf(sintheta, wd, 10*log10(abs(LL)))
shading interp
set(gca,'ydir','normal'), xlabel('sine angle'), ylabel('normalized doppler')
title('SNR after SMAA STAP Detection of target, clutter, noise & jammer');
```

MATLAB Function "stap_std.m" Listing

```
function [LL, sintheta, wd] = stap_std(sintheta_t1, wd_t1, sintheta_t2, wd_t2);
do_plot = 1;
N = 10;        % Sensors
M = 12;        % Pulses
No = 250;      % k-th clutter bins (refers to fig. 5)
beta = 1;      % The way the clutter fills the angle Doppler
dol = 0.5;     % d over lambda
CNR = 30;   % dB Clutter to Noise Ratio
SNR = 10;   % dB Signal to Noise Ratio
JSR = 0;   % dB Jammer to Signal Ratio
% Set the noise power
sigma2_n = 1;
% Clutter power
sigma2_c = sigma2_n * 10^(CNR/10);
sigma_c = sqrt(sigma2_c);
% Target 1 power
sigma2_t1 = sigma2_n * 10^(SNR/10);
sigma_t1 = sqrt(sigma2_t1);
% Target 2 (Jammer) power
sigma2_t2 = sigma2_t1 * 10^(JSR/10);
sigma_t2 = sqrt(sigma2_t2);
% Ground clutter is the primary source of interference
sintheta = linspace(-1, 1, No);
phi = 2 * dol * sintheta;
```

```
wd = beta * phi;
Rc = zeros(N*M);
ac_all = zeros(N*M,1);
for k = 1: length(phi),
   ac = sigma_c * st_steering_vector(phi(k), N, beta*phi(k), M);  % Xc
   Rc = Rc + ac * ac';    % covarience matrix of target "1" ,  "'" --> conjugate transpose
   ac_all = ac_all + ac;  % "w" not optimized yet
end
Rc = Rc / length(phi);
% Noise signals decorrelate from pulse-to-pulse
% With this assumption, noise covariance matrix is
Rn = sigma2_n * eye(M*N);
% Target 1 covariance matrix
% at1 = st_steering_vector(sintheta_t1, N, wd_t1, M);
% Rt1 = sigma2_t1 * at1 * at1';
at1 = sigma_t1 * st_steering_vector(sintheta_t1, N, wd_t1, M); % Xj1
Rt1 = at1 * at1';   % covarience matrix of target "1"
at2 = sigma_t2 * st_steering_vector(sintheta_t2, N, wd_t2, M);
Rt2 = at2 * at2';  % covarience matrix of target "2" == jammer
% Total covariance matrix
R = Rc + Rn + Rt1 + Rt2;
% Unweighted spectrum of the total return from the beamformer
sintheta = linspace(-1, 1);
wd = beta * sintheta;
Pb = zeros(length(wd), length(sintheta));
for nn = 1: length(sintheta),
   for mm = 1: length(wd),
      a = st_steering_vector(sintheta(nn), N, wd(mm), M);
      Pb(mm, nn) = a' * R * a;
   end
end
if do_plot,
   % Display the total return spectrum
   figure (1)
   imagesc(sintheta, wd, 10*log10(abs(Pb)))
   colorbar
   title('Total Return spectrum before STAP Detection of target, clutter, noise & jammer');
   set(gca,'ydir','normal'), xlabel('sine angle'), ylabel('normalized doppler')
   figure (2)
   surf(sintheta, wd, 10*log10(abs(Pb)))
   shading interp, , xlabel('sine angle'), ylabel('normalized doppler')
   title('Total Return spectrum before STAP Detection of target, clutter, noise & jammer');
end
% Total covariance matrix
R = Rc + Rn + Rt1 + Rt2;
% Calculate optimal weights
Rc = (ac_all * ac_all') / length(phi);
Rinv = inv(Rc + Rn );  n
wopt = Rinv' * (at1 + at2 );
% Log-Likelihood Function
% Calculating the SNR and switching to the run_stap.m to execute the log part
sintheta = linspace(-1, 1);
wd = beta * sintheta;
LL = zeros(length(wd), length(sintheta));
```

```
for nn = 1: length(sintheta),
   for mm = 1: length(wd),
      a = st_steering_vector(sintheta(nn), N, wd(mm), M);
   %   LL(mm,nn) = abs( a' * Rinv * (at1+at2+ac_all) )^2 / ( a' * Rinv * a ); % Original by Keith
     LL(mm,nn) = abs(a' * Rinv * (at1+at2+ac_all)  )^2 / ( a' * (Rc + Rn ) * a ); % our expectation
   end
end
disp(size(a))
disp (size(Rinv))
```

MATLAB Function "stap_smaa.m" Listing

```
function [LL, sintheta, wd] = stap_smaa(sintheta_t1, wd_t1, sintheta_t2, wd_t2);
do_plot = 1;
N = 10; Na = 2*N-1;
M = 12;
No = 250;
beta = 1;
dol = 0.5;
CNR = 20; % dB
SNR = 0; % dB
JSR = 20; % dB
% Set the noise power
sigma2_n = 1;
% Clutter power
sigma2_c = sigma2_n * 10^(CNR/10);
sigma_c = sqrt(sigma2_c);
% Target 1 power
sigma2_t1 = sigma2_n * 10^(SNR/10);
sigma_t1 = sqrt(sigma2_t1);
% Target 2 (Jammer) power
sigma2_t2 = sigma2_t1 * 10^(JSR/10);
sigma_t2 = sqrt(sigma2_t2);
% Ground clutter is the primary source of interference
sintheta = linspace(-1, 1, No);
phi = 2 * dol * sintheta;
wd = beta * phi;
Rc = zeros(Na*M);
ac_all = zeros(Na*M,1);
for k = 1: length(phi),
   ac = sigma_c * smaa_st_steering_vector(phi(k), N, beta*phi(k), M);
   Rc = Rc + ac * ac';
   ac_all = ac_all + ac;
end
Rc = Rc / length(phi);
% Noise signals decorrelate from pulse-to-pulse
% With this assumption, noise covariance matrix is
Rn = sigma2_n * eye(M*Na);
% Target 1 covariance matrix
% at1 = smaa_st_steering_vector(sintheta_t1, N, wd_t1, M);
% Rt1 = sigma2_t1 * at1 * at1';
at1 = sigma_t1 * smaa_st_steering_vector(sintheta_t1, N, wd_t1, M);
Rt1 = at1 * at1';
% Target 1 covariance matrix
```

```
% at2 = smaa_st_steering_vector(sintheta_t2, N, wd_t2, M);
% Rt2 = sigma2_t2 * at2 * at2';
at2 = sigma_t2 * smaa_st_steering_vector(sintheta_t2, N, wd_t2, M);
Rt2 = at2 * at2';
% Total covariance matrix
R = Rc + Rn + Rt1 + Rt2;
% Unweighted spectrum of the total return from the beamformer
sintheta = linspace(-1, 1);
wd = beta * sintheta;
Pb = zeros(length(wd), length(sintheta));
for nn = 1: length(sintheta),
   for mm = 1: length(wd),
      a = smaa_st_steering_vector(sintheta(nn), N, wd(mm), M);
      Pb(mm, nn) = a' * R * a;
   end
end
if do_plot,
   % Display the total return spectrum
   figure (5)
   imagesc(sintheta, wd, 10*log10(abs(Pb)))
   set(gca,'ydir','normal'), xlabel('sine angle'), ylabel('normalized doppler')
   figure (6)
   surf(sintheta, wd, 10*log10(abs(Pb)))
   shading interp, xlabel('sine angle'), ylabel('normalized doppler')
end
% Calculate optimal weights
Rc = (ac_all * ac_all') / length(phi);
Rinv = inv(Rc + Rn);
wopt = Rinv * (at1 + at2);
% Log-Likelihood Function
sintheta = linspace(-1, 1);
wd = beta * sintheta;
LL = zeros(length(wd), length(sintheta));
for nn = 1: length(sintheta),
   for mm = 1: length(wd),
      a = smaa_st_steering_vector(sintheta(nn), N, wd(mm), M);
      LL(mm,nn) = abs( a' * Rinv * (at1+at2+ac_all) )^2 / ( a' * Rinv * a );
   end
end
```

MATLAB Function "st_steering_vector.m" Listing

```
function a = st_steering_vector(sintheta, N, wd, M)
a_N = exp(-j*pi*sintheta*[0:N-1]');
b_M = exp(-j*pi*wd      *[0:M-1]');
a = kron(b_M, a_N);
```

MATLAB Function "smaa_st_steering_vector.m" Listing

```
function a = smaa_st_steering_vector(sintheta, N, wd, M)
a_N = exp(-j*pi*sintheta*[-(N-1):+(N-1)]');
b_M = exp(-j*pi*wd      *[0:M-1]');
a_N = a_N .* ts_weighting(N);
a = kron(b_M, a_N);
```

Chapter 17

Target Tracking

Single Target Tracking

Tracking radar systems are used to measure the target's relative position in range, azimuth angle, elevation angle, and velocity. Then, by using and keeping track of these measured parameters the radar can predict their future values. Target tracking is important to military radars as well as to most civilian radars. In military radars, tracking is responsible for fire control and missile guidance; in fact, missile guidance is almost impossible without proper target tracking. Commercial radar systems, such as civilian airport traffic control radars, may utilize tracking as a means of controlling incoming and departing airplanes.

Tracking techniques can be divided into range/velocity tracking and angle tracking. It is also customary to distinguish between continuous single-target tracking radars and multi-target track-while-scan (TWS) radars. Tracking radars utilize pencil beam (very narrow) antenna patterns. It is for this reason that a separate search radar is needed to facilitate target acquisition by the tracker. Still, the tracking radar has to search the volume where the target's presence is suspected. For this purpose, tracking radars use special search patterns, such as helical, T.V. raster, cluster, and spiral patterns, to name a few.

17.1. Angle Tracking

Angle tracking is concerned with generating continuous measurements of the target's angular position in the azimuth and elevation coordinates. The accuracy of early-generation angle tracking radars depended heavily on the size of the pencil beam employed. Most modern radar systems achieve very fine angular measurements by utilizing monopulse tracking techniques.

Tracking radars use the angular deviation from the antenna main axis of the target within the beam to generate an error signal. This deviation is normally measured from the antenna's main axis. The resultant error signal describes how much the target has deviated from the beam main axis. Then, the beam position is continuously changed in an attempt to produce a zero error signal. If the radar beam is normal to the target (maximum gain), then the target angular position would be the same as that of the beam. In practice, this is rarely the case.

In order to be able to quickly change the beam position, the error signal needs to be a linear function of the deviation angle. It can be shown that this condition requires the beam's axis to be squinted by some angle (squint angle) off the antenna's main axis.

17.1.1. Sequential Lobing

Sequential lobing is one of the first tracking techniques that was utilized by the early generation of radar systems. Sequential lobing is often referred to as lobe switching or sequential switching. It has a tracking accuracy that is limited by the pencil beamwidth used and by the noise caused by either mechanical or electronic switching mechanisms. However, it is very simple to implement. The pencil beam used in sequential lobing must be symmetrical (equal azimuth and elevation beamwidths).

Tracking is achieved (in one coordinate) by continuously switching the pencil beam between two pre-determined symmetrical positions around the antenna's Line of Sight (LOS) axis. Hence, the name sequential lobing is adopted. The LOS is called the radar tracking axis, as illustrated in Fig. 17.1.

As the beam is switched between the two positions, the radar measures the returned signal levels. The difference between the two measured signal levels is used to compute the angular error signal. For example, when the target is tracked on the tracking axis, as the case in Fig. 17.1a, the voltage difference is zero. However, when the target is off the tracking axis, as in Fig. 17.1b, a nonzero error signal is produced. The sign of the voltage difference determines the direction in which the antenna must be moved. Keep in mind, the goal here is to make the voltage difference be equal to zero.

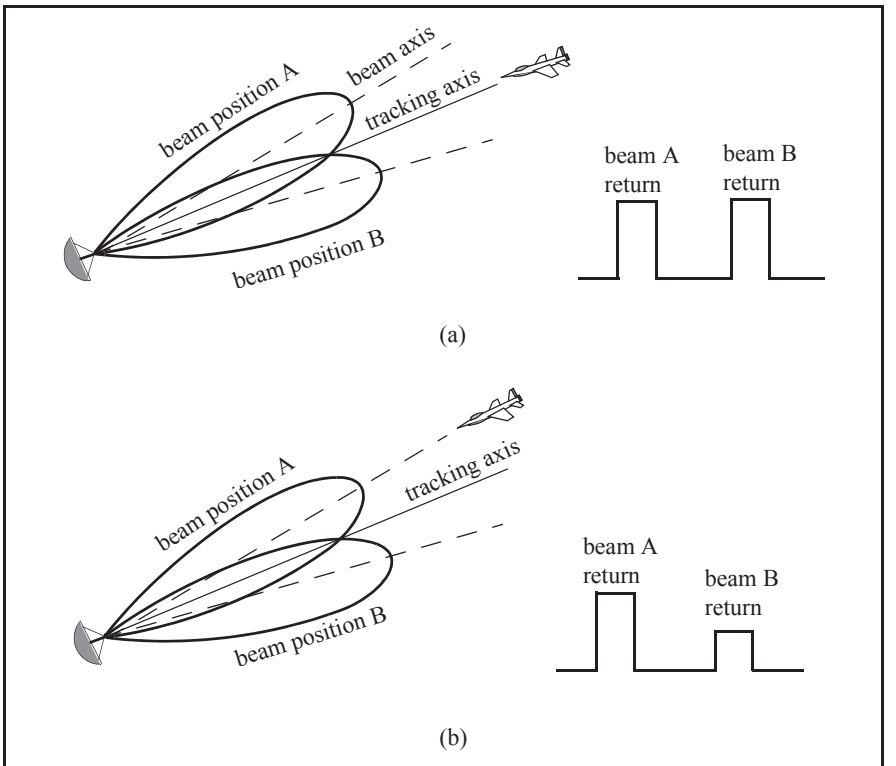


Figure 17.1. Sequential lobing. (a) Target is located on track axis. (b) Target is off track axis.

In order to obtain the angular error in the orthogonal coordinate, two more switching positions are required for that coordinate. Thus, tracking in two coordinates can be accomplished by using a cluster of four antennas (two for each coordinate) or by a cluster of five antennas. In the latter case, the middle antenna is used to transmit, while the other four are used to receive.

17.1.2. Conical Scan

Conical scan is a logical extension of sequential lobing where, in this case, the antenna is continuously rotated at an offset angle, or has a feed that is rotated about the antenna's main axis. Figure 17.2 shows a typical conical scan beam. The beam scan frequency, in radians per second, is denoted as ω_s. The angle between the antenna's LOS and the rotation axis is the squint angle φ. The antenna's beam position is continuously changed so that the target will always be on the tracking axis.

Figure 17.3 shows a simplified conical scan radar system. The envelope detector is used to extract the return signal amplitude, and the Automatic Gain Control (AGC) tries to hold the receiver output to a constant value. Since the AGC operates on large time constants, it can hold the average signal level constant and still preserve the signal rapid scan variation. It follows that the tracking error signals (azimuth and elevation) are functions of the target's RCS; they are functions of its angular position off the main beam axis.

In order to illustrate how conical scan tracking is achieved, we will first consider the case shown in Fig. 17.4. In this case, as the antenna rotates around the tracking axis, all target returns have the same amplitude (zero error signal). Thus, no further action is required.

Next, consider the case depicted by Fig. 17.5. Here, when the beam is at position B, returns from the target will have maximum amplitude, and when the antenna is at position A, returns from the target have minimum amplitude. Between those two positions, the amplitude of the target returns will vary between the maximum value at position B, and the minimum value at position A. In other words, Amplitude Modulation (AM) exists on top of the returned signal. This AM envelope corresponds to the relative position of the target within the beam. Thus, the extracted AM envelope can be used to derive a servo-control system in order to position the target on the tracking axis.

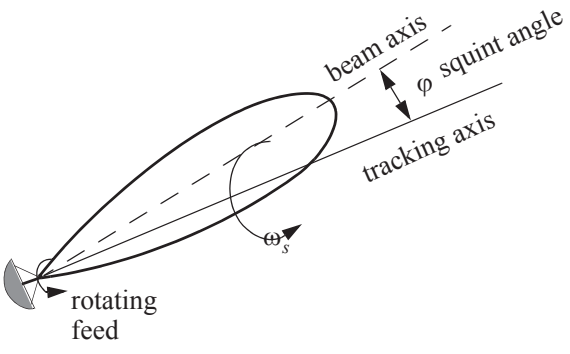


Figure 17.2. Conical scan beam.

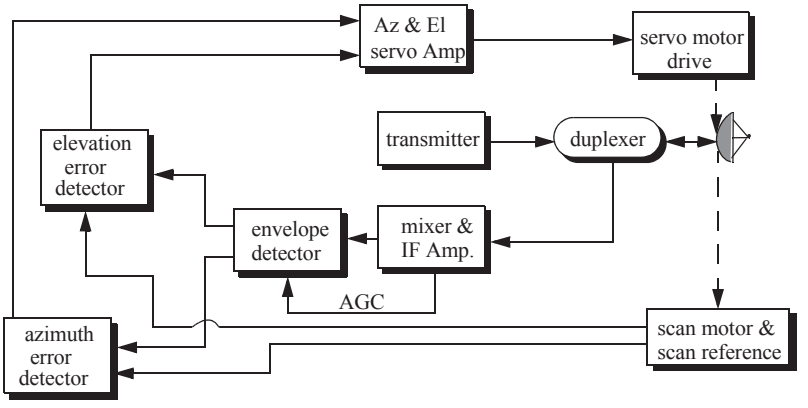


Figure 17.3. Simplified conical scan radar system.

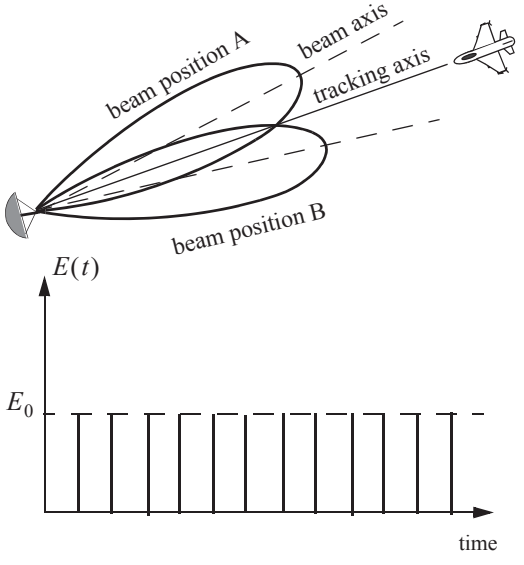


Figure 17.4. Error signal produced when the target is on the tracking axis for a conical scan.

Now, let us derive the error signal expression that is used to drive the servo-control system. Consider the top view of the beam axis location shown in Fig. 17.6. Assume that $t = 0$ is the starting beam position. The locations for maximum and minimum target returns are also identified. The quantity ε defines the distance between the target location and the antenna's tracking axis. It follows that the azimuth and elevation errors are, respectively, given by

$$\varepsilon_a = \varepsilon \sin\varphi \qquad \textbf{Eq. (17.1)}$$

$$\varepsilon_e = \varepsilon \cos\varphi\,. \qquad \textbf{Eq. (17.2)}$$

These are the error signals that the radar uses to align the tracking axis on the target.

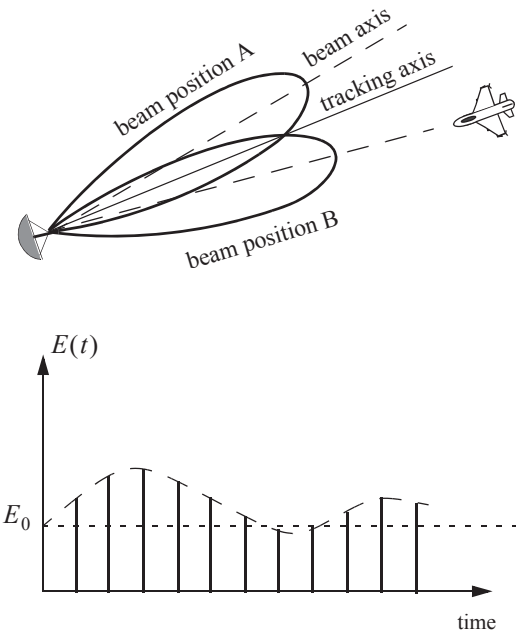


Figure 17.5. Error signal produced when the target is off the tracking axis for a conical scan.

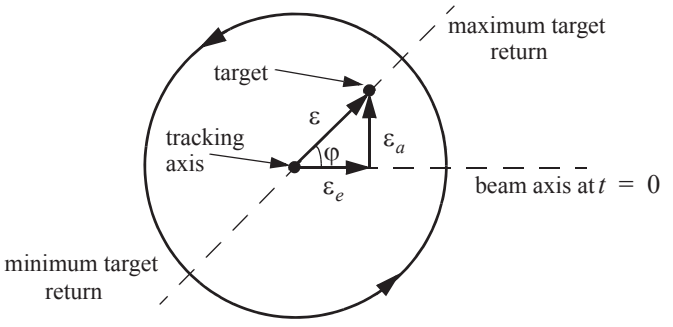


Figure 17.6. Top view of beam axis for a complete scan.

The AM signal $E(t)$ can then be written as

$$E(t) = E_0 \cos(\omega_s t - \varphi) = E_0 \varepsilon_e \cos\omega_s t + E_0 \varepsilon_a \sin\omega_s t \qquad \textbf{Eq. (17.3)}$$

where E_0 is a constant called the error slope, ω_s is the scan frequency in radians per second, and φ is the angle already defined. The scan reference is the signal that the radar generates to

keep track of the antenna's position around a complete path (scan). The elevation error signal is obtained by mixing the signal $E(t)$ with $\cos\omega_s t$ (the reference signal) followed by lowpass filtering. More precisely,

$$E_e(t) = E_0\cos(\omega_s t-\varphi)\cos\omega_s t = -\frac{1}{2}E_0\cos\varphi+\frac{1}{2}\cos(2\omega_s t-\varphi), \qquad \textbf{Eq. (17.4)}$$

and after lowpass filtering we get

$$E_e(t) = -\frac{1}{2}E_0\cos\varphi. \qquad \textbf{Eq. (17.5)}$$

Negative elevation error drives the antenna beam downward, while positive elevation error drives the antenna beam upward. Similarly, the azimuth error signal is obtained by multiplying $E(t)$ by $\sin\omega_s t$ followed by lowpass filtering. It follows that

$$E_a(t) = \frac{1}{2}E_0\sin\varphi. \qquad \textbf{Eq. (17.6)}$$

The antenna scan rate is limited by the scanning mechanism (mechanical or electronic), where electronic scanning is much faster and more accurate than mechanical scanning. In either case, the radar needs at least four target returns to be able to determine the target azimuth and elevation coordinates (two returns per coordinate). Therefore, the maximum conical scan rate is equal to one fourth of the PRF. Rates as high as 30 scans per second are commonly used.

The conical scan squint angle needs to be large enough so that a good error signal can be measured. However, due to the squint angle, the antenna gain in the direction of the tracking axis is less than maximum. Thus, when the target is in track (located on the tracking axis), the SNR suffers a loss equal to the drop in the antenna gain. This loss is known as the squint or crossover loss. The squint angle is normally chosen such that the two-way (transmit and receive) crossover loss is less than a few decibels.

17.2. Amplitude Comparison Monopulse

Amplitude comparison monopulse tracking is similar to lobing in the sense that four squinted beams are required to measure the target's angular position. The difference is that the four beams are generated simultaneously rather than sequentially. For this purpose, a special antenna feed is utilized such that the four beams are produced using a single pulse, hence the name "monopulse." Additionally, monopulse tracking is more accurate and is not susceptible to lobing anomalies, such as AM jamming and gain inversion ECM. Finally, in sequential and conical lobing, variations in the radar echoes degrade the tracking accuracy; however, this is not a problem for monopulse techniques since a single pulse is used to produce the error signals. Monopulse tracking radars can employ both antenna reflectors as well as phased array antennas.

Figure 17.7 show a typical monopulse antenna pattern. The four beams A, B, C, and D represent the four conical scan beam positions. Four feeds, mainly horns, are used to produce the monopulse antenna pattern. Amplitude monopulse processing requires that the four signals have the same phase and different amplitudes.

A good way to explain the concept of amplitude monopulse technique is to represent the target echo signal by a circle centered at the antenna's tracking axis, as illustrated by Fig. 17.8a,

where the four quadrants represent the four beams. In this case, the four horns receive an equal amount of energy, which indicates that the target is located on the antenna's tracking axis. However, when the target is off the tracking axis (Figs. 17.8b-d), an imbalance of energy occurs in the different beams. This imbalance of energy is used to generate an error signal that drives the servo-control system. Monopulse processing consists of computing a sum Σ and two difference Δ (azimuth and elevation) antenna patterns. Then by dividing a Δ channel voltage by the Σ channel voltage, the angle of the signal can be determined.

The radar continuously compares the amplitudes and phases of all beam returns to sense the amount of target displacement off the tracking axis. It is critical that the phases of the four signals be constant in both transmit and receive modes. For this purpose, either digital networks or microwave comparator circuitry are utilized. Figure 17.9 shows a block diagram for a typical microwave comparator, where the three receiver channels are declared as the sum channel, elevation angle difference channel, and azimuth angle difference channel.

To generate the elevation difference beam, one can use the beam difference (A-D) or (B-C). However, by first forming the sum patterns (A+B) and (D+C) and then computing the difference (A+B)-(D+C), we achieve a stronger elevation difference signal, Δ_{el}. Similarly, by first forming the sum patterns (A+D) and (B+C) and then computing the difference (A+D)-(B+C), a stronger azimuth difference signal, Δ_{az}, is produced.

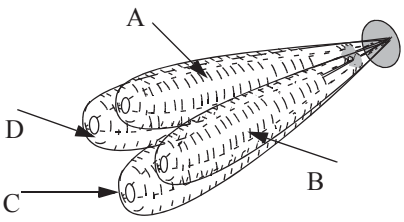


Figure 17.7. Monopulse antenna pattern.

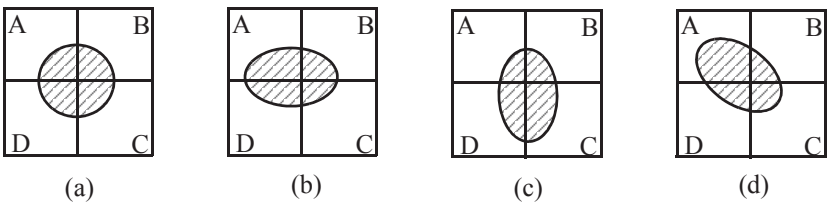


Figure 17.8. Illustration of monopulse concept. (a) Target is on the tracking axis. (b) - (d) Target is off the tracking axis.

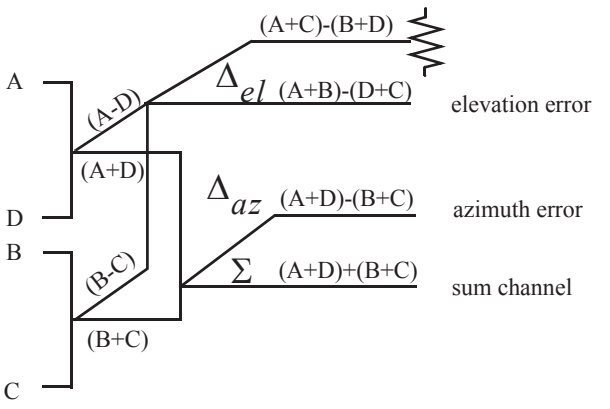


Figure 17.9. Monopulse comparator.

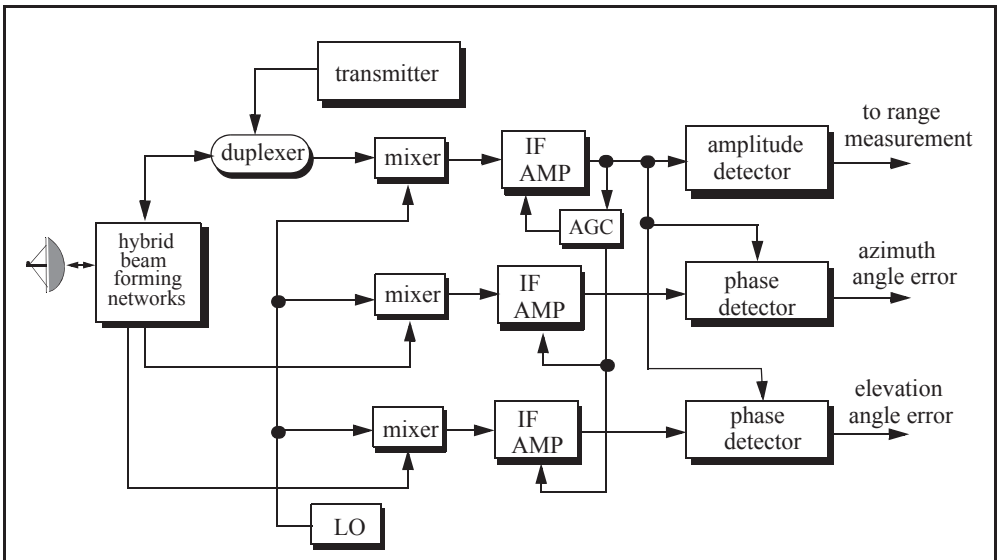


Figure 17.10. Simplified amplitude comparison monopulse radar block diagram.

A simplified monopulse radar block diagram is shown in Fig. 17.10. The sum channel is used for both transmit and receive. In the receive mode, the sum channel provides the phase reference for the other two difference channels. Range measurements can also be obtained from the sum channel. In order to illustrate how the sum and difference antenna patterns are formed, we will assume a $\sin\varphi/\varphi$ single-element antenna pattern and squint angle φ_0. The sum signal in one coordinate (azimuth or elevation) is then given by

$$\Sigma(\varphi) = \frac{\sin(\varphi-\varphi_0)}{(\varphi-\varphi_0)} + \frac{\sin(\varphi+\varphi_0)}{(\varphi+\varphi_0)}, \qquad \text{Eq. (17.7)}$$

and a difference signal in the same coordinate is

$$\Delta(\varphi) = \frac{\sin(\varphi - \varphi_0)}{(\varphi - \varphi_0)} - \frac{\sin(\varphi + \varphi_0)}{(\varphi + \varphi_0)}. \quad \textbf{Eq. (17.8)}$$

MATLAB Function "mono_pulse.m"

The function "mono_pulse.m" implements Eqs. (17.7) and (17.8). Its output includes plots of the sum and difference antenna patterns as well as the difference-to-sum ratio. The syntax is as follows:

mono_pulse (phi0)

where *phi0* is the squint angle in radians.

Figure 17.11 (a-c) shows the corresponding plots for the sum and difference patterns for $\varphi_0 = 0.15$ radians. Fig. 17.12 (a-c) is similar to Fig. 17.11, except in this case $\varphi_0 = 0.75$ radians. Clearly, the sum and difference patterns depend heavily on the squint angle. Using a relatively small squint angle produces a better sum pattern than that resulting from a larger angle. Additionally, the difference pattern slope is steeper for the small squint angle.

The difference channels give us an indication of whether the target is on or off the tracking axis. However, this signal amplitude depends not only on the target angular position, but also on the target's range and RCS. For this reason, the ratio Δ/Σ (delta over sum) can be used to accurately estimate the error angle that only depends on the target's angular position.

Let us now address how the error signals are computed. First, consider the azimuth error signal. Define the signals S_1 and S_2 as

$$S_1 = A + D \quad \textbf{Eq. (17.9)}$$

$$S_2 = B + C. \quad \textbf{Eq. (17.10)}$$

The sum signal is $\Sigma = S_1 + S_2$, and the azimuth difference signal is $\Delta_{az} = S_1 - S_2$. If $S_1 \geq S_2$, then both channels have the same phase $0°$ (since the sum channel is used for phase reference). Alternatively, if $S_1 < S_2$, then the two channels are $180°$ out of phase. Similar analysis can be done for the elevation channel, where in this case $S_1 = A + B$ and $S_2 = D + C$. Thus, the error signal output is

$$\varepsilon_\varphi = \frac{|\Delta|}{|\Sigma|} \cos\xi \quad \textbf{Eq. (17.11)}$$

where ξ is the phase angle between the sum and difference channels and it is equal to $0°$ or $180°$. More precisely, if $\xi = 0$, then the target is on the tracking axis; otherwise it is off the tracking axis. Figure 17.13 (a,b) shows a plot for the ratio Δ/Σ for the monopulse radar whose sum and difference patterns are in Figs. 17.11 and 17.12.

17.3. Phase Comparison Monopulse

Phase comparison monopulse is similar to amplitude comparison monopulse in the sense that the target angular coordinates are extracted from one sum and two difference channels. The main difference is that the four signals produced in amplitude comparison monopulse will have similar phases but different amplitudes; however, in phase comparison monopulse, the signals

have the same amplitude and different phases. Phase comparison monopulse tracking radars use a minimum of a two-element array antenna for each coordinate (azimuth and elevation), as illustrated in Fig. 17.14. A phase error signal (for each coordinate) is computed from the phase difference between the signals generated in the antenna elements.

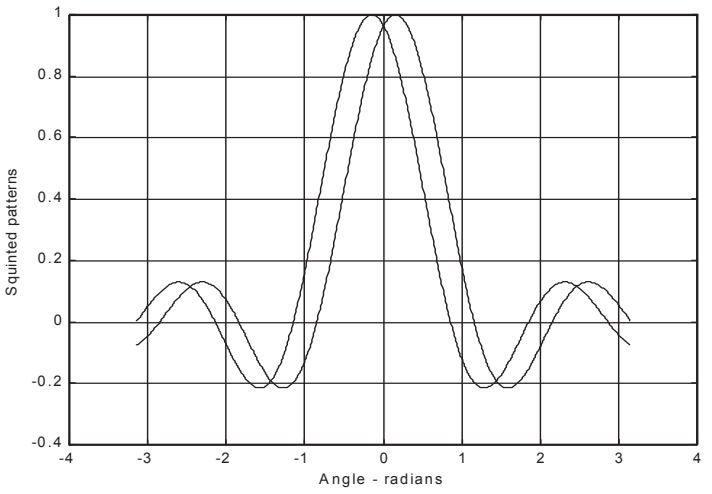


Figure 17.11a. Two squinted patterns. Squint angle is $\varphi_0 = 0.15$ radians.

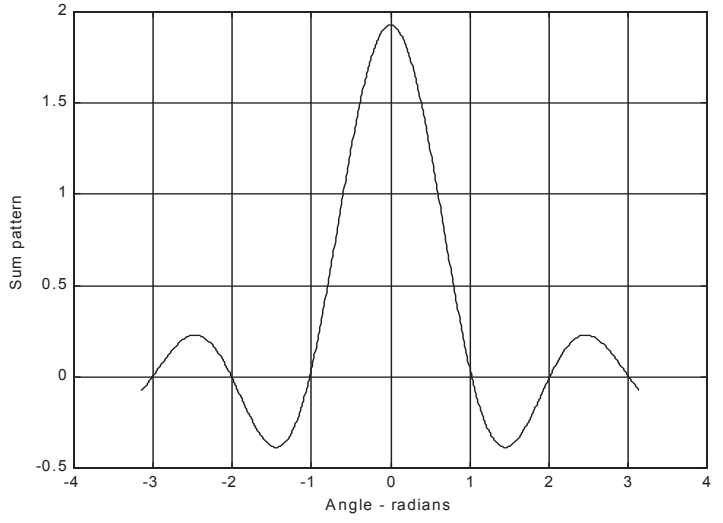


Figure 17.11b. Sum pattern corresponding to Fig. 17.11a.

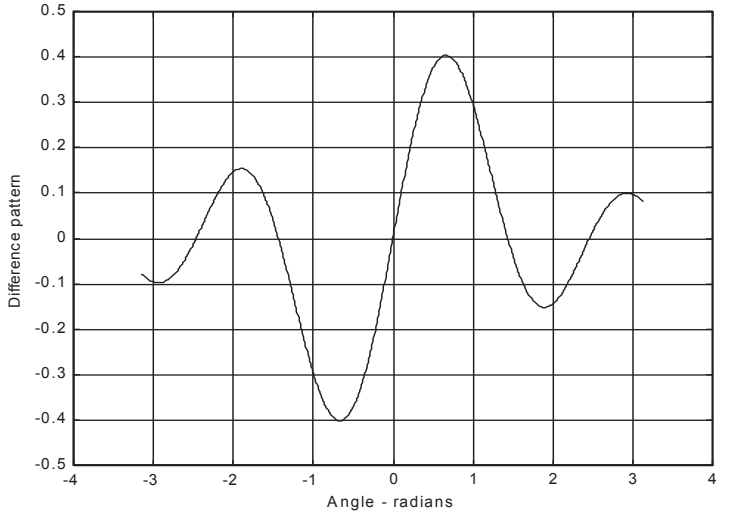


Figure 17.11c. Difference pattern corresponding to Fig. 17.11a.

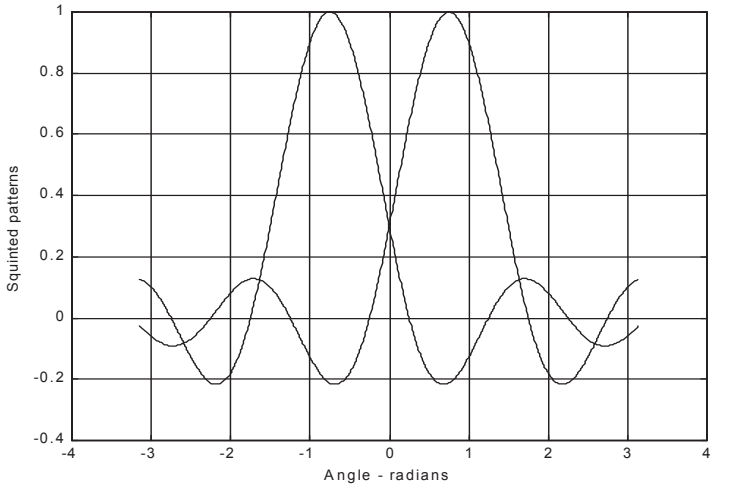


Figure 17.12a. Two squinted patterns. Squint angle is $\varphi_0 = 0.75$ radians.

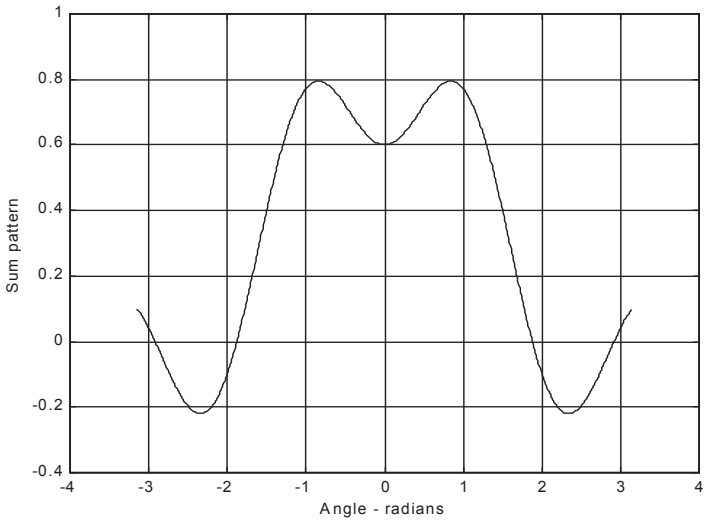


Figure 17.12b. Sum pattern corresponding to Fig. 17.12a.

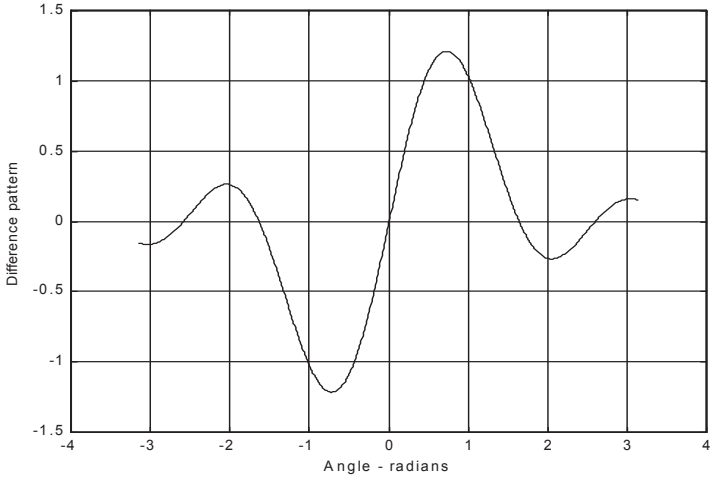


Figure 17.12c. Difference pattern corresponding to Fig. 17.12a.

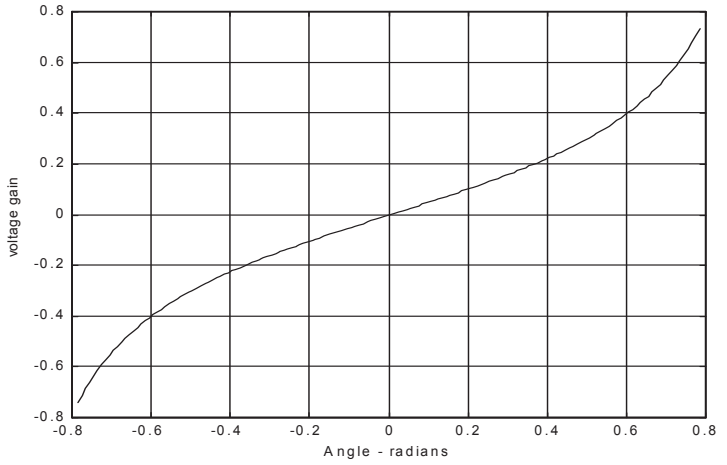


Figure 17.13a. Difference-to-sum ratio corresponding to Fig. 17.11a.

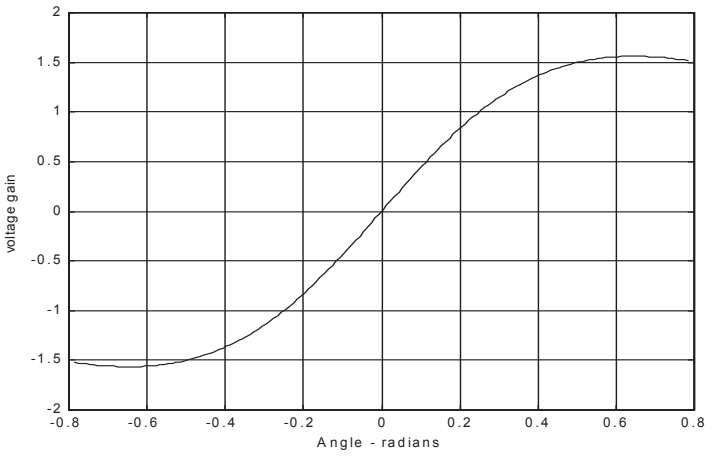


Figure 17.13b. Difference-to-sum ratio corresponding to Fig. 17.12a.

Consider Fig. 17.14; since the angle α is equal to $\varphi + \pi/2$, it follows that

$$R_1^2 = R^2 + \left(\frac{d}{2}\right)^2 - 2\frac{d}{2}R\cos\left(\varphi + \frac{\pi}{2}\right) = R^2 + \frac{d^2}{4} - dR\sin\varphi, \qquad \textbf{Eq. (17.12)}$$

and since $d \ll R$, we can use the binomial series expansion to get

$$R_1 \approx R\left(1 + \frac{d}{2R}\sin\varphi\right). \qquad \textbf{Eq. (17.13)}$$

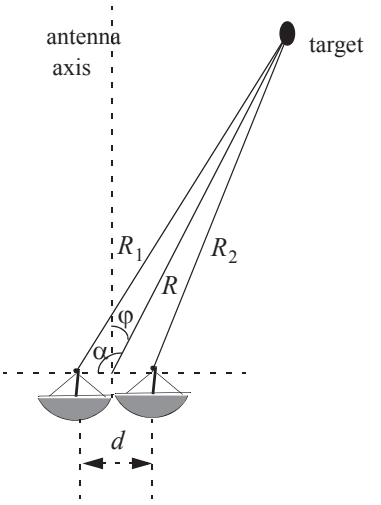


Figure 17.14. Single coordinate phase comparison monopulse antenna.

Similarly,

$$R_2 \approx R\left(1-\frac{d}{2R}\sin\varphi\right). \qquad \textbf{Eq. (17.14)}$$

The phase difference between the two elements is then given by

$$\phi = \frac{2\pi}{\lambda}(R_1 - R_2) = \frac{2\pi}{\lambda}d\sin\varphi \qquad \textbf{Eq. (17.15)}$$

where λ is the wavelength. The phase difference ϕ is used to determine the angular target location. Note that if $\phi = 0$, then the target would be on the antenna's main axis. The problem with this phase comparison monopulse technique is that it is quite difficult to maintain a stable measurement of the off-boresight angle φ, which causes serious performance degradation. This problem can be overcome by implementing a phase comparison monopulse system as illustrated in Fig. 17.15.

The (single coordinate) sum and difference signals are, respectively, given by

$$\Sigma(\varphi) = S_1 + S_2 \qquad \textbf{Eq. (17.16)}$$

$$\Delta(\varphi) = S_1 - S_2 \qquad \textbf{Eq. (17.17)}$$

where the S_1 and S_2 are the signals in the two elements. Now, since S_1 and S_2 have similar amplitude and are different in phase by ϕ, we can write

$$S_1 = S_2 e^{-j\phi}. \qquad \textbf{Eq. (17.18)}$$

It follows that

$$\Delta(\varphi) = S_2(1 - e^{-j\phi}) \qquad \textbf{Eq. (17.19)}$$

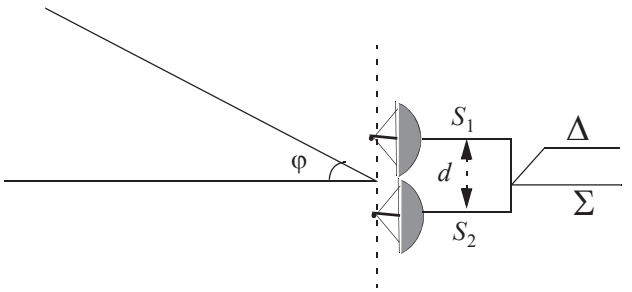


Figure 17.15. Single coordinate phase monopulse antenna, with sum and difference channels.

$$\Sigma(\varphi) = S_2(1 + e^{-j\phi}) . \qquad \text{Eq. (17.20)}$$

The phase error signal is computed from the ratio Δ/Σ. More precisely,

$$\frac{\Delta}{\Sigma} = \frac{1 - e^{-j\phi}}{1 + e^{-j\phi}} = j\tan\left(\frac{\phi}{2}\right), \qquad \text{Eq. (17.21)}$$

which is purely imaginary. The modulus of the error signal is then given by

$$\frac{|\Delta|}{|\Sigma|} = \tan\left(\frac{\phi}{2}\right). \qquad \text{Eq. (17.22)}$$

This kind of phase comparison monopulse tracker is often called the half-angle tracker.

17.4. Range Tracking

Target range is measured by estimating the round-trip delay of the transmitted pulses. The process of continuously estimating the range of a moving target is known as range tracking. Since the range to a moving target is changing with time, the range tracker must be constantly adjusted to keep the target locked in range. This can be accomplished using a split gate system, where two range gates (early and late) are utilized. The concept of split gate tracking is illustrated in Fig. 17.16, where a sketch of a typical pulsed radar echo is shown in the figure. The early gate opens at the anticipated starting time of the radar echo and lasts for half its duration. The late gate opens at the center and closes at the end of the echo signal. For this purpose, good estimates of the echo duration and the pulse center time must be reported to the range tracker so that the early and late gates can be placed properly at the start and center times of the expected echo. This reporting process is widely known as the "designation process."

The early gate produces positive voltage output while the late gate produces negative voltage output. The outputs of the early and late gates are subtracted, and the difference signal is fed into an integrator to generate an error signal. If both gates are placed properly in time, the integrator output will be equal to zero. Alternatively, when the gates are not timed properly, the integrator output is not zero, which gives an indication that the gates must be moved in time, left or right, depending on the sign of the integrator output.

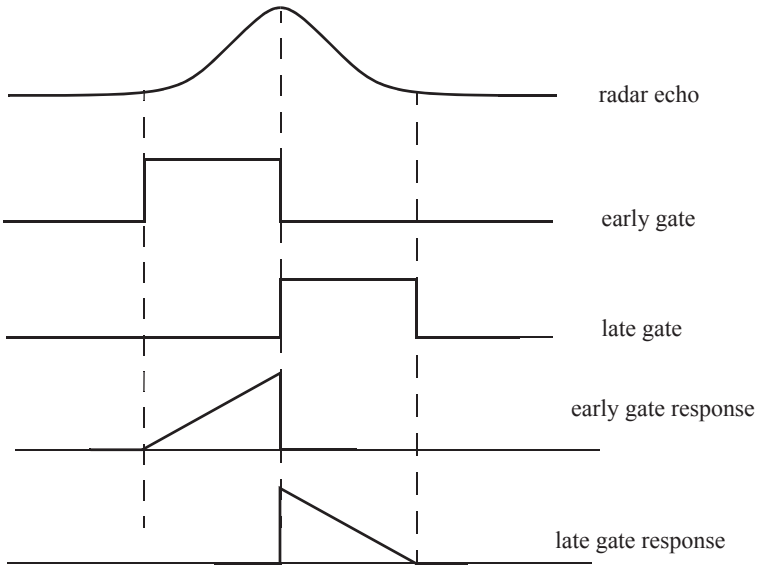


Figure 17.16. Illustration of split-range gate.

Multiple Target Tracking

Track-while-scan radar systems sample each target once per scan interval, and use sophisticated smoothing and prediction filters to estimate the target parameters between scans. To this end, the Kalman filter and the Alpha-Beta-Gamma ($\alpha\beta\gamma$) filter are commonly used. Once a particular target is detected, the radar may transmit up to a few pulses to verify the target parameters, before it establishes a track file for that target. Target position, velocity, and acceleration comprise the major components of the data maintained by a track file.

The principles of recursive tracking and prediction filters are presented in this part. First, an overview of state representation for Linear Time Invariant (LTI) systems is discussed. Then, second- and third-order one-dimensional fixed-gain polynomial filter trackers are developed. These filters are, respectively, known as the $\alpha\beta$ and $\alpha\beta\gamma$ filters (also known as the g-h and g-h-k filters). Finally, the equations for an n-dimensional multi-state Kalman filter is introduced and analyzed. As a matter of notation, lower case letters, with an underbar, are used.

17.5. Track-While-Scan (TWS)

Modern radar systems are designed to perform multi-function operations, such as detection, tracking, and discrimination. With the aid of sophisticated computer systems, multi-function radars are capable of simultaneously tracking many targets. In this case, each target is sampled once (mainly range and angular position) during a dwell interval (scan). Then, by using smoothing and prediction techniques, future samples can be estimated. Radar systems that can perform multi-tasking and multi-target tracking are known as Track-While-Scan (TWS) radars.

Once a TWS radar detects a new target, it initiates a separate track file for that detection; this ensures that sequential detections from that target are processed together to estimate the target's future parameters. Position, velocity, and acceleration comprise the main components of the track file. Typically, at least one other confirmation detection (verify detection) is required before the track file is established.

Unlike single target tracking systems, TWS radars must decide whether each detection (observation) belongs to a new target or belongs to a target that has been detected in earlier scans. And in order to accomplish this task, TWS radar systems utilize correlation and association algorithms. In the correlation process, each new detection is correlated with all previous detections in order to avoid establishing redundant tracks. If a certain detection correlates with more than one track, then a pre-determined set of association rules are exercised so that the detection is assigned to the proper track. A simplified TWS data processing block diagram is shown in Fig. 17.17.

Choosing a suitable tracking coordinate system is the first problem a TWS radar has to confront. It is desirable that a fixed reference of an inertial coordinate system be adopted. The radar measurements consist of target range, velocity, azimuth angle, and elevation angle. The TWS system places a gate around the target position and attempts to track the signal within this gate. The gate dimensions are normally azimuth, elevation, and range. Because of the uncertainty associated with the exact target position during the initial detections, a gate has to be large enough so that targets do not move appreciably from scan to scan; more precisely, targets must stay within the gate boundary during successive scans. After the target has been observed for several scans, the size of the gate is reduced considerably.

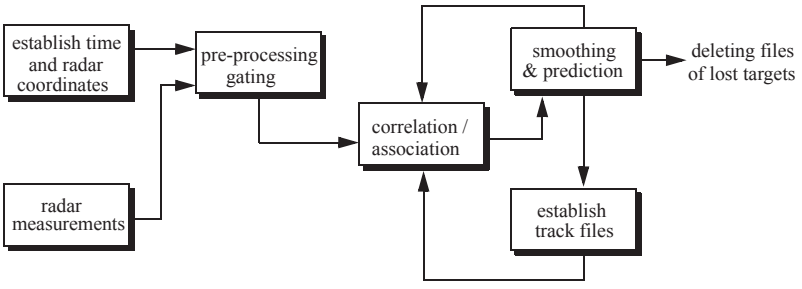


Figure. 9.17. Simplified block diagram of TWS data processing.

Gating is used to decide whether an observation is assigned to an existing track file, or to a new track file (new detection). Gating algorithms are normally based on computing a statistical error distance between a measured and an estimated radar observation. For each track file, an upper bound for this error distance is normally set. If the computed difference for a certain radar observation is less than the maximum error distance of a given track file, then the observation is assigned to that track.

All observations that have an error distance less than the maximum distance of a given track are said to correlate with that track. For each observation that does not correlate with any existing tracks, a new track file is established accordingly. Since new detections (measurements) are compared to all existing track files, a track file may then correlate with no observations or with one or more observations. The correlation between observations and all existing track files is identified using a correlation matrix. Rows of the correlation matrix represent radar observations, while columns represent track files. In cases where several observations correlate with more than one track file, a set of pre-determined association rules can be utilized so that a single observation is assigned to a single track file.

17.6. State Variable Representation of an LTI System

A linear time invariant system (continuous or discrete) can be described mathematically using three variables. They are the input, output, and the state variables. In this representation, any LTI system has observable or measurable objects (abstracts). For example, in the case of a radar system, range may be an object measured or observed by the radar tracking filter. States can be derived in many different ways. For the scope of this book, states of an object or an abstract are the components of the vector that contains the object and its time derivatives. For example, a third-order one-dimensional (in this case range) state vector representing range can be given by

$$\mathbf{x} = \begin{bmatrix} R \\ \dot{R} \\ \ddot{R} \end{bmatrix} \qquad \text{Eq. (17.23)}$$

where R, $\dot{R}$, and $\ddot{R}$ are, respectively, the range measurement, range rate (velocity), and acceleration. The state vector defined in Eq. (17.23) can be representative of continuous or discrete states. In this book, the emphasis is on discrete time representation, since most radar signal processing is executed using digital computers. For this purpose, an n-dimensional state vector has the following form:

$$\mathbf{x} = \left[x_1\ \dot{x}_1\ \ldots\ x_2\ \dot{x}_2\ \ldots\ x_n\ \dot{x}_n\ \ldots\right]^t \qquad \textbf{Eq. (17.24)}$$

where the superscript indicates the transpose operation.

The LTI system of interest can be represented using the following state equations:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{w}(t) \qquad \textbf{Eq. (17.25)}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{w}(t) \qquad \textbf{Eq. (17.26)}$$

where $\dot{\mathbf{x}}$ is the value of the $n \times 1$ state vector; $\mathbf{y}$ is the value of the $p \times 1$ output vector; $\mathbf{w}$ is the value of the $m \times 1$ input vector; $\mathbf{A}$ is an $n \times n$ matrix; $\mathbf{B}$ is an $n \times m$ matrix; $\mathbf{C}$ is $p \times n$ matrix; and $\mathbf{D}$ is an $p \times m$ matrix. The homogeneous solution (i.e., $\mathbf{w} = \mathbf{0}$) to this linear system, assuming known initial condition $\mathbf{x}(0)$ at time t_0, has the form

$$\mathbf{x}(t) = \Phi(t - t_0)\mathbf{x}(t - t_0)\,. \qquad \textbf{Eq. (17.27)}$$

The matrix Φ is known as the state transition matrix, or fundamental matrix, and is equal to

$$\Phi(t - t_0) = e^{\mathbf{A}(t - t_0)}\,. \qquad \textbf{Eq. (17.28)}$$

Eq. (17.28) can be expressed in series format as

$$\Phi(t - t_0)\big|_{t_0 = 0} = e^{\mathbf{A}(t)} = \mathbf{I} + \mathbf{A}t + \mathbf{A}^2\frac{t^2}{2!} + \ldots = \sum_{k=0}^{\infty}\mathbf{A}^k\ \frac{t^k}{k!} \qquad \textbf{Eq. (17.29)}$$

where $\mathbf{I}$ is the identity matrix.

Example:

Compute the state transition matrix for an LTI system when

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -0.5 & -1 \end{bmatrix}.$$

Solution:

The state transition matrix can be computed using Eq. (17.29). For this purpose, compute $\mathbf{A}^2$ *and* $\mathbf{A}^3$ *…. It follows*

$$\mathbf{A}^2 = \begin{bmatrix} -\frac{1}{2} & -1 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad \mathbf{A}^3 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{4} & 0 \end{bmatrix} \qquad \ldots$$

Therefore,

$$\Phi = \begin{bmatrix} 1 + 0t - \frac{\frac{1}{2}t^2}{2!} + \frac{\frac{1}{2}t^3}{3!} + \ldots & 0 + t - \frac{t^2}{2!} + \frac{\frac{1}{2}t^3}{3!} + \ldots \\ 0 - \frac{1}{2}t + \frac{\frac{1}{2}t^2}{2!} - \frac{\frac{1}{4}t^3}{3!} + \ldots & 1 - t + \frac{\frac{1}{2}t^2}{2!} + \frac{0t^3}{3!} + \ldots \end{bmatrix}.$$

The state transition matrix has the following properties (the proof is left as an exercise):

1. *Derivative property*

$$\frac{\partial}{\partial t}\Phi(t - t_0) = \mathbf{A}\Phi(t - t_0) \qquad \text{Eq. (17.30)}$$

2. *Identity property*

$$\Phi(t_0 - t_0) = \Phi(0) = \mathbf{I} \qquad \text{Eq. (17.31)}$$

3. *Initial value property*

$$\left.\frac{\partial}{\partial t}\Phi(t - t_0)\right|_{t = t_0} = \mathbf{A} \qquad \text{Eq. (17.32)}$$

4. *Transition property*

$$\Phi(t_2 - t_0) = \Phi(t_2 - t_1)\Phi(t_1 - t_0) \quad ; \; t_0 \le t_1 \le t_2 \qquad \text{Eq. (17.33)}$$

5. *Inverse property*

$$\Phi(t_0 - t_1) = \Phi^{-1}(t_1 - t_0) \qquad \text{Eq. (17.34)}$$

6. *Separation property*

$$\Phi(t_1 - t_0) = \Phi(t_1)\Phi^{-1}(t_0) \qquad \text{Eq. (17.35)}$$

The general solution to the system defined in Eq. (17.25) can be written as

$$\mathbf{x}(t) = \Phi(t - t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \Phi(t - \tau)\mathbf{B}\mathbf{w}(\tau)d\tau\,. \qquad \text{Eq. (17.36)}$$

The first term of the right-hand side of Eq. (17.36) represents the contribution from the system response to the initial condition. The second term is the contribution due to the driving force $\underline{w}$. By combining Eqs. (17.26) and (17.36), an expression for the output is computed as

$$\mathbf{y}(t) = \mathbf{C}e^{\mathbf{A}(t - t_0)}\mathbf{x}(t_0) + \int_{t_0}^{t} [\mathbf{C}e^{\mathbf{A}(t - \tau)}\mathbf{B} - \mathbf{D}\delta(t - \tau)]\mathbf{w}(\tau)d\tau\,. \qquad \text{Eq. (17.37)}$$

Note that the system impulse response is equal to $\mathbf{C}e^{\mathbf{A}t}\mathbf{B} - \mathbf{D}\delta(t)$.

The difference equations describing a discrete time system, equivalent to Eqs. (17.25) and (17.26), are

$$\mathbf{x}(n + 1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\mathbf{w}(n) \qquad \text{Eq. (17.38)}$$

$$\mathbf{y}(n) = \mathbf{C}\ \mathbf{x}(n) + \mathbf{D}\mathbf{w}(n) \qquad \text{Eq. (17.39)}$$

where n defines the discrete time nT and T is the sampling interval. All other vectors and matrices were defined earlier. The homogeneous solution to the system defined in Eq. (17.38), with initial condition $\mathbf{x}(n_0)$, is

$$\mathbf{x}(n) = \mathbf{A}^{n-n_0}\mathbf{x}(n_0). \qquad \text{Eq. (17.40)}$$

In this case, the state transition matrix is an $n \times n$ matrix given by

$$\Phi(n, n_0) = \Phi(n - n_0) = \mathbf{A}^{n-n_0}. \qquad \text{Eq. (17.41)}$$

The following is the list of properties associated with the discrete transition matrix:

$$\Phi(n + 1 - n_0) = \mathbf{A}\Phi(n - n_0) \qquad \text{Eq. (17.42)}$$

$$\Phi(n_0 - n_0) = \Phi(0) = \mathbf{I} \qquad \text{Eq. (17.43)}$$

$$\Phi(n_0 + 1 - n_0) = \Phi(1) = \mathbf{A} \qquad \text{Eq. (17.44)}$$

$$\Phi(n_2 - n_0) = \Phi(n_2 - n_1)\Phi(n_1 - n_0) \qquad \text{Eq. (17.45)}$$

$$\Phi(n_0 - n_1) = \Phi^{-1}(n_1 - n_0) \qquad \text{Eq. (17.46)}$$

$$\Phi(n_1 - n_0) = \Phi(n_1)\Phi^{-1}(n_0) \qquad \text{Eq. (17.47)}$$

The solution to the general case (i.e., non-homogeneous system) is given by

$$\mathbf{x}(n) = \Phi(n - n_0)\mathbf{x}(n_0) + \sum_{m=n_0}^{n-1} \Phi(n - m - 1)\mathbf{B}\mathbf{w}(m). \qquad \text{Eq. (17.48)}$$

It follows that the output is given by

$$\mathbf{y}(n) = \mathbf{C}\Phi(n - n_0)\mathbf{x}(n_0) + \sum_{m=n_0}^{n-1} \mathbf{C}\ \Phi(n - m - 1)\mathbf{B}\mathbf{w}(m) + \mathbf{D}\mathbf{w}(n) \qquad \text{Eq. (17.49)}$$

where the system impulse response is given by

$$\mathbf{h}(n) = \sum_{m=n_0}^{n-1} \mathbf{C}\ \Phi(n - m - 1)\mathbf{B}\underline{\delta}(m) + \mathbf{D}\underline{\delta}(n) \qquad \text{Eq. (17.50)}$$

where $\underline{\delta}$ is a vector.

Taking the Z-transform for Eqs. (17.38) and (17.39) yields

$$z\mathbf{x}(z) = \mathbf{A}\mathbf{x}(z) + \mathbf{B}\mathbf{w}(z) + z\mathbf{x}(0) \qquad \text{Eq. (17.51)}$$

$$\mathbf{y}(z) = \mathbf{C}\mathbf{x}(z) + \mathbf{D}\mathbf{w}(z). \qquad \text{Eq. (17.52)}$$

Manipulating Eqs. (17.51) and (17.52) yields

$$\mathbf{x}(z) = [z\mathbf{I} - \mathbf{A}]^{-1}\underline{B}\mathbf{w}(z) + [z\mathbf{I} - \mathbf{A}]^{-1}z\mathbf{x}(0) \qquad \text{Eq. (17.53)}$$

$$\mathbf{y}(z) = \{\mathbf{C}[z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B} + \mathbf{D}\}\mathbf{w}(z) + \mathbf{C}[z\mathbf{I} - \mathbf{A}]^{-1}z\mathbf{x}(0)\,. \qquad \text{Eq. (17.54)}$$

It follows that the state transition matrix is

$$\Phi(z) = z[z\mathbf{I} - \mathbf{A}]^{-1} = [\mathbf{I} - z^{-1}\mathbf{A}]^{-1}, \qquad \text{Eq. (17.55)}$$

and the system impulse response in the z-domain is

$$\mathbf{h}(z) = \mathbf{C}\Phi(z)z^{-1}\mathbf{B} + \mathbf{D}\,. \qquad \text{Eq. (17.56)}$$

17.7. The LTI System of Interest

For the purpose of establishing the framework necessary for the Kalman filter development, consider the LTI system shown in Fig. 17.18. This system (which is a special case of the system described in the previous section) can be described by the following first-order differential vector equations

$$\dot{\mathbf{x}}(t) = \mathbf{A}\ \mathbf{x}(t) + \mathbf{u}(t) \qquad \text{Eq. (17.57)}$$

$$\mathbf{y}(t) = \mathbf{G}\ \mathbf{x}(t) + \mathbf{v}(t) \qquad \text{Eq. (17.58)}$$

where $\mathbf{y}$ is the observable part of the system (i.e., output), $\mathbf{u}$ is a driving force, and $\mathbf{v}$ is the measurement noise. The matrices $\mathbf{A}$ and $\mathbf{G}$ vary depending on the system. The noise observation $\mathbf{v}$ is assumed to be uncorrelated. If the initial condition vector is $\mathbf{x}(t_0)$, then from Eq. (17.36) we get

$$\mathbf{x}(t) = \Phi(t - t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \Phi(t - \tau)\mathbf{u}(\tau)d\tau\,. \qquad \text{Eq. (17.59)}$$

The object (abstract) is observed only at discrete times determined by the system. These observation times are declared by discrete time nT where T is the sampling interval. Using the same notation adopted in the previous section, the discrete time representations of Eqs. (17.57) and (17.58) are

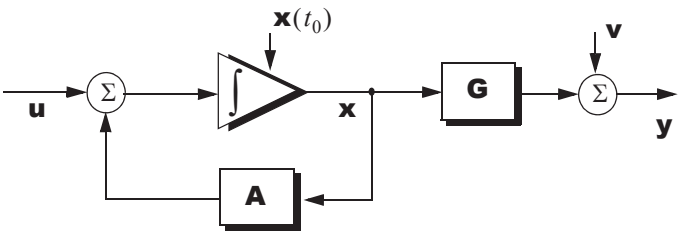


Figure 17.18. An LTI system.

$$\mathbf{x}(n) = \mathbf{A}\ \mathbf{x}(n-1) + \mathbf{u}(n) \qquad \text{Eq. (17.60)}$$

$$\mathbf{y}(n) = \mathbf{G}\mathbf{x}(n) + \mathbf{v}(n)\ . \qquad \text{Eq. (17.61)}$$

The homogeneous solution to this system is given in Eq. (17.27) for continuous time, and in Eq. (17.40) for discrete time.

The state transition matrix corresponding to this system can be obtained using Taylor series expansion of the vector $\mathbf{x}$. More precisely,

$$\begin{aligned} x &= x + T\dot{x} + \frac{T^2}{2!}\ddot{x} + \ldots \\ \dot{x} &= \dot{x} + T\ddot{x} + \ldots \\ \ddot{x} &= \ddot{x} + \ldots \end{aligned} \qquad \text{Eq. (17.62)}$$

It follows that the elements of the state transition matrix are defined by

$$\Phi[ij] = \left\{ \begin{matrix} T^{j-i} \div (j-i)! & 1 \le i, j \le n \\ 0 & j < i \end{matrix} \right\}. \qquad \text{Eq. (17.63)}$$

Using matrix notation, the state transition matrix is then given by

$$\Phi = \begin{bmatrix} 1 & T & \frac{T^2}{2!} & \ldots \\ 0 & 1 & T & \ldots \\ 0 & 0 & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{bmatrix}. \qquad \text{Eq. (17.64)}$$

The matrix given in Eq. (17.64) is often called the Newtonian matrix.

17.8. Fixed-Gain Tracking Filters

This class of filters (or estimators) is also known as "Fixed-Coefficient" filters. The most common examples of this class of filters are the $\alpha\beta$ and $\alpha\beta\gamma$ filters and their variations. The $\alpha\beta$ and $\alpha\beta\gamma$ trackers are one-dimensional second- and third-order filters, respectively. They are equivalent to special cases of the one-dimensional Kalman filter. The general structure of this class of estimators is similar to that of the Kalman filter.

The standard $\alpha\beta\gamma$ filter provides smoothed and predicted data for target position, velocity (Doppler), and acceleration. It is a polynomial predictor/corrector linear recursive filter. This filter can reconstruct position, velocity, and constant acceleration based on position measurements. The $\alpha\beta\gamma$ filter can also provide a smoothed (corrected) estimate of the present position, which can be used in guidance and fire control operations.

Notation:

For the purpose of the discussion presented in the remainder of this chapter, the following notation is adopted: $x(n|m)$ represents the estimate during the nth sampling interval, using all data up to and including the mth sampling interval; y_n is the nth measured value; and e_n is the nth residual (error).

The fixed-gain filter equation is given by

$$\mathbf{x}(n|n) = \Phi\underline{x}(n-1|n-1) + \mathbf{K}[y_n - \mathbf{G}\Phi\mathbf{x}(n-1|n-1)]. \quad \text{Eq. (17.65)}$$

Since the transition matrix assists in predicting the next state,

$$\mathbf{x}(n+1|n) = \Phi\underline{x}(n|n). \quad \text{Eq. (17.66)}$$

Substituting Eq. (17.66) into Eq. (17.65) yields

$$\mathbf{x}(n|n) = \mathbf{x}(n|n-1) + \mathbf{K}[y_n - \mathbf{G}\mathbf{x}(n|n-1)]. \quad \text{Eq. (17.67)}$$

The term enclosed within the brackets on the right-hand side of Eq. (17.67) is often called the residual (error), which is the difference between the measured input and predicted output. Eq. (17.67) means that the estimate of $\mathbf{x}(n)$ is the sum of the prediction and the weighted residual. The term $\mathbf{G}\mathbf{x}(n|n-1)$ represents the prediction state. In the case of the $\alpha\beta\gamma$ estimator, $\mathbf{G}$ is the row vector given by

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & \ldots \end{bmatrix}, \quad \text{Eq. (17.68)}$$

and the gain matrix $\mathbf{K}$ is given by

$$\mathbf{K} = \begin{bmatrix} \alpha \\ \beta/T \\ \gamma/T^2 \end{bmatrix}. \quad \text{Eq. (17.69)}$$

One of the main objectives of a tracking filter is to decrease the effect of the noise observation on the measurement. For this purpose, the noise covariance matrix is calculated. More precisely, the noise covariance matrix is

$$\mathbf{C}(n|n) = E[(\mathbf{x}(n|n))\mathbf{x}^t(n|n)] \quad ; \; y_n = v_n \quad \text{Eq. (17.70)}$$

where E indicates the expected value operator. Noise is assumed to be a zero mean random process with variance equal to σ_v^2. Additionally, noise measurements are assumed to be uncorrelated,

$$E[v_n v_m] = \begin{cases} \delta\sigma_v^2 & n = m \\ 0 & n \neq m \end{cases}. \quad \text{Eq. (17.71)}$$

Eq. (17.65) can be written as

$$\mathbf{x}(n|n) = \mathbf{A}\mathbf{x}(n-1|n-1) + \mathbf{K}y_n \quad \text{Eq. (17.72)}$$

where

$$\mathbf{A} = (\mathbf{I} - \mathbf{K}\mathbf{G})\Phi. \quad \text{Eq. (17.73)}$$

Substituting Eqs. (17.72) and (17.73) into Eq. (17.70) yields

$$\mathbf{C}(n|n) = E[(\mathbf{A}\mathbf{x}(n-1|n-1) + \mathbf{K}y_n)(\mathbf{A}\mathbf{x}(n-1|n-1) + \mathbf{K}y_n)^t]. \quad \text{Eq. (17.74)}$$

Expanding the right-hand side of Eq. (17.74), and using Eq. (17.71), gives

$$\mathbf{C}(n|n) = \mathbf{A}\mathbf{C}(n-1|n-1)\mathbf{A}^t + \mathbf{K}\sigma_v^2\mathbf{K}^t . \qquad \text{Eq. (17.75)}$$

Under the steady-state condition, Eq. (17.75) collapses to

$$\mathbf{C}(n|n) = \mathbf{A}\mathbf{C}\mathbf{A}^t + \mathbf{K}\sigma_v^2\mathbf{K}^t \qquad \text{Eq. (17.76)}$$

where $\mathbf{C}$ is the steady-state noise covariance matrix. In the steady-state,

$$\mathbf{C}(n|n) = \mathbf{C}(n-1|n-1) = \mathbf{C} \qquad for\ any\ n \qquad \text{Eq. (17.77)}$$

Several criteria can be used to establish the performance of the fixed-gain tracking filter. The most commonly used technique is to compute the Variance Reduction Ratio (VRR). The VRR is defined only when the input to the tracker is noise measurements. It follows that in the steady-state case, the VRR is the steady-state ratio of the output variance (auto-covariance) to the input measurement variance.

In order to determine the stability of the tracker under consideration, consider the Z-transform for Eq. (17.72),

$$\mathbf{x}(z) = \mathbf{A}z^{-1}\mathbf{x}(z) + \mathbf{K}y_n(z) . \qquad \text{Eq. (17.78)}$$

Rearranging Eq. (17.78) yields the following system transfer functions:

$$\mathbf{h}(z) = \frac{\mathbf{x}(z)}{y_n(z)} = (\mathbf{I} - \mathbf{A}z^{-1})^{-1}\mathbf{K} \qquad \text{Eq. (17.79)}$$

where $(\mathbf{I} - \mathbf{A}z^{-1})$ is called the characteristic matrix. Note that the system transfer functions can exist only when the characteristic matrix is a non-singular matrix. Additionally, the system is stable if and only if the roots of the characteristic equation are within the unit circle in the z-plane,

$$\left|(\mathbf{I} - \mathbf{A}z^{-1})\right| = 0 . \qquad \text{Eq. (17.80)}$$

The filter's steady-state errors can be determined with the help of Fig. 17.19. The error transfer function is

$$\mathbf{e}(z) = \frac{\mathbf{y}(z)}{1 + \mathbf{h}(z)}, \qquad \text{Eq. (17.81)}$$

and by using Abel's theorem, the steady-state error is

$$\mathbf{e}_\infty = \lim_{t \to \infty} \mathbf{e}(t) = \lim_{z \to 1}\left(\frac{z-1}{z}\right)\ \mathbf{e}(z) . \qquad \text{Eq. (17.82)}$$

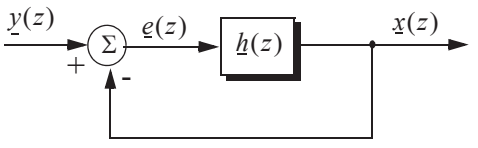


Figure 17.19. Steady-state error computation.

Substituting Eq. (17.82) into (17.81) yields

$$\mathbf{e}_{\infty} = \lim_{z \to 1} \frac{z-1}{z} \frac{\mathbf{y}(z)}{1+\mathbf{h}(z)}. \qquad \text{Eq. (17.83)}$$

17.8.1. The αβ Filter

The αβ tracker produces, on the nth observation, smoothed estimates for position and velocity, and a predicted position for the $(n+1)th$ observation. Figure 17.20 shows an implementation of this filter. Note that the subscripts "p" and "s" are used to indicate, respectively, the predicated and smoothed values. The αβ tracker can follow an input ramp (constant velocity) with no steady-state errors. However, a steady-state error will accumulate when constant acceleration is present in the input. Smoothing is done to reduce errors in the predicted position through adding a weighted difference between the measured and predicted values to the predicted position, as follows:

$$x_s(n) = x(n|n) = x_p(n) + \alpha(x_0(n) - x_p(n)) \qquad \text{Eq. (17.84)}$$

$$\dot{x}_s(n) = x'(n|n) = \dot{x}_s(n-1) + \frac{\beta}{T}(x_0(n) - x_p(n)). \qquad \text{Eq. (17.85)}$$

x_0 is the position input samples. The predicted position is given by

$$x_p(n) = x_s(n|n-1) = x_s(n-1) + T\dot{x}_s(n-1). \qquad \text{Eq. (17.86)}$$

The initialization process is defined by

$$x_s(1) = x_p(2) = x_0(1)$$

$$\dot{x}_s(1) = 0$$

$$\dot{x}_s(2) = \frac{x_0(2) - x_0(1)}{T}.$$

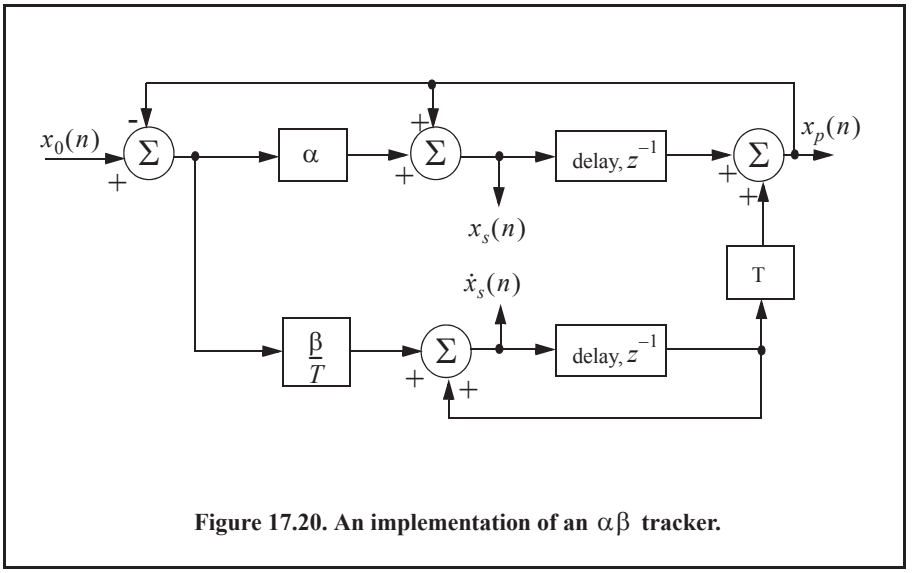


Figure 17.20. An implementation of an αβ tracker.

A general form for the covariance matrix was developed in the previous section, and is given in Eq. (17.75). In general, a second-order one-dimensional covariance matrix (in the context of the $\alpha\beta$ filter) can be written as

$$\mathbf{C}(n|n) = \begin{bmatrix} C_{xx} & C_{x\dot{x}} \\ C_{\dot{x}x} & C_{\dot{x}\dot{x}} \end{bmatrix} \qquad \text{Eq. (17.87)}$$

where, in general, C_{xy} is

$$C_{xy} = E\{xy^t\}. \qquad \text{Eq. (17.88)}$$

By inspection, the $\alpha\beta$ filter has

$$\mathbf{A} = \begin{bmatrix} 1-\alpha & (1-\alpha)T \\ -\beta/T & (1-\beta) \end{bmatrix} \qquad \text{Eq. (17.89)}$$

$$\mathbf{K} = \begin{bmatrix} \alpha \\ \beta/T \end{bmatrix} \qquad \text{Eq. (17.90)}$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \text{Eq. (17.91)}$$

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}. \qquad \text{Eq. (17.92)}$$

Finally, using Eqs. (17.89) through (17.92) in Eq. (17.72) yields the steady-state noise covariance matrix,

$$\mathbf{C} = \frac{\sigma_v^2}{\alpha(4-2\alpha-\beta)} \begin{bmatrix} 2\alpha^2 - 3\alpha\beta + 2\beta & \dfrac{\beta(2\alpha-\beta)}{T} \\ \dfrac{\beta(2\alpha-\beta)}{T} & \dfrac{2\beta^2}{T^2} \end{bmatrix}. \qquad \text{Eq. (17.93)}$$

It follows that the position and velocity VRR ratios are, respectively, given by

$$(VRR)_x = C_{xx}/\sigma_v^2 = \frac{2\alpha^2 - 3\alpha\beta + 2\beta}{\alpha(4-2\alpha-\beta)} \qquad \text{Eq. (17.94)}$$

$$(VRR)_{\dot{x}} = C_{\dot{x}\dot{x}}/\sigma_v^2 = \frac{1}{T^2}\ \frac{2\beta^2}{\alpha(4-2\alpha-\beta)}. \qquad \text{Eq. (17.95)}$$

The stability of the $\alpha\beta$ filter is determined from its system transfer functions. For this purpose, compute the roots for Eq. (17.80) with $\mathbf{A}$ from Eq. (17.89),

$$\left|\mathbf{I} - \mathbf{A}z^{-1}\right| = 1 - (2-\alpha-\beta)z^{-1} + (1-\alpha)z^{-2} = 0. \qquad \text{Eq. (17.96)}$$

Solving Eq. (17.96) for z yields

$$z_{1,2} = 1 - \frac{\alpha+\beta}{2} \pm \frac{1}{2}\sqrt{(\alpha-\beta)^2 - 4\beta}, \qquad \text{Eq. (17.97)}$$

and in order to guarantee stability,

$$|z_{1,2}| < 1 . \qquad \textbf{Eq. (17.98)}$$

Two cases are analyzed. First, $z_{1,2}$ are real. In this case (the details are left as an exercise),

$$\beta > 0 \quad ; \quad \alpha > -\beta . \qquad \textbf{Eq. (17.99)}$$

The second case is when the roots are complex; in this case we find

$$\alpha > 0 . \qquad \textbf{Eq. (17.100)}$$

The system transfer functions can be derived by using Eqs. (17.79), (17.89), and (17.90),

$$\begin{bmatrix} h_x(z) \\ h_{\dot{x}}(z) \end{bmatrix} = \frac{1}{z^2 - z(2-\alpha-\beta) + (1-\alpha)} \begin{bmatrix} \alpha z\left(z - \dfrac{(\alpha-\beta)}{\alpha}\right) \\ \dfrac{\beta z(z-1)}{T} \end{bmatrix} . \qquad \textbf{Eq. (17.101)}$$

Up to this point all relevant relations concerning the $\alpha\beta$ filter were made with no regard to how to choose the gain coefficients (α and β). Before considering the methodology of selecting these coefficients, consider the main objective behind using this filter. The twofold purpose of the $\alpha\beta$ tracker can be described as follows:

1. *The tracker must reduce the measurement noise as much as possible.*
2. *The filter must be able to track maneuvering targets, with as little residual (tracking error) as possible.*

The reduction of measurement noise is normally determined by the VRR ratios. However, the maneuverability performance of the filter depends heavily on the choice of the parameters α and β.

A special variation of the $\alpha\beta$ filter was developed by Benedict and Bordner[1] and is often referred to as the Benedict-Bordner filter. The main advantage of the Benedict-Bordner is reducing the transient errors associated with the $\alpha\beta$ tracker. This filter uses both the position and velocity VRR ratios as measures of performance. It computes the sum of the squared differences between the input (position) and the output when the input has a unit step velocity at time zero. Additionally, it computes the squared differences between the real velocity and the velocity output when the input is as described earlier. Both error differences are minimized when

$$\beta = \frac{\alpha^2}{2-\alpha} . \qquad \textbf{Eq. (17.102)}$$

In this case, the position and velocity VRR ratios are, respectively, given by

$$(VRR)_x = \frac{\alpha(6-5\alpha)}{\alpha^2 - 8\alpha + 8} \qquad \textbf{Eq. (17.103)}$$

$$(VRR)_{\dot{x}} = \frac{2}{T^2} \frac{\alpha^3/(2-\alpha)}{\alpha^2 - 8\alpha + 8} . \qquad \textbf{Eq. (17.104)}$$

1. Benedict, T. R. and Bordner, G. W., Synthesis of an Optimal Set of Radar Track-While-Scan Smoothing Equations. *IRE Transaction on Automatic Control, AC-7*. July 1962, pp. 27-32.

Another important sub-class of the $\alpha\beta$ tracker is the critically damped filter, often called the fading memory filter. In this case, the filter coefficients are chosen on the basis of a smoothing factor ξ, where $0 \le \xi \le 1$. The gain coefficients are given by

$$\alpha = 1 - \xi^2 \qquad \textbf{Eq. (17.105)}$$

$$\beta = (1 - \xi)^2 . \qquad \textbf{Eq. (17.106)}$$

Heavy smoothing means $\xi \to 1$ and little smoothing means $\xi \to 0$. The elements of the covariance matrix for a fading memory filter are

$$C_{xx} = \frac{1 - \xi}{(1 + \xi)^3} \; (1 + 4\xi + 5\xi^2) \; \sigma_v^2 \qquad \textbf{Eq. (17.107)}$$

$$C_{x\dot{x}} = C_{\dot{x}x} = \frac{1}{T} \; \frac{1 - \xi}{(1 + \xi)^3} \; (1 + 2\xi + 3\xi^2) \; \sigma_v^2 \qquad \textbf{Eq. (17.108)}$$

$$C_{\dot{x}\dot{x}} = \frac{2}{T^2} \; \frac{1 - \xi}{(1 + \xi)^3} \; (1 - \xi)^2 \; \sigma_v^2 . \qquad \textbf{Eq. (17.109)}$$

17.8.2. The $\alpha\beta\gamma$ Filter

The $\alpha\beta\gamma$ tracker produces, for the nth observation, smoothed estimates of position, velocity, and acceleration. It also produces the predicted position and velocity for the $(n+1)th$ observation. An implementation of the $\alpha\beta\gamma$ tracker is shown in Fig. 17.21.

The $\alpha\beta\gamma$ tracker will follow an input whose acceleration is constant with no steady-state errors. Again, in order to reduce the error at the output of the tracker, a weighted difference between the measured and predicted values is used in estimating the smoothed position, velocity, and acceleration as follows:

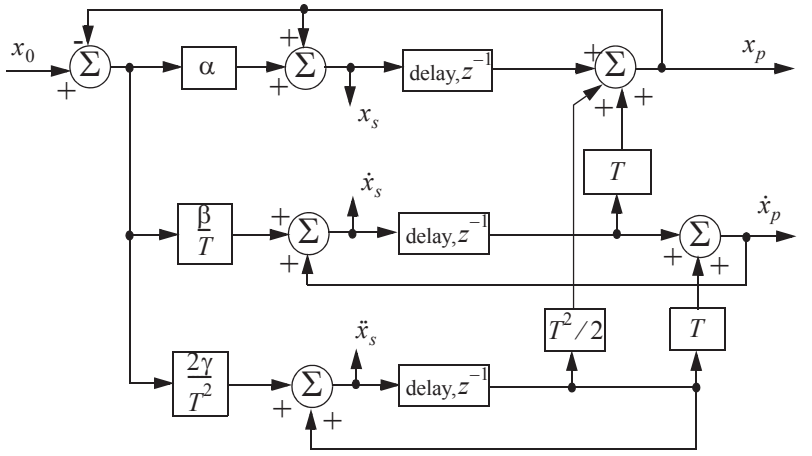


Figure 17.21. An implementation for an $\alpha\beta\gamma$ tracker.

$$x_s(n) = x_p(n) + \alpha(x_0(n) - x_p(n)) \quad \text{Eq. (17.110)}$$

$$\dot{x}_s(n) = \dot{x}_s(n-1) + T\ddot{x}_s(n-1) + \frac{\beta}{T}\ (x_0(n) - x_p(n)) \quad \text{Eq. (17.111)}$$

$$\ddot{x}_s(n) = \ddot{x}_s(n-1) + \frac{2\gamma}{T^2}\ (x_0(n) - x_p(n)) \quad \text{Eq. (17.112)}$$

$$x_p(n+1) = x_s(n) + T\ \dot{x}_s(n) + \frac{T^2}{2}\ \ddot{x}_s(n)\ . \quad \text{Eq. (17.113)}$$

and the initialization process is

$$x_s(1) = x_p(2) = x_0(1)$$

$$\dot{x}_s(1) = \ddot{x}_s(1) = \ddot{x}_s(2) = 0$$

$$\dot{x}_s(2) = \frac{x_0(2) - x_0(1)}{T}$$

$$\ddot{x}_s(3) = \frac{x_0(3) + x_0(1) - 2x_0(2)}{T^2}\ .$$

Using Eq. (17.63), the state transition matrix for the $\alpha\beta\gamma$ filter is

$$\Phi = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}. \quad \text{Eq. (17.114)}$$

The covariance matrix (which is symmetric) can be computed from Eq. (17.76). For this purpose, note that

$$\mathbf{K} = \begin{bmatrix} \alpha \\ \beta/T \\ \gamma/T^2 \end{bmatrix} \quad \text{Eq. (17.115)}$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad \text{Eq. (17.116)}$$

and

$$\mathbf{A} = (\mathbf{I} - \mathbf{KG})\Phi = \begin{bmatrix} 1-\alpha & (1-\alpha)T & (1-\alpha)T^2/2 \\ -\beta/T & -\beta+1 & (1-\beta/2)T \\ -2\gamma/T^2 & -2\gamma/T & (1-\gamma) \end{bmatrix}. \quad \text{Eq. (17.117)}$$

Substituting Eq. (17.117) into (17.76) and collecting terms, the VRR ratios are computed as

$$(VRR)_x = \frac{2\beta(2\alpha^2 + 2\beta - 3\alpha\beta) - \alpha\gamma(4 - 2\alpha - \beta)}{(4 - 2\alpha - \beta)(2\alpha\beta + \alpha\gamma - 2\gamma)} \quad \text{Eq. (17.118)}$$

$$(VRR)_{\dot{x}} = \frac{4\beta^3 - 4\beta^2\gamma + 2\gamma^2(2-\alpha)}{T^2(4-2\alpha-\beta)(2\alpha\beta+\alpha\gamma-2\gamma)} \quad \textbf{Eq. (17.119)}$$

$$(VRR)_{\ddot{x}} = \frac{4\beta\gamma^2}{T^4(4-2\alpha-\beta)(2\alpha\beta+\alpha\gamma-2\gamma)} . \quad \textbf{Eq. (17.120)}$$

As in the case of any discrete time system, this filter will be stable if and only if all of its poles fall within the unit circle in the z-plane.

The $\alpha\beta\gamma$ characteristic equation is computed by setting

$$\left|\mathbf{I} - \mathbf{A}z^{-1}\right| = 0 . \quad \textbf{Eq. (17.121)}$$

Substituting Eq. (17.117) into (17.121) and collecting terms yields the following characteristic function:

$$f(z) = z^3 + (-3\alpha + \beta + \gamma)z^2 + (3 - \beta - 2\alpha + \gamma)z - (1-\alpha) . \quad \textbf{Eq. (17.122)}$$

The $\alpha\beta\gamma$ becomes a Benedict-Bordner filter when

$$2\beta - \alpha\left(\alpha + \beta + \frac{\gamma}{2}\right) = 0 . \quad \textbf{Eq. (17.123)}$$

Note that for $\gamma = 0$, Eq. (17.123) reduces to Eq. (17.102). For a critically damped filter the gain coefficients are

$$\alpha = 1 - \xi^3 \quad \textbf{Eq. (17.124)}$$

$$\beta = 1.5(1-\xi^2)(1-\xi) = 1.5(1-\xi)^2(1+\xi) \quad \textbf{Eq. (17.125)}$$

$$\gamma = (1-\xi)^3 . \quad \textbf{Eq. (17.126)}$$

Note that heavy smoothing takes place when $\xi \to 1$, while $\xi = 0$ means that no smoothing is present.

MATLAB Function "ghk_tracker.m"

The function *"ghk_tracker.m"* implements the steady-state $\alpha\beta\gamma$ filter. The syntax is as follows:

[residual, estimate] = ghk_tracker (X0, smoocof, inp, npts, T, nvar)

where

Symbol	Description	Status
X0	*initial state vector*	*input*
smoocof	*desired smoothing coefficient*	*input*
inp	*array of position measurements*	*input*
npts	*number of points in input position*	*input*
T	*sampling interval*	*input*
nvar	*desired noise variance*	*input*

Symbol	Description	Status
residual	*array of position error (residual)*	*output*
estimate	*array of predicted position*	*output*

Note that *"ghk_tracker.m"* uses MATLAB's function *"normrnd.m"* to generate zero mean Gaussian noise, which is part of MATLAB's Statistics Toolbox. If this toolbox is not available to the user, then *"ghk_tracker.m"* function-call must be modified to

[residual, estimate] = ghk_tracker1 (X0, smoocof, inp, npts, T)

In this case, noise measurements are either considered to be unavailable or are part of the position input array.

To illustrate how to use the functions "*ghk_tracker.m"* and *"ghk_tracker1.m,"* consider the inputs shown in Figs. 17.22 and 17.23. Figure 17.22 assumes an input with lazy maneuvering, while Figure 17.23 assumes an aggressive maneuvering case. These figures can be reproduced using MATLAB program *"Fig17_20s.m,"* listed in Appendix 17-A.

Figures 17.24 and 17.25 show the residual error and predicted position corresponding to Fig. 17.22 assuming the cases: heavy smoothing and little smoothing with and without noise. The noise is white Gaussian with zero mean and variance of $\sigma_v^2 = 0.05$. Figures 17. 26 and 17.27 show the residual error and predicted position corresponding to Fig. 17.23 with and without noise.

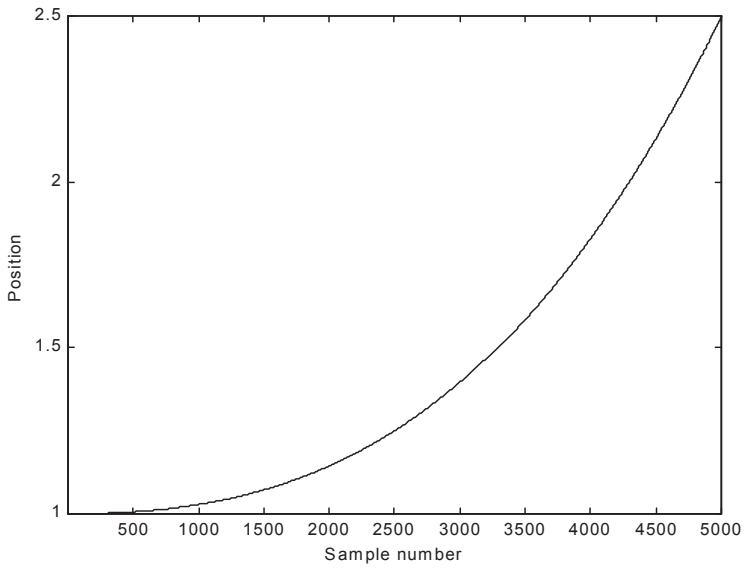


Figure 17.22. Position (truth-data); lazy maneuvering.

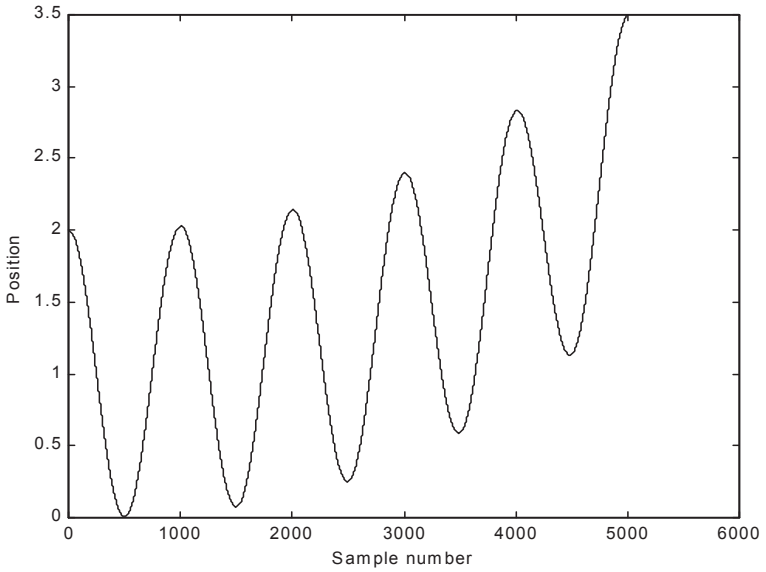


Figure 17.23. Position (truth-data); aggressive maneuvering.

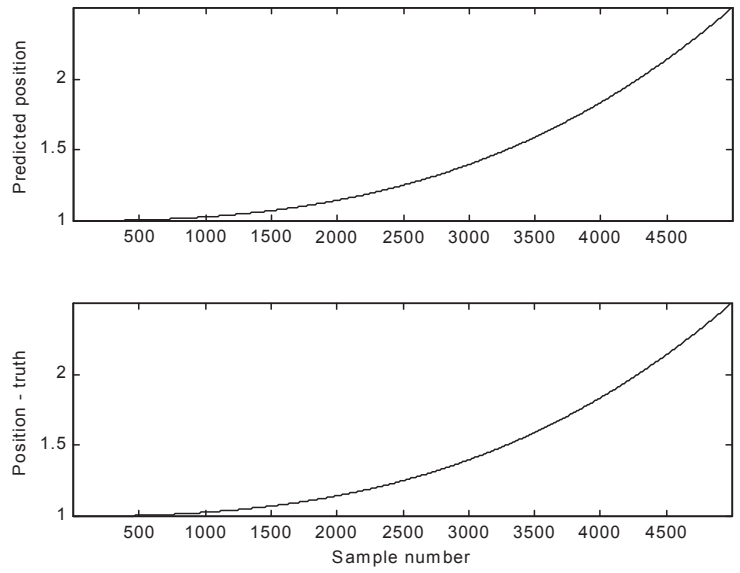


Figure 17.24a-1. Predicted and true position. $\xi = 0.1$ (i.e., large gain coefficients). No noise present.

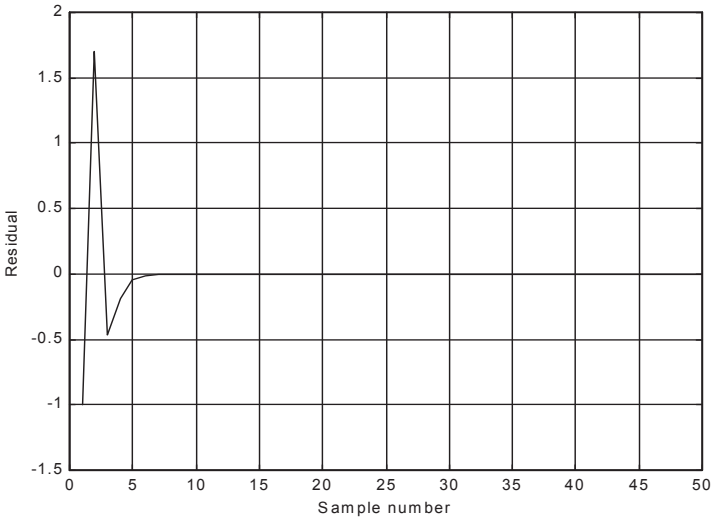


Figure 17.24a-2. Position residual (error). Large gain coefficients. No noise. The error settles to zero fairly quickly.

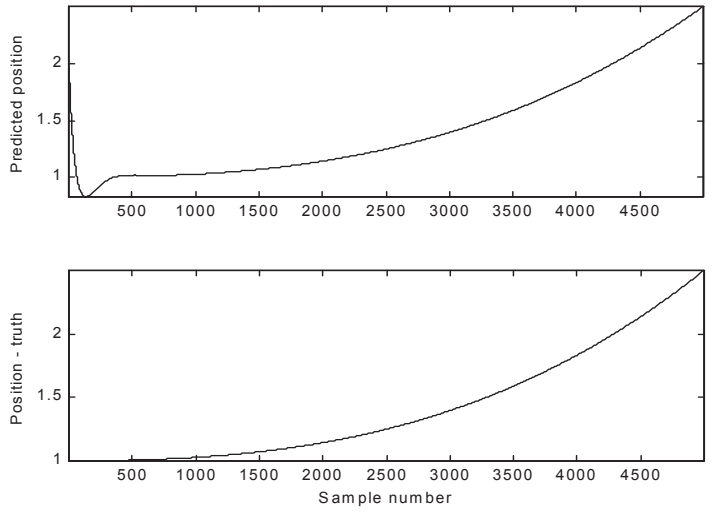


Figure 17.24b-1. Predicted and true position. $\xi = 0.9$ (i.e., small gain coefficients). No noise present.

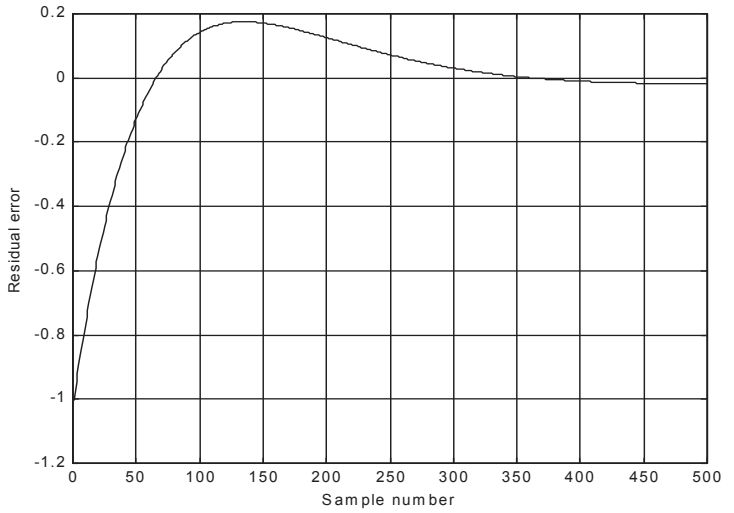


Figure 17.24b-2. Position residual (error). Small gain coefficients. No noise. It takes the filter a longer time for the error to settle down.

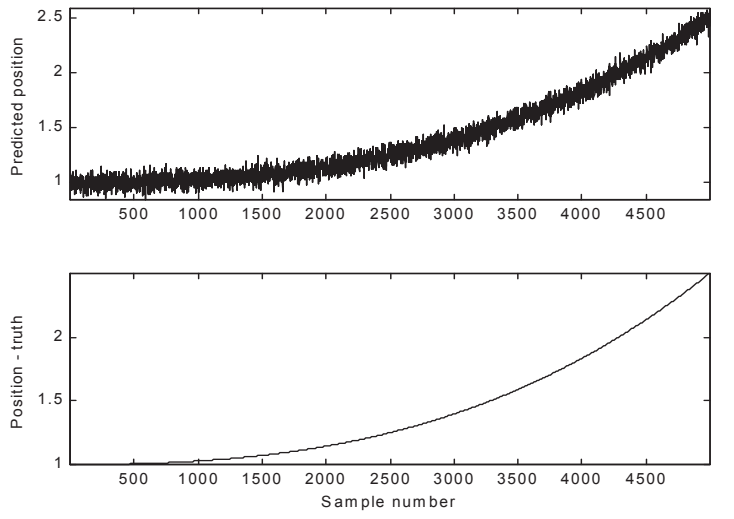


Figure 17.25a-1. Predicted and true position. $\xi = 0.1$ (i.e., large gain coefficients). Noise is present.

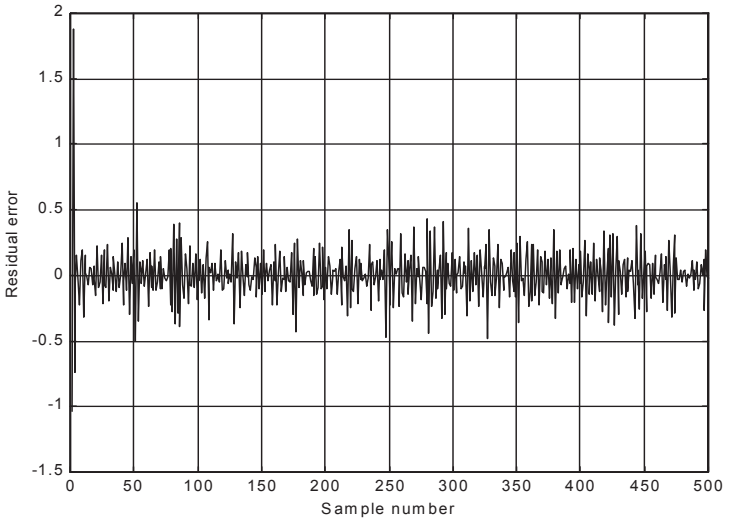


Figure 17.25a-2. Position residual (error). Large gain coefficients. Noise present. The error settles down quickly. The variation is due to noise.

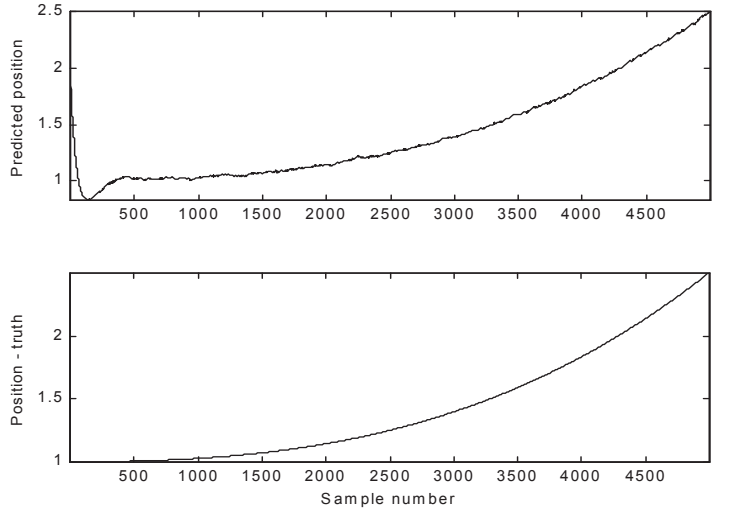


Figure 17.25b-1. Predicted and true position. $\xi = 0.9$ (i.e., small gain coefficients). Noise is present.

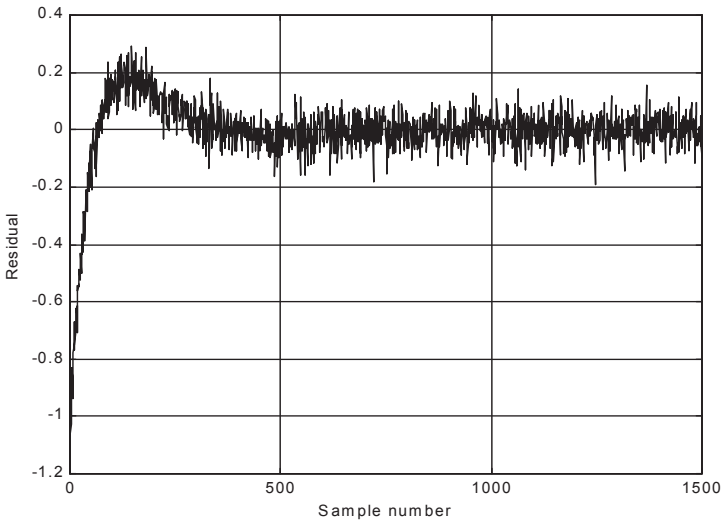


Figure 17.25b-2. Position residual (error). Small gain coefficients. Noise present. The error requires more time before settling down. The variation is due to noise.

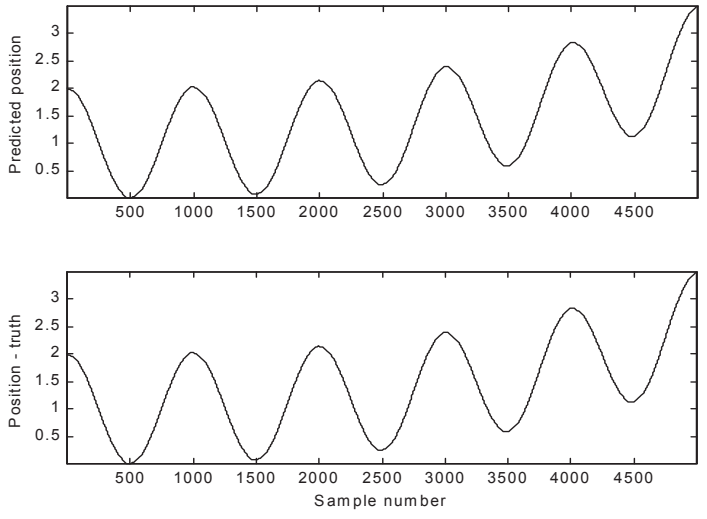


Figure 17.26a. Predicted and true position. $\xi = 0.1$ (i.e., large gain coefficients). Noise is present.

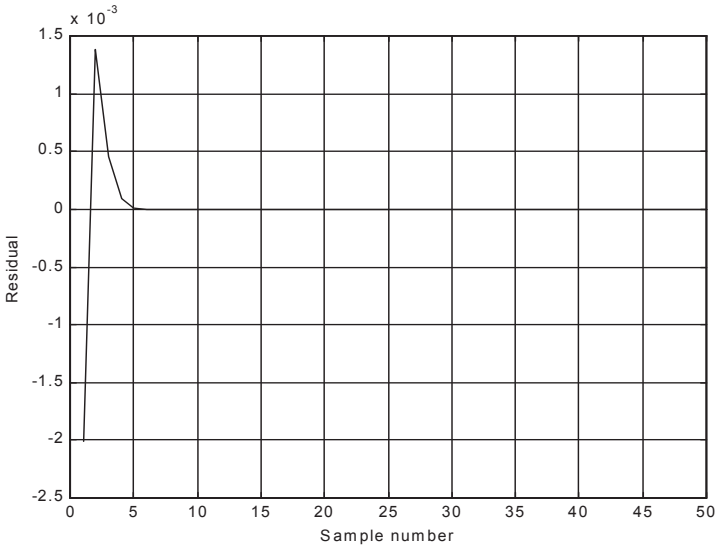


Figure 17.26b. Position residual (error). Large gain coefficients. No noise. The error settles down quickly.

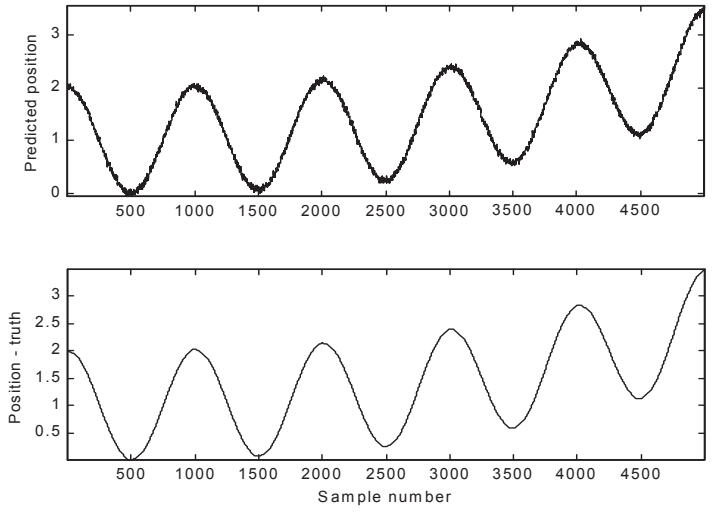


Figure 17.27a. Predicted and true position. $\xi = 0.8$ (i.e., small gain coefficients). Noise is present.

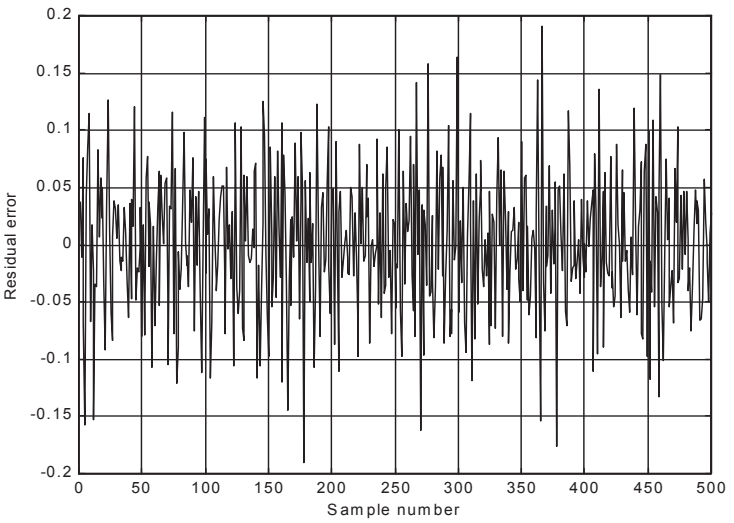


Figure 17.27b. Position residual (error). Small gain coefficients. Noise present. The error stays fairly large; however, its average is around zero. The variation is due to noise.

17.9. The Kalman Filter

The Kalman filter is a linear estimator that minimizes the mean squared error as long as the target dynamics are modeled accurately. All other recursive filters, such as the $\alpha\beta\gamma$ and the Benedict-Bordner filters, are special cases of the general solution provided by the Kalman filter for the mean squared estimation problem. Additionally, the Kalman filter has the following advantages:

1. *The gain coefficients are computed dynamically. This means that the same filter can be used for a variety of maneuvering target environments.*
2. *The Kalman filter gain computation adapts to varying detection histories, including missed detections.*
3. *The Kalman filter provides an accurate measure of the covariance matrix. This allows for better implementation of the gating and association processes.*
4. *The Kalman filter makes it possible to partially compensate for the effects of mis-correlation and mis-association.*

Many derivations of the Kalman filter exist in the literature; only results are provided in this chapter. Figure17.28 shows a block diagram for the Kalman filter. The Kalman filter equations can be deduced from Fig. 17.28. The filtering equation is

$$\mathbf{x}(n|n) = \mathbf{x}_s(n) = \mathbf{x}(n|n-1) + \mathbf{K}(n)[\mathbf{y}(n) - \mathbf{G}\mathbf{x}(n|n-1)]. \qquad \text{Eq. (17.127)}$$

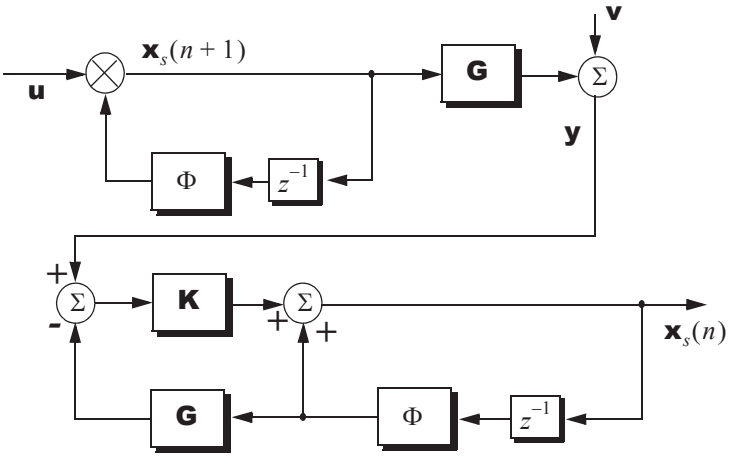


Figure 17.28. Structure of the Kalman filter.

The measurement vector is

$$\mathbf{y}(n) = \mathbf{G}\mathbf{x}(n) + \mathbf{v}(n) \qquad \text{Eq. (17.128)}$$

where $\mathbf{y}(n)$ is zero mean, white Gaussian noise with covariance $\Re_c$,

$$\Re_c = E\{\mathbf{y}(n)\ \mathbf{y}^t(n)\}. \qquad \text{Eq. (17.129)}$$

The gain (weight) vector is dynamically computed as

$$\mathbf{K}(n) = \mathbf{P}(n|n-1)\mathbf{G}^t[\mathbf{G}\mathbf{P}(n|n-1)\mathbf{G}^t + \Re_c]^{-1} \qquad \text{Eq. (17.130)}$$

where the measurement noise matrix $\mathbf{P}$ represents the predictor covariance matrix, and is equal to

$$\mathbf{P}(n+1|n) = E\{\mathbf{x}_s(n+1)\mathbf{x}^*_s(n)\} = \Phi\mathbf{P}(n|n)\Phi^t + \mathbf{Q} \qquad \text{Eq. (17.131)}$$

where $\mathbf{Q}$ is the covariance matrix for the input $\mathbf{u}$,

$$\mathbf{Q} = E\{\mathbf{u}(n)\ \mathbf{u}^t(n)\}. \qquad \text{Eq. (17.132)}$$

The corrector equation (covariance of the smoothed estimate) is

$$\mathbf{P}(n|n) = [\mathbf{I} - \mathbf{K}(n)\mathbf{G}]\mathbf{P}(n|n-1). \qquad \text{Eq. (17.133)}$$

Finally, the predictor equation is

$$\mathbf{x}(n+1|n) = \Phi\mathbf{x}(n|n). \qquad \text{Eq. (17.134)}$$

17.9.1. The Singer $\alpha\beta\gamma$-Kalman Filter

The Singer[1] filter is a special case of the Kalman, where the filter is governed by a specified target dynamic model whose acceleration is a random process with autocorrelation function given by

$$E\{\ddot{x}(t)\ \ddot{x}(t+t_1)\} = \sigma_a^2\ e^{-\frac{|t_1|}{\tau_m}} \qquad \textbf{Eq. (17.135)}$$

where τ_m is the correlation time of the acceleration due to target maneuvering or atmospheric turbulence. The correlation time τ_m may vary from as low as 10 seconds for aggressive maneuvering to as large as 60 seconds for lazy maneuvering cases.

Singer defined the random target acceleration model by a first-order Markov process given by

$$\ddot{x}(n+1) = \rho_m\ \ddot{x}(n) + \sqrt{1-\rho_m^2}\ \ \sigma_m\ \ w(n) \qquad \textbf{Eq. (17.136)}$$

where $w(n)$ is a zero mean, Gaussian random variable with unity variance, σ_m is the maneuver standard deviation, and the maneuvering correlation coefficient ρ_m is given by

$$\rho_m = e^{-\frac{T}{\tau_m}}. \qquad \textbf{Eq. (17.137)}$$

The continuous time domain system that corresponds to these conditions is the same as the Wiener-Kolmogorov whitening filter, which is defined by the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}v(t) = -\beta_m v(t) + w(t) \qquad \textbf{Eq. (17.138)}$$

where β_m is equal to $1/\tau_m$. The maneuvering variance using Singer's model is given by

$$\sigma_m^2 = \frac{A_{max}^2}{3}[1 + 4P_{max} - P_0]. \qquad \textbf{Eq. (17.139)}$$

A_{max} is the maximum target acceleration with probability P_{max}, and the term P_0 defines the probability that the target has no acceleration.

The transition matrix that corresponds to the Singer filter is given by

$$\Phi = \begin{bmatrix} 1 & T & \frac{1}{\beta_m^2}(-1 + \beta_m T + \rho_m) \\ 0 & 1 & \frac{1}{\beta_m}(1 - \rho_m) \\ 0 & 0 & \rho_m \end{bmatrix}. \qquad \textbf{Eq. (17.140)}$$

Note that when $T\beta_m = T/\tau_m$ is small (the target has constant acceleration), then Eq. (17.140) reduces to Eq. (17.114). Typically, the sampling interval T is much less than the maneuvering

1. Singer, R. A., Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets, *IEEE Transaction on Aerospace and Electronics, AES-5*, July, 1970. pp. 473-483.

time constant τ_m ; hence, Eq. (17.140) can be accurately replaced by its second-order approximation. More precisely,

$$\Phi = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T(1-T/2\tau_m) \\ 0 & 0 & \rho_m \end{bmatrix}. \qquad \textbf{Eq. (17.141)}$$

The covariance matrix was derived by Singer, and it is equal to

$$\mathbf{C} = \frac{2\sigma_m^2}{\tau_m} \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \qquad \textbf{Eq. (17.142)}$$

where

$$C_{11} = \sigma_x^2 = \frac{1}{2\beta_m^5}\left[1 - e^{-2\beta_m T} + 2\beta_m T + \frac{2\beta_m^3 T^3}{3} - 2\beta_m^2 T^2 - 4\beta_m T e^{-\beta_m T}\right] \qquad \textbf{Eq. (17.143)}$$

$$C_{12} = C_{21} = \frac{1}{2\beta_m^4}[e^{-2\beta_m T} + 1 - 2e^{-\beta_m T} + 2\beta_m T e^{-\beta_m T} - 2\beta_m T + \beta_m^2 T^2] \qquad \textbf{Eq. (17.144)}$$

$$C_{13} = C_{31} = \frac{1}{2\beta_m^3}[1 - e^{-2\beta_m T} - 2\beta_m T e^{-\beta_m T}] \qquad \textbf{Eq. (17.145)}$$

$$C_{22} = \frac{1}{2\beta_m^3}[4e^{-\beta_m T} - 3 - e^{-2\beta_m T} + 2\beta_m T] \qquad \textbf{Eq. (17.146)}$$

$$C_{23} = C_{32} = \frac{1}{2\beta_m^2}[e^{-2\beta_m T} + 1 - 2e^{-\beta_m T}] \qquad \textbf{Eq. (17.147)}$$

$$C_{33} = \frac{1}{2\beta_m}[1 - e^{-2\beta_m T}]. \qquad \textbf{Eq. (17.148)}$$

Two limiting cases are of interest:

1. *The short sampling interval case ($T \ll \tau_m$),*

$$\lim_{\beta_m T \to 0} \mathbf{C} = \frac{2\sigma_m^2}{\tau_m} \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix} \qquad \textbf{Eq. (17.149)}$$

and the state transition matrix is computed from Eq. (17.141) as

$$\lim_{\beta_m T \to 0} \Phi = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \qquad \textbf{Eq. (17.150)}$$

which is the same as the case for the $\alpha\beta\gamma$ filter (constant acceleration).

2. *The long sampling interval ($T \gg \tau_m$). This condition represents the case when acceleration is a white noise process. The corresponding covariance and transition matrices are, respectively, given by*

$$\lim_{\beta_m T \to \infty} \mathbf{C} = \sigma_m^2 \begin{bmatrix} \frac{2T^3\tau_m}{3} & T^2\tau_m & \tau_m^2 \\ T^2\tau_m & 2T\tau_m & \tau_m \\ \tau_m^2 & \tau_m & 1 \end{bmatrix} \qquad \text{Eq. (17.151)}$$

$$\lim_{\beta_m T \to \infty} \Phi = \begin{bmatrix} 1 & T & T\tau_m \\ 0 & 1 & \tau_m \\ 0 & 0 & 0 \end{bmatrix}. \qquad \text{Eq. (17.152)}$$

Note that under the condition that $T \gg \tau_m$, the cross correlation terms C_{13} and C_{23} become very small. It follows that estimates of acceleration are no longer available, and thus a two-state filter model can be used to replace the three-state model. In this case,

$$\mathbf{C} = 2\sigma_m^2\tau_m \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \qquad \text{Eq. (17.153)}$$

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}. \qquad \text{Eq. (17.154)}$$

17.9.2. Relationship between Kalman and $\alpha\beta\gamma$ Filters

The relationship between the Kalman filter and the $\alpha\beta\gamma$ filters can be easily obtained by using the appropriate state transition matrix $\underline{\Phi}$, and gain vector $\underline{K}$ corresponding to the $\alpha\beta\gamma$ in Eq. (17.127). Thus,

$$\begin{bmatrix} x(n|n) \\ \dot{x}(n|n) \\ \ddot{x}(n|n) \end{bmatrix} = \begin{bmatrix} x(n|n-1) \\ \dot{x}(n|n-1) \\ \ddot{x}(n|n-1) \end{bmatrix} + \begin{bmatrix} k_1(n) \\ k_2(n) \\ k_3(n) \end{bmatrix} [x_0(n) - x(n|n-1)] \qquad \text{Eq. (17.155)}$$

with (see Fig. 17.21)

$$x(n|n-1) = x_s(n-1) + T\ \dot{x}_s(n-1) + \frac{T^2}{2}\ \ddot{x}_s(n-1) \qquad \text{Eq. (17.156)}$$

$$\dot{x}(n|n-1) = \dot{x}_s(n-1) + T\ \ddot{x}_s(n-1) \qquad \text{Eq. (17.157)}$$

$$\ddot{x}(n|n-1) = \ddot{x}_s(n-1). \qquad \text{Eq. (17.158)}$$

Comparing the previous three equations with the $\alpha\beta\gamma$ filter equations yields,

$$\begin{bmatrix} \alpha \\ \beta \\ T \\ \gamma \\ \overline{T^2} \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix}. \qquad \text{Eq. (17.159)}$$

Additionally, the covariance matrix elements are related to the gain coefficients by

$$\begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = \frac{1}{C_{11} + \sigma_v^2} \begin{bmatrix} C_{11} \\ C_{12} \\ C_{13} \end{bmatrix}. \qquad \text{Eq. (17.160)}$$

Eq. (17.160) indicates that the first gain coefficient depends on the estimation error variance of the total residual variance, while the other two gain coefficients are calculated through the covariances between the second and third states and the first observed state.

MATLAB Function "kalman_filter.m"

The function *"kalman_filter.m"* implements a state Singer-$\alpha\beta\gamma$ Kalman filter. The syntax is as follows:

[residual, estimate] = kalman_filter(npts, T, X0, inp, R, nvar)

where

Symbol	Description	Status
npts	*number of points in input position*	*input*
T	*sampling interval*	*input*
X0	*initial state vector*	*input*
inp	*input array*	*input*
R	*noise variance see Eq. (10-129)*	*input*
nvar	*desired state noise variance*	*input*
residual	*array of position error (residual)*	*output*
estimate	*array of predicted position*	*output*

Note that *"kalman_filter.m"* uses MATLAB's function *"normrnd.m"* to generate zero mean Gaussian noise, which is part of MATLAB's Statistics Toolbox.

To illustrate how to use the functions "*kalman_filter.m,"* consider the inputs shown in Figs. 17.22 and 17.23. Figures 17.29 and 17.30 show the residual error and predicted position corresponding to Figures 17.22 and 17.23. These plots can be reproduced using the MATLAM program *"Fig17_28.m,"* listed in Appendix 17-A.

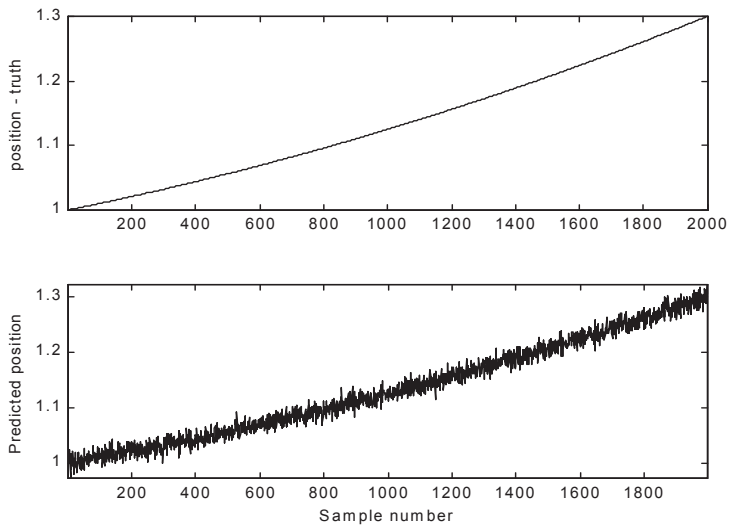


Figure 17.29a. True and predicted positions. Lazy maneuvering. Plot produced using the function *"kalman_filter.m."*

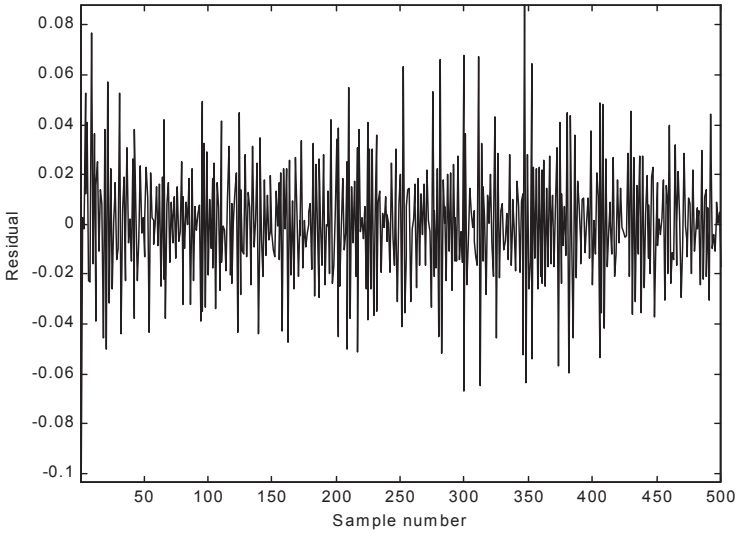


Figure 17.29b. Residual corresponding to Fig. 17.29a.

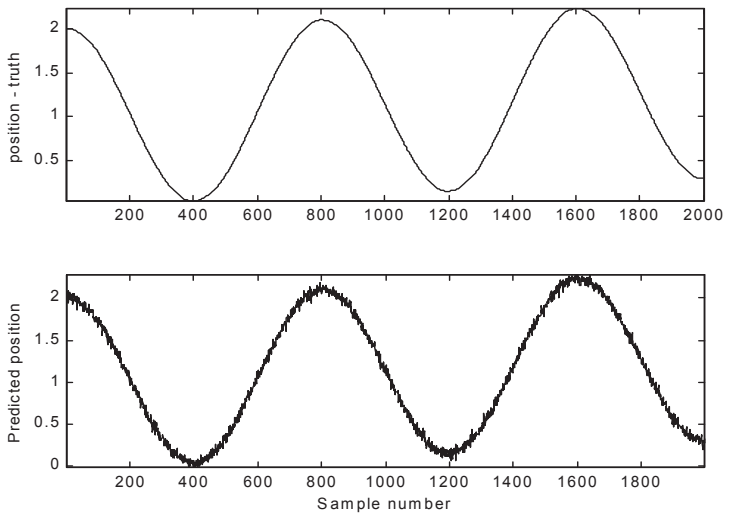


Figure 17.30a. True and predicted positions. Aggressive maneuvering. Plot produced using the function *"kalman_filter.m."*

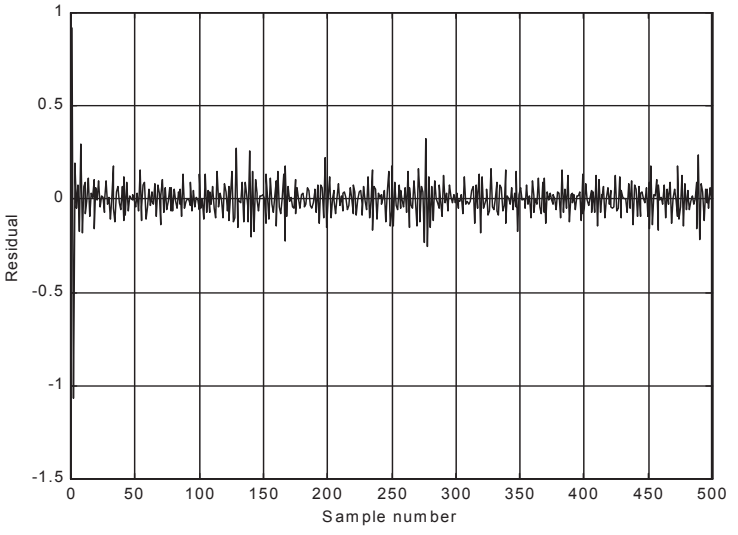


Figure 17.30b. Residual corresponding to Fig. 17.30a.

17.10. MATLAB Kalman Filter Simulation

For this purpose, the MATLAB GUI workspace entitled *"kalman_gui.m"* was developed. It is shown in Fig. 17.31. In this design, the inputs can be initialized to correspond to two target type kinematics (aircraft and missile). For example, when you click on the button *"ResetMissile,"* the initial *x-, y-,* and *z*-detection coordinates for the missile are loaded into the *"Starting Location"* field. The corresponding target velocity is also loaded in the *"velocity in x direction"* field. Finally, all other fields associated with the Kalman filter are also loaded using default values that are appropriate for this design case study. Note that the user can alter these entries as appropriate.

This program generates a fictitious default trajectory for the selected target type. This is accomplished using the function *"maketraj.m,"* listed in Appendix 17-A. Users can either use this program with its default trajectories, or import their own specific trajectory files. The function *"maketraj.m"* assumes constant altitude, and generates a maneuvering trajectory in the *x-y* plane, as shown in Fig. 17.32. This trajectory can be changed using the different fields in the *"trajectory Parameter"* fields.

Next the program corrupts the trajectory by adding white Gaussian noise. This is accomplished by the function *"addnoise.m,"* which is listed in Appendix 17-A. A six-state Kalman filter named *"kalfilt.m"* is then utilized to perform the tracking task. This function is also listed in Appendix 17-A.

The azimuth, elevation, and range errors are input to the program using their corresponding fields on the GUI. In the example used in this chapter, these entries are assumed constant throughout the simulation. In practice, this is not true and these values will change. They are calculated by the radar signal processor on a "per-processing-interval" basis and then are input into the tracker. For example, the standard deviation of the error in the range measurement is

$$\sigma_R = \frac{\Delta R}{\sqrt{2 \times SNR}} = \frac{c}{2B\sqrt{2 \times SNR}} \qquad \textbf{Eq. (17.161)}$$

where ΔR is the range resolution, c is the speed of light, B is the bandwidth, and SNR is the measurement SNR.

The standard deviation of the error in the velocity measurement is

$$\sigma_v = \frac{\lambda}{2\tau\sqrt{2 \times SNR}} \qquad \textbf{Eq. (17.162)}$$

where λ is the wavelength and τ is the uncompressed pulse width. The standard deviation of the error in the angle measurement is

$$\sigma_a = \frac{\Theta}{1.6\sqrt{2 \times SNR}} \qquad \textbf{Eq. (17.163)}$$

where Θ is the antenna beamwidth of the angular coordinate of the measurement (azimuth and elevation).

Table 17.1 lists the type of plots generated by this simulation. Figures 17.32 through Fig. 17.42 show typical outputs produced using this simulation, assuming the missile case, during any given run.

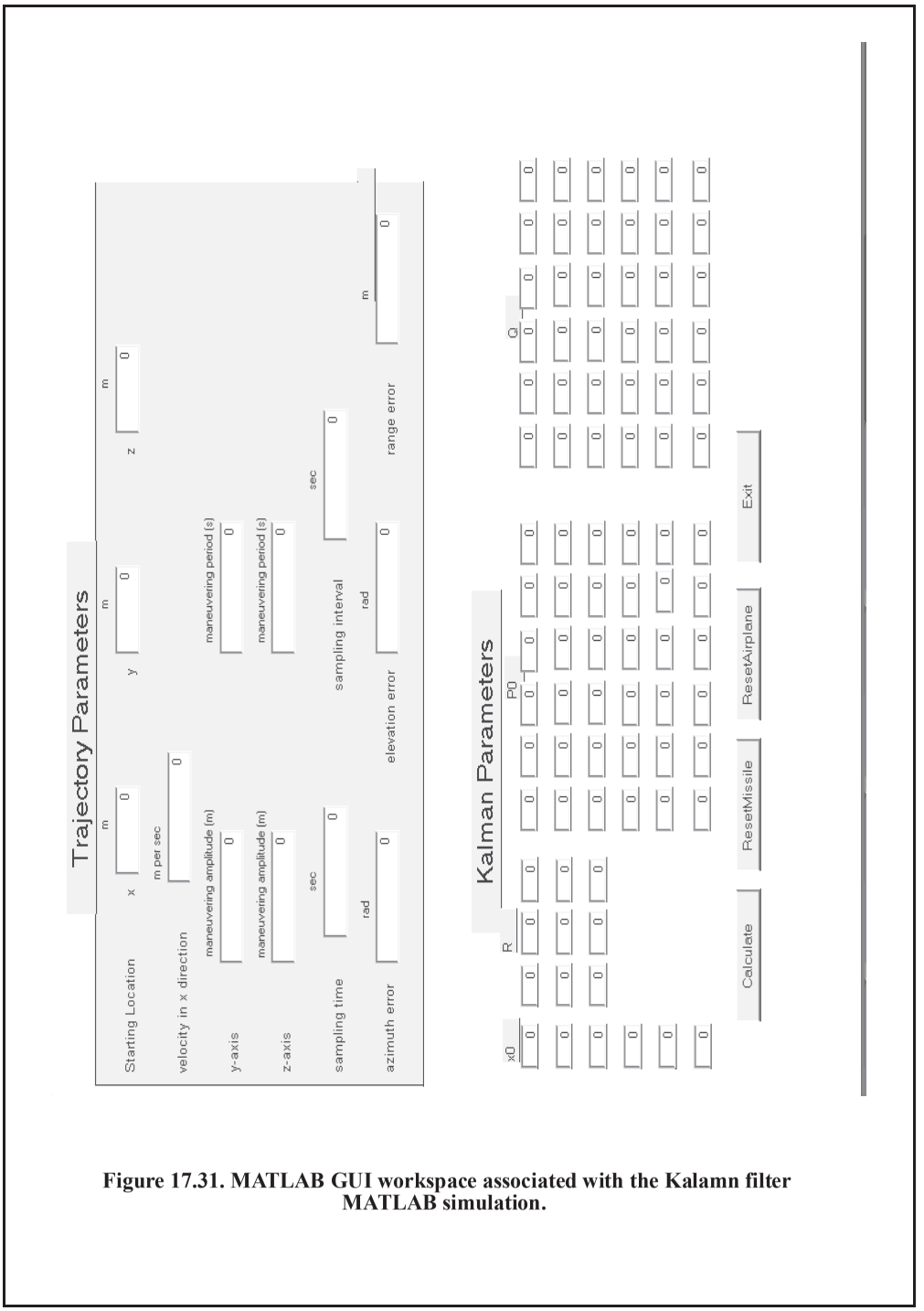


Figure 17.31. MATLAB GUI workspace associated with the Kalamn filter MATLAB simulation.

TABLE 17.1. Output list generated by the *"kalman_gui.m"* simulation

Figure #	Description
1	*uncorrupted input trajectory*
2	*corrupted input trajectory*
3	*corrupted and uncorrupted x-position*
4	*corrupted and uncorrupted y-position*
5	*corrupted and uncorrupted z-position*
6	*corrupted and filtered x-, y- and z- positions*
7	*predicted x-, y- and z- velocities*
8	*position residuals*
9	*velocity residuals*
10	*covariance matrix components versus time*
11	*Kalman filter gains versus time*

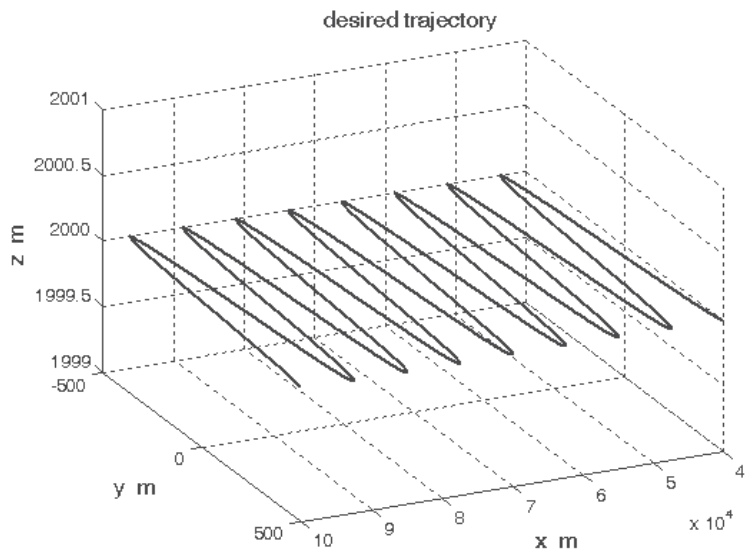


Figure 17.32. Default missile uncorrupted trajectory.

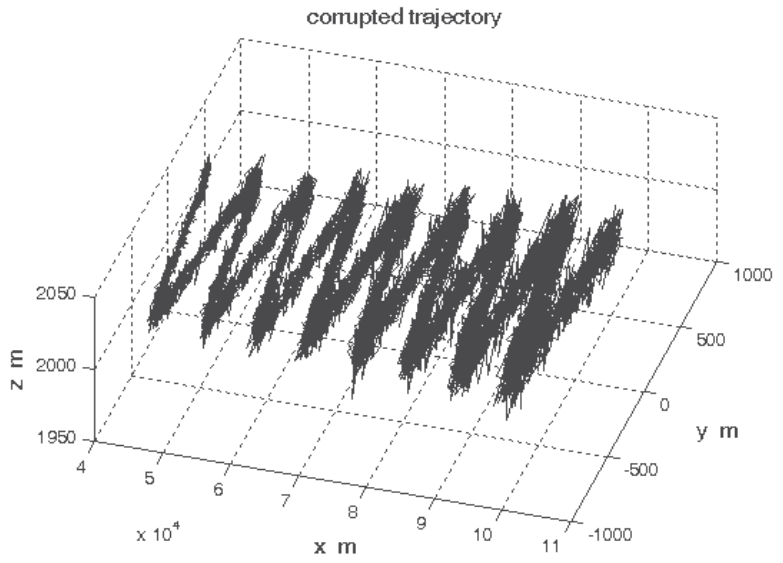


Figure 17.33. Default missile corrupted trajectory.

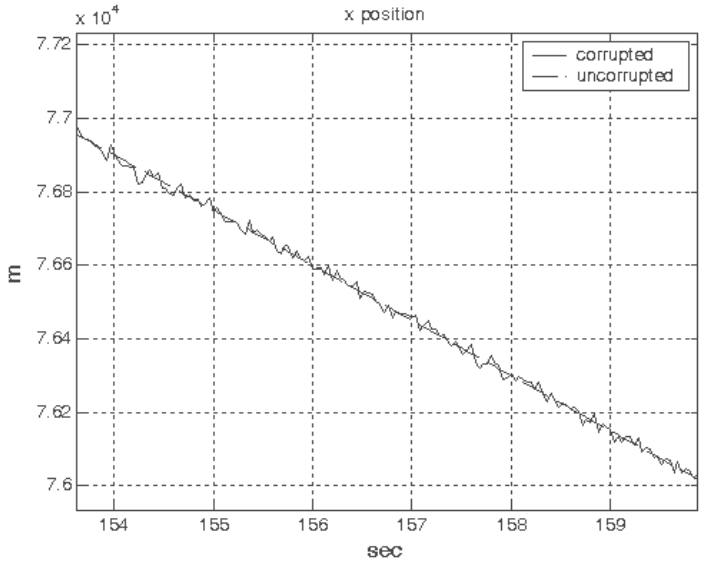


Figure 17.34. Missile x-position from 153 to 160 seconds.

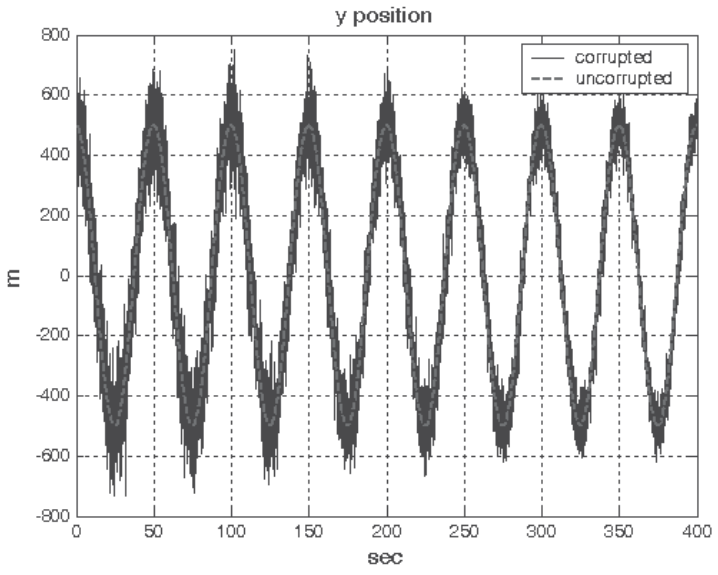


Figure 17.35. Missile y-position.

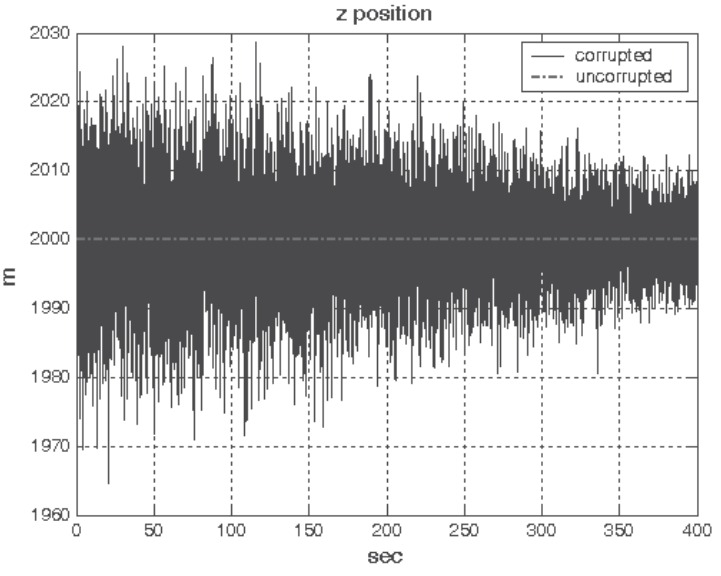


Figure 17.36. Missile z-position.

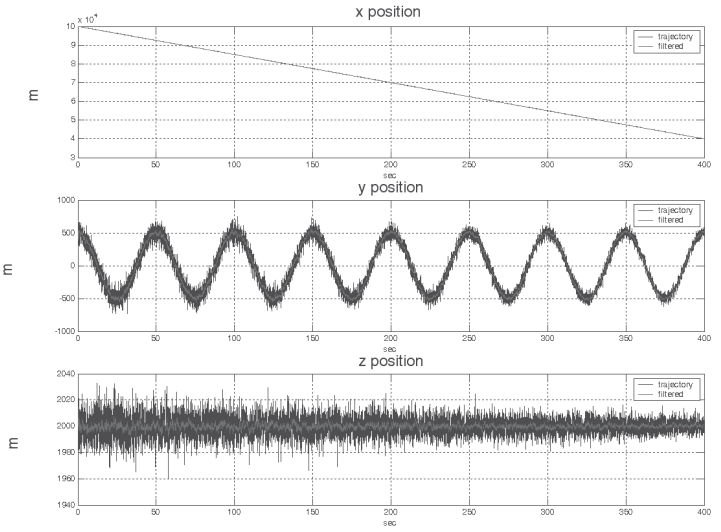


Figure 17.37. Missile trajectory and filtered trajectory.

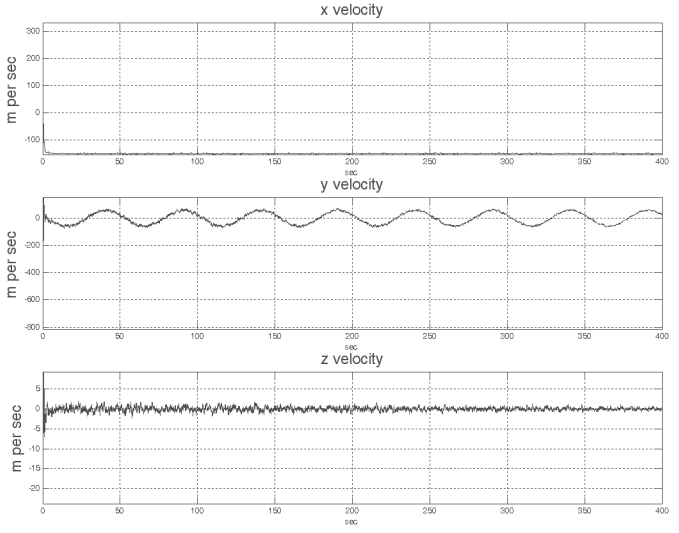


Figure 17.38. Missile velocity filtered.

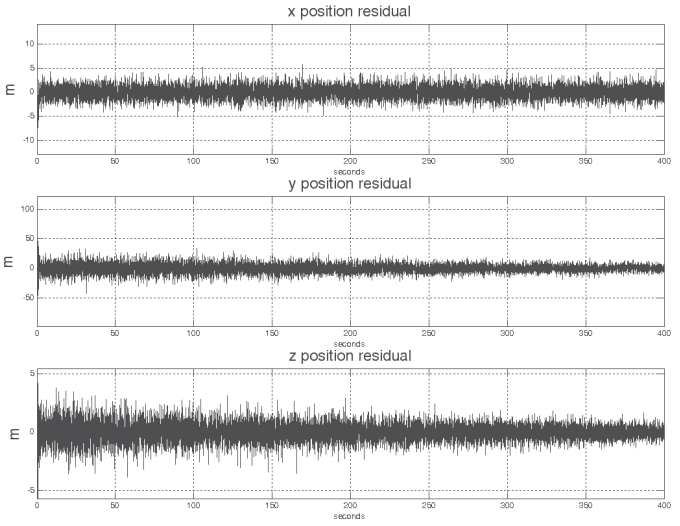


Figure 17.39. Missile position residuals.

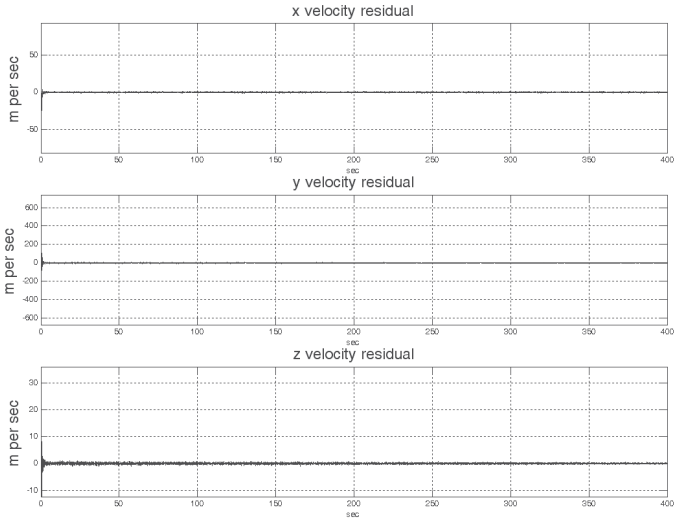


Figure 17.40. Missile velocity residuals.

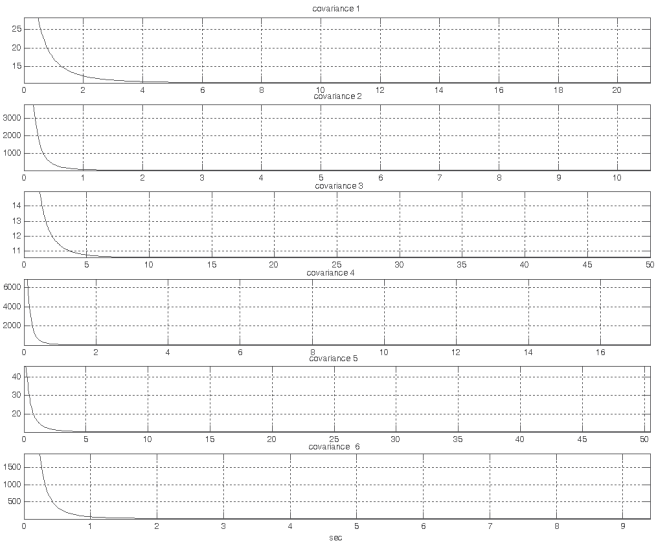


Figure 17.41. Missile covariance matrix components versus time.

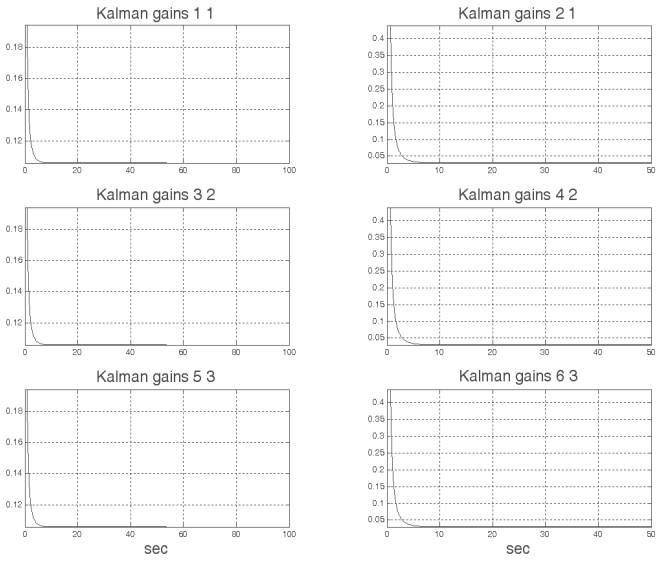


Figure 17.42. Kalman filter gains versus time.

Problems

17.1. Show that in order to be able to quickly achieve changing the beam position, the error signal needs to be a linear function of the deviation angle.

17.2. Prepare a short report on the vulnerability of conical scan to amplitude modulation jamming. In particular, consider the self-protecting technique called "Gain Inversion."

17.3. Consider a conical scan radar. The pulse repetition interval is $10\mu s$. Calculate the scan rate so that at least ten pulses are emitted within one scan.

17.4. Consider a conical scan antenna whose rotation around the tracking axis is completed in 4 seconds. If during this time 20 pulses are emitted and received, calculate the radar PRF and the unambiguous range.

17.5. Reproduce Fig. 17.11 for $\varphi_0 = 0.05, 0.1$ and $\varphi_0 = 0.15$ radians.

17.6. Reproduce Fig. 17.13 for the squint angles defined in the previous problem.

17.7. Derive Eq. (17.33) and Eq. (17.34).

17.8. Consider a monopulse radar where the input signal is comprised of both target return and additive white Gaussian noise. Develop an expression for the complex ratio Σ/Δ.

17.9. To generate the sum and difference patterns for a linear array of size N, follow this algorithm: To form the difference pattern, multiply the first $N/2$ elements by -1 and the second $N/2$ elements by +1. Plot the sum and difference patterns for a linear array of size 60.

17.10. Generate the delta/sum patterns for a 21-element linear array using the form

$$\frac{\Delta}{\Sigma} = j\frac{V_\Delta}{\sqrt{|V_\Delta|^2 + |V_\Sigma|^2}}$$

where V_Δ is the difference voltage pattern and V_Σ is the sum voltage pattern.

17.11. Consider the sum and difference signals defined in Eqs. (17.7) and (17.8). What is the squint angle φ_0 that maximizes $\Sigma(\varphi = 0)$?

17.12. A certain system is defined by the following difference equation:

$$y(n) + 4y(n-1) + 2y(n-2) = w(n)$$

Find the solution to this system for $n > 0$ and $w = \delta$.

17.13. Prove the state transition matrix properties (i.e., Eqs. (17.30) through (17.36)).

17.14. Suppose that the state equations for a certain discrete time LTI system are

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w(n).$$

If $y(0) = y(1) = 1$, find $y(n)$ when the input is a step function.

17.15. Derive Eq. (17.55).

17.16. Derive Eq. (17.75).

17.17. Using Eq. (17.83), compute a general expression (in terms of the transfer function) for the steady-state errors when the input sequence is:

$$u1 = \{0, 1, 1, 1, 1, \ldots\}$$

$$u2 = \{0, 1, 2, 3, \ldots\}$$

$$u3 = \{0, 1^2, 2^2, 3^2, \ldots\}$$

$$u4 = \{0, 1^3, 2^3, 3^3, \ldots\}$$

17.18. Verify the results in Eqs. (17.99) and (17.100).

17.19. Develop an expression for the steady-state error transfer function for an $\alpha\beta$ tracker.

17.20. Using the result of the previous problem and Eq. (17.83), compute the steady-state errors for the $\alpha\beta$ tracker with the inputs defined in Problem 17.13.

17.21. Design a critically damped $\alpha\beta$, when the measurement noise variance associated with position is $\sigma_v^2 = 50m$ and when the desired standard deviation of the filter prediction error is $5.5m$.

17.22. Derive Eqs. (17.118) through (17.120).

17.23. Derive Eq. (17.122).

17.24. Consider a $\alpha\beta\gamma$ filter. We can define six transfer functions: $H_1(z)$, $H_2(z)$, $H_3(z)$, $H_4(z)$, $H_5(z)$, and $H_6(z)$ (predicted position, predicted velocity, predicted acceleration, smoothed position, smoothed velocity, and smoothed acceleration). Each transfer function has the form

$$H(z) = \frac{a_3 + a_2 z^{-1} + a_1 z^{-2}}{1 + b_2 z^{-1} + b_1 z^{-2} + b_0 z^{-3}}.$$

The denominator remains the same for all six transfer functions. Compute all the relevant coefficients for each transfer function.

17.25. Verify the results obtained for the two limiting cases of the Singer-Kalman filter.

17.26. Verify Eq. (17.160).

Appendix 17-A: Chapter 17 MATLAB Code Listings

The MATLAB code provided in this chapter was designed as an academic standalone tool and is not adequate for other purposes. The code was written in a way to assist the reader in gaining a better understanding of the theory. The code was not developed, nor is it intended to be used as part of an open-loop or a closed-loop simulation of any kind. The MATLAB code found in this textbook can be downloaded from this book's web page on the CRC Press website. Simply use your favorite web browser, go to www.crcpress.com, and search for keyword *"Mahafza"* to locate this book's web page.

MATLAB Function "mono_pulse.m" Listing

```
function mono_pulse(phi0)
eps = 0.0000001;
angle = -pi:0.01:pi;
y1 = sinc(angle + phi0);
y2 = sinc((angle - phi0));
ysum = y1 + y2;
ydif = -y1 + y2;
figure (1)
plot (angle,y1,'k',angle,y2,'k');
grid;
xlabel ('Angle - radians')
ylabel ('Squinted patterns')
figure (2)
plot(angle,ysum,'k');
grid;
xlabel ('Angle - radians')
ylabel ('Sum pattern')
figure (3)
plot (angle,ydif,'k');
grid;
xlabel ('Angle - radians')
ylabel ('Difference pattern')
angle = -pi/4:0.01:pi/4;
y1 = sinc(angle + phi0);
y2 = sinc((angle - phi0));
ydif = -y1 + y2;
ysum = y1 + y2;
dovrs = ydif ./ ysum;
figure(4)
plot (angle,dovrs,'k');
grid;
xlabel ('Angle - radians')
ylabel ('voltage gain')
```

MATLAB Function "ghk_tracker.m" Listing

```
function [residual, estimate] = ghk_tracker (X0, smoocof, inp, npts, T, nvar)
rn = 1.;
% read the initial estimate for the state vector
X = X0;
theta = smoocof;
%compute values for alpha, beta, gamma
```

```
w1 = 1. - (theta^3);
w2 = 1.5 * (1. + theta) * ((1. - theta)^2) / T;
w3 = ((1. - theta)^3) / (T^2);
% setup the transition matrix PHI
PHI = [1. T (T^2)/2.;0. 1. T;0. 0. 1.];
while rn < npts ;
  %use the transition matrix to predict the next state
  XN = PHI * X;
  error = (inp(rn) + normrnd(0,nvar)) - XN(1);
  residual(rn) = error;
  tmp1 = w1 * error;
  tmp2 = w2 * error;
  tmp3 = w3 * error;
  % compute the next state
  X(1) = XN(1) + tmp1;
  X(2) = XN(2) + tmp2;
  X(3) = XN(3) + tmp3;
  estimate(rn) = X(1);
  rn = rn + 1.;
end
return
```

MATLAB Function "ghk_tracker1.m" Listing

```
function [residual, estimate] = ghk_tracker1 (X0, smoocof, inp, npts, T)
rn = 1.;
% read the initial estimate for the state vector
X = X0;
theta = smoocof;
%compute values for alpha, beta, gamma
w1 = 1. - (theta^3);
w2 = 1.5 * (1. + theta) * ((1. - theta)^2) / T;
w3 = ((1. - theta)^3) / (T^2);
% setup the transition matrix PHI
PHI = [1. T (T^2)/2.;0. 1. T;0. 0. 1.];
while rn < npts ;
  %use the transition matrix to predict the next state
  XN = PHI * X;
  error = inp(rn)  - XN(1);
  residual(rn) = error;
  tmp1 = w1 * error;
  tmp2 = w2 * error;
  tmp3 = w3 * error;
  % compute the next state
  X(1) = XN(1) + tmp1;
  X(2) = XN(2) + tmp2;
  X(3) = XN(3) + tmp3;
  estimate(rn) = X(1);
  rn = rn + 1.;
end
return
```

MATLAB Program "Fig17_20s.m" Listing

```
clear all
eps = 0.0000001;
npts = 5000;
del = 1./ 5000.;
t = 0. : del : 1.;
% generate input sequence
inp = 1.+ t.^3 + .5 .*t.^2 + cos(2.*pi*10 .* t) ;
% read the initial estimate for the state vector
X0 = [2,.1,.01]';
% this is the update interval in seconds
T = 100. * del;
% this is the value of the smoothing coefficient
xi = .91;
[residual, estimate] = ghk_tracker (X0, xi, inp, npts, T, .01);
figure(1)
plot (residual(1:500))
xlabel ('Sample number')
ylabel ('Residual error')
grid
figure(2)
NN = 4999.;
n = 1:NN;
plot (n,estimate(1:NN),'b',n,inp(1:NN),'r')
xlabel ('Sample number')
ylabel ('Position')
legend ('Estimated','Input')
```

MATLAB Function "kalman_filter.m" Listing

```
function [residual, estimate] = kalman_filter(npts, T, X0, inp, R, nvar)
N = npts;
rn=1;
% read the initial estimate for the state vector
X = X0;
% it is assumed that the measurement vector H=[1,0,0]
% this is the state noise variance
VAR = nvar;
% setup the initial value for the prediction covariance.
S = [1. 1. 1.; 1. 1. 1.; 1. 1. 1.];
% setup the transition matrix PHI
PHI = [1. T (T^2)/2.; 0. 1. T; 0. 0. 1.];
% setup the state noise covariance matrix
Q(1,1) = (VAR * (T^5)) / 20.;
Q(1,2) = (VAR * (T^4)) / 8.;
Q(1,3) = (VAR * (T^3)) / 6.;
Q(2,1) = Q(1,2);
Q(2,2) = (VAR * (T^3)) / 3.;
Q(2,3) = (VAR * (T^2)) / 2.;
Q(3,1) = Q(1,3);
Q(3,2) = Q(2,3);
Q(3,3) = VAR * T;
while rn < N ;
  %use the transition matrix to predict the next state
```

```
  XN = PHI * X;
  % Perform error covariance extrapolation
  S = PHI * S * PHI' + Q;
  % compute the Kalman gains
  ak(1) = S(1,1) / (S(1,1) + R);
  ak(2) = S(1,2) / (S(1,1) + R);
  ak(3) = S(1,3) / (S(1,1) + R);
  %perform state estimate update:
  error = inp(rn) + normrnd(0,R) - XN(1);
  residual(rn) = error;
  tmp1 = ak(1) * error;
  tmp2 = ak(2) * error;
  tmp3 = ak(3) * error;
  X(1) = XN(1) + tmp1;
  X(2) = XN(2) + tmp2;
  X(3) = XN(3) + tmp3;
  estimate(rn) = X(1);
  % update the error covariance
  S(1,1) = S(1,1) * (1. -ak(1));
  S(1,2) = S(1,2) * (1. -ak(1));
  S(1,3) = S(1,3) * (1. -ak(1));
  S(2,1) = S(1,2);
  S(2,2) = -ak(2) * S(1,2) + S(2,2);
  S(2,3) = -ak(2) * S(1,3) + S(2,3);
  S(3,1) = S(1,3);
  S(3,3) = -ak(3) * S(1,3) + S(3,3);
  rn = rn + 1.;
end
```

MATLAB Program "Fig17-29.m" Listing

```
% generates Fig 17.29
clc
close ll
clear all
npts = 2000;
del = 1/2000;
t = 0:del:1;
inp = (1+.2 .* t + .1 .*t.^2);% + cos(2. * pi * 2.5 .* t);
X0 = [1,.1,.01]';
% it is assumed that the measurmeny vector H=[1,0,0]
% this is the update interval in seconds
T = 1.;
% enter the mesurement noise variance
R = .01;
% this is the state noise variance
nvar = .18;
[residual, estimate] = kalman_filter(npts, T, X0, inp, R, nvar);
figure(1)
plot(residual(1:500),'k')
xlabel ('Sample number')
ylabel ('Residual')
figure(2)
subplot(2,1,1)
```

```
plot(inp,'k')
axis tight
ylabel ('position - truth')
subplot(2,1,2)
plot(estimate,'k')
axis tight
xlabel ('Sample number')
ylabel ('Predicted position')
```

MATLAB Program "Fig17_30.m" Listing

```
% generates Fig 17.30 of text
clc
close all
clear all
npts = 2000;
del = 1/2000;
t = 0:del:1;
inp = (1+.2 .* t + .1 .*t.^2) + cos(2. * pi * 2.5 .* t);
X0 = [1,.1,.01]';
% it is assumed that the measurement vector H=[1,0,0]
% this is the update interval in seconds
T = 1.;
% enter the mesurement noise variance
R = .035;
% this is the state noise variance
nvar = .5;
[residual, estimate] = kalman_filter(npts, T, X0, inp, R, nvar);
figure(1)
plot(residual,'k')
xlabel ('Sample number')
ylabel ('Residual')
figure(2)
subplot(2,1,1)
plot(inp,'k')
axis tight
ylabel ('position - truth')
subplot(2,1,2)
plot(estimate,'k')
axis tight
xlabel ('Sample number')
ylabel ('Predicted position')
```

MATLAB Function "maketraj.m" Listing

```
function [times , trajectory] = maketraj(start_loc, xvelocity, yamp, yperiod, zamp, zperiod, samplingtime,
deltat)
% maketraj.m
% USAGE:  [times , trajectory] = maketraj(start_loc, xvelocity, yamp, yperiod, zamp, zperiod, sampling-
time, deltat)
% NOTE: all coordinates are in radar reference coordinates.
% INPUTS
% name        dimension explanation                        units
%------      ------   ---------------                    -------
```

```
% start_loc     3 X 1    starting location of target            m
% xvelocity     1        velocity of target                     m/s
% yamp          1        amplitude of oscillation y direction   m
% yperiod       1        period of oscillation y direction      m
% zamp          1        amplitude of oscillation z direction   m
% zperiod       1        period of oscillation z direction      m
% samplingtime  1        length of interval of trajectory       sec
% deltat        1        time between samples                   sec
%
% OUTPUTS
%
% name         dimension              explanation              units
%------       ----------             ---------------          ------
% times        1 X samplingtime/deltat vector of times
%                                     corresponding to samples sec
% trajectory   3 X samplingtime/deltat trajectory x,y,z        m
%
times = 0: deltat: samplingtime ;
x = start_loc(1)+xvelocity.*times ;
if yperiod~=0
  y = start_loc(2)+yamp*cos(2*pi*(1/yperiod).*times) ;
else
  y = ones(1, length(times))*start_loc(2) ;
end
if zperiod~=0
  z = start_loc(3)+zamp*cos(2*pi*(1/zperiod).*times)  ;
else
  z = ones(1, length(times))*start_loc(3) ;
end
trajectory = [x ; y  ; z] ;
```

MATLAB Function "addnoise.m" Listing

```
function [noisytraj ] = addnoise(trajectory, sigmaaz, sigmael, sigmarange )
% addnoise.m
% USAGE: [noisytraj ] = addnoise(trajectory, sigmaaz, sigmael, sigmarange )
% INPUTS
% name         dimension  explanation                          units
%------       ------     ---------------                      -------
% trajectory   3 X POINTS trajectory in radar reference coords    [m;m;m]
% sigmaaz      1          standard deviation of azimuth error     radians
% sigmael      1          standard deviation of elevation error   radians
% sigmarange   1          standard deviation of range error       m
%
% OUTPUTS
% name         dimension   explanation                          units
%------       ------      ---------------                      -------
% noisytraj    3 X POINTS  noisy trajectory                       [m;m;m]
noisytraj = zeros(3, size(trajectory,2)) ;

for loop = 1 : size(trajectory,2)
  x = trajectory(1,loop);
  y = trajectory(2,loop);
  z = trajectory(3,loop);
```

```
  azimuth_corrupted =  atan2(y,x) + sigmaaz*randn(1) ;
  elevation_corrupted = atan2(z, sqrt(x^2+y^2)) + sigmael*randn(1) ;
  range_corrupted = sqrt(x^2+y^2+z^2)  + sigmarange*randn(1) ;
  x_corrupted = range_corrupted*cos(elevation_corrupted)*cos(azimuth_corrupted) ;
  y_corrupted = range_corrupted*cos(elevation_corrupted)*sin(azimuth_corrupted) ;
  z_corrupted = range_corrupted*sin(elevation_corrupted) ;
  noisytraj(:,loop) = [x_corrupted ; y_corrupted; z_corrupted ] ;
end % next loop
```

MATLAB Function "kalfilt.m" Listing

```
function [filtered, residuals , covariances, kalmgains] = kalfilt(trajectory, x0, P0, phi, R, Q )
% kalfilt.m
% USAGE: [filtered, residuals , covariances, kalmgains] = kalfilt(trajectory, x0, P0, phi, R, Q)
%
% INPUTS
% name        dimension                  explanation                              units
%------       ------                  ---------------                          -------
% trajectory      NUMMEASUREMENTS  X  NUMPOINTS    trajectory in radar reference coords
[m;m;m]
% x0          NUMSTATES X 1               initial estimate of state vector          m, m/s
% P0          NUMSTATES X NUMSTATES       initial estimate of covariance matrix       m, m/s
% phi         NUMSTATES X NUMSTATES       state transition matrix                   -
% R           NUMMEASUREMENTS X NUMMEASUREMENTS   measurement error covariance matrix
m
% Q           NUMSTATES X NUMSTATES       state error covariance matrix             m, m/s
%
% OUTPUTS
% name        dimension                  explanation                              units
%------       ------                  ---------------                          -------
% filtered    NUMSTATES X NUMPOINTS       filtered trajectory x,y,z pos, vel    [m; m/s; m; m/s; m; m/s]
% residuals   NUMSTATES X NUMPOINTS       residuals of filtering                [m;m;m]
% covariances  NUMSTATES X NUMPOINTS      diagonal of covariance matrix         [m;m;m]
% kalmgains   (NUMSTATES X NUMMEASUREMENTS)
%              X NUMPOINTS              Kalman gain matrix                       -
NUMSTATES = 6 ;
NUMMEASUREMENTS = 3 ;
NUMPOINTS = size(trajectory, 2) ;
% initialize output matrices
filtered = zeros(NUMSTATES, NUMPOINTS) ;
residuals = zeros(NUMSTATES, NUMPOINTS) ;
covariances = zeros(NUMSTATES, NUMPOINTS) ;
kalmgains = zeros(NUMSTATES*NUMMEASUREMENTS, NUMPOINTS) ;
% set matrix relating measurements to states
H = [1 0 0 0 0 0 ; 0 0 1 0 0 0 ; 0 0 0 0 1 0];
xhatminus = x0 ;
Pminus = P0 ;
 for loop = 1: NUMPOINTS
   % compute the Kalman gain
  K = Pminus*H'*inv(H*Pminus*H' + R) ;
  kalmgains(:,loop) = reshape(K, NUMSTATES*NUMMEASUREMENTS, 1) ;
  % update the estimate with the measurement z
  z = trajectory(:,loop) ;
  xhat = xhatminus + K*(z - H*xhatminus) ;
```

```
  filtered(:,loop) = xhat ;
  residuals(:,loop) = xhat - xhatminus ;
  % update the error covariance for the updated estimate
  P = ( eye(NUMSTATES, NUMSTATES) - K*H)*Pminus ;
  covariances(:,loop) = diag(P) ;  % only save diagonal of covariance matrix
  % project ahead
  xhatminus_next = phi*xhat ;
  Pminus_next = phi*P*phi' + Q ;
   xhatminus = xhatminus_next ;
  Pminus = Pminus_next ;
end
```

Chapter 18

Tactical Synthetic Aperture Radars

This chapter was coauthored with Brian J. Smith.[1]

This chapter provides an introduction to Tactical Synthetic Aperture Radar (TSAR). The purpose of this chapter is to further develop the readers' understanding of SAR by taking a closer look at high resolution spotlight SAR image formation algorithms, motion compensation techniques, autofocus algorithms, and performance metrics.

18.1. Introduction

Modern airborne radar systems are designed to perform a large number of functions which range from detection and discrimination of targets to mapping large areas of ground terrain. This mapping can be performed by the Synthetic Aperture Radar (SAR). Through illuminating the ground with coherent radiation and measuring the echo signals, SAR can produce high resolution two-dimensional (and in some cases three-dimensional) imagery of the ground surface. The quality of ground maps generated by SAR is determined by the size of the resolution cell. A resolution cell is specified by both range and azimuth resolutions of the system. Other factors affecting the size of the resolution cells are (1) size of the processed map and the amount of signal processing involved; (2) cost consideration; and (3) size of the objects that need to be resolved in the map. For example, mapping gross features of cities and coastlines does not require as much resolution when compared to resolving houses, vehicles, and streets.

SAR systems can produce maps of reflectivity versus range and Doppler (cross range). Range resolution is accomplished through range gating. Fine range resolution can be accomplished by using pulse compression techniques. The azimuth resolution depends on antenna size and radar wavelength. Fine azimuth resolution is enhanced by taking advantage of the radar motion in order to synthesize a larger antenna aperture. Let N_r denote the number of range bins and let N_a denote the number of azimuth cells. It follows that the total number of resolution cells in the map is $N_r N_a$. SAR systems that are generally concerned with improving azimuth resolution are often referred to as Doppler Beam-Sharpening (DBS) SARs. In this case, each range bin is processed to resolve targets in Doppler which corresponds to azimuth. This chapter is presented in the context of DBS.

1. Dr. Brian J. Smith is with the US Army Aviation and Missile Command (AMCOM), Redstone Arsenal, Alabama.

Due to the large amount of signal processing required in SAR imagery, the early SAR designs implemented optical processing techniques. Although such optical processors can produce high-quality radar images, they have several shortcomings. They can be very costly and are, in general, limited to making strip maps. Motion compensation is not easy to implement for radars that utilize optical processors. With the recent advances in solid state electronics and Very Large Scale Integration (VLSI) technologies, digital signal processing in real time has been made possible in SAR systems.

18.1.1. Side Looking SAR Geometry

Fig. 18.1 shows the geometry of the standard side looking SAR. We will assume that the platform carrying the radar maintains both fixed altitude h and velocity v. The antenna $3dB$ beamwidth is θ, and the elevation angle (measured from the z-axis to the antenna axis) is β. The intersection of the antenna beam with the ground defines a footprint. As the platform moves, the footprint scans a swath on the ground.

The radar position with respect to the absolute origin $\vec{O} = (0, 0, 0)$, at any time is the vector $\vec{a}(t)$. The velocity vector $\vec{a}'(t)$ is

$$\vec{a}'(t) = 0 \times \hat{a}_x + v \times \hat{a}_y + 0 \times \hat{a}_z . \qquad \textbf{Eq. (18.1)}$$

The Line of Sight (LOS) for the current footprint centered at $\vec{q}(t_c)$ is defined by the vector $\vec{R}(t_c)$, where t_c denotes the central time of the observation interval T_{ob} (coherent integration interval). More precisely,

$$(t = t_a + t_c) \;\; ; \;\; -\frac{T_{ob}}{2} \le t \le \frac{T_{ob}}{2} \qquad \textbf{Eq. (18.2)}$$

where t_a and t are the absolute and relative times, respectively. The vector $\vec{m}_g$ defines the ground projection of the antenna at central time. The minimum slant range to the swath is R_{min}, and the maximum range is denoted R_{max}, as illustrated by Fig. 18.2. It follows that

$$\begin{aligned} R_{min} &= h/\cos(\beta - \theta/2) \\ R_{max} &= h/\cos(\beta + \theta/2) \\ \left|\vec{R}(t_c)\right| &= h/\cos\beta \end{aligned} \qquad \textbf{Eq. (18.3)}$$

Notice that the elevation angle β is equal to

$$\beta = 90 - \psi_g \qquad \textbf{Eq. (18.4)}$$

where ψ_g is the grazing angle. The size of the footprint is a function of the grazing angle and the antenna beamwidth, as illustrated in Fig. 18.3. The SAR geometry described in this section is referred to as SAR "strip mode" of operation. Another SAR mode of operation, which will not be discussed in this chapter, is called "spot-light mode," where the antenna is steered (mechanically or electronically) to continuously illuminate one spot (footprint) on the ground. In this case, one high resolution image of the current footprint is generated during an observation interval.

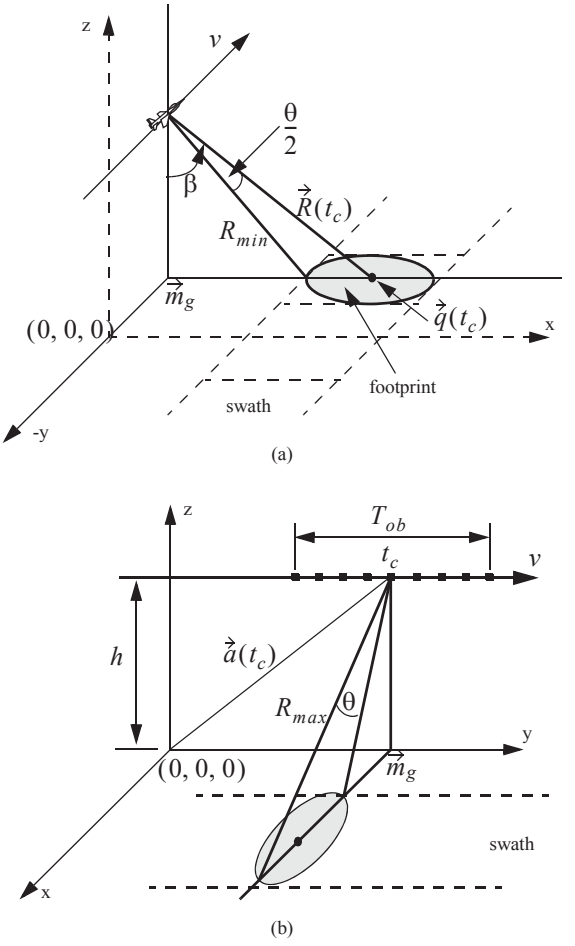


Figure 18.1. Side looking SAR geometry.

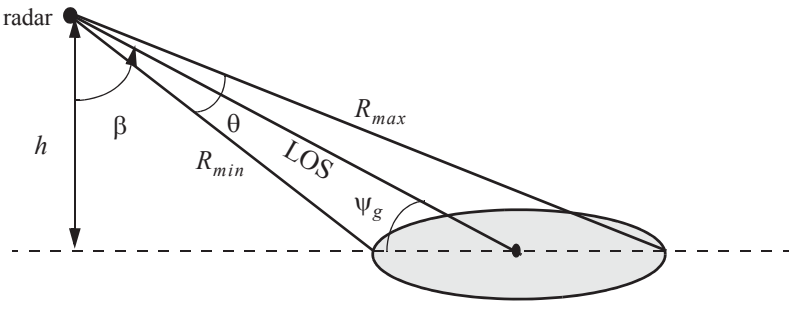


Figure 18.2. Definition of minimum and maximum range.

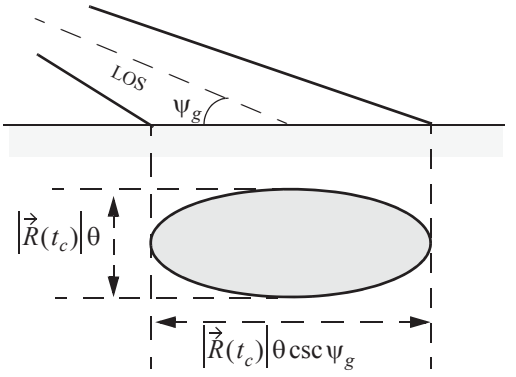


Figure 18.3. Footprint definition.

18.2. SAR Design Considerations

The quality of SAR images is heavily dependent on the size of the map resolution cell shown in Fig. 18.4. The range resolution, ΔR, is computed on the beam LOS, and is given by

$$\Delta R = (c\tau)/2 \qquad \textbf{Eq. (18.5)}$$

where τ is the pulse width. From the geometry in Fig. 18.5, the extent of the range cell ground projection ΔR_g is computed as

$$\Delta R_g = \frac{c\tau}{2}\sec\psi_g . \qquad \textbf{Eq. (18.6)}$$

The azimuth or cross range resolution for a real antenna with a $3dB$ beamwidth θ (radians) at range R is

$$\Delta A = \theta R . \qquad \textbf{Eq. (18.7)}$$

However, the antenna beamwidth is proportional to the aperture size,

$$\theta \approx \frac{\lambda}{L} \qquad \textbf{Eq. (18.8)}$$

where λ is the wavelength and L is the aperture length. It follows that

$$\Delta A = \frac{\lambda R}{L} . \qquad \textbf{Eq. (18.9)}$$

And since the effective synthetic aperture size is twice that of a real array, the azimuth resolution for a synthetic array is then given by

$$\Delta A = \frac{\lambda R}{2L} . \qquad \textbf{Eq. (18.10)}$$

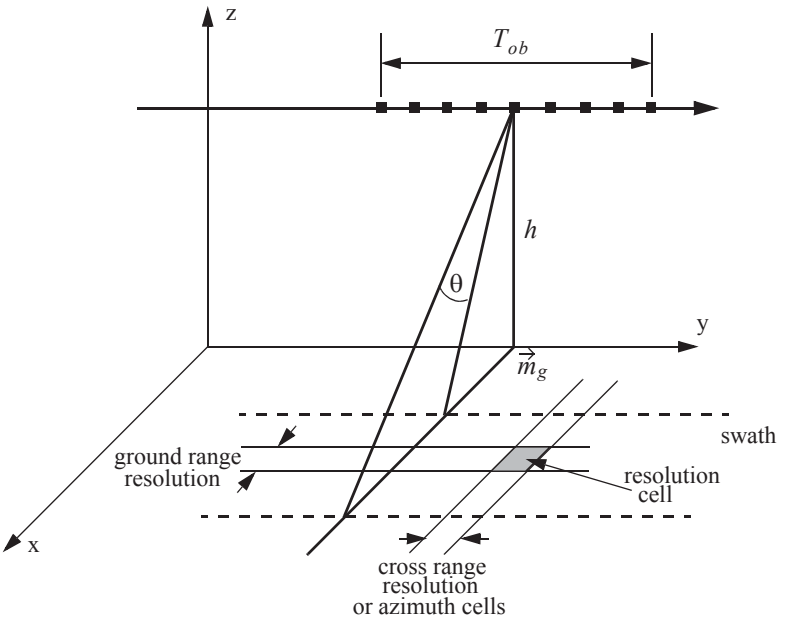


Figure 18.4a. Definition of a resolution cell.

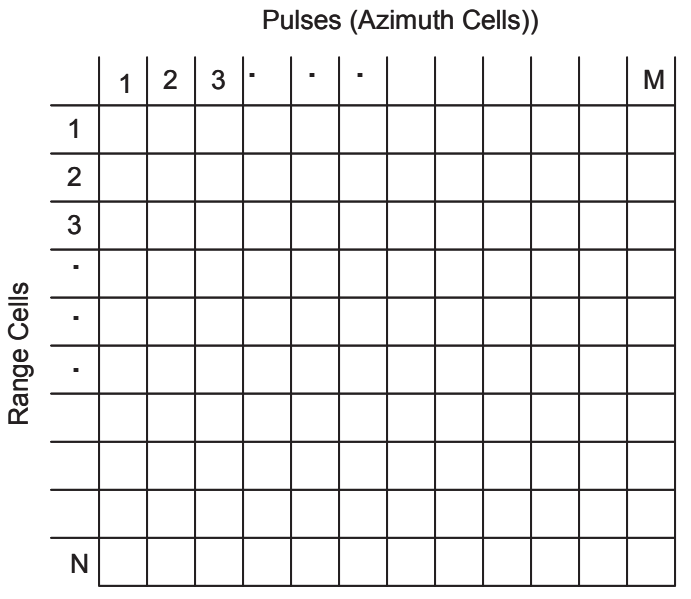


Figure 18.4b. Definition of a resolution cell.

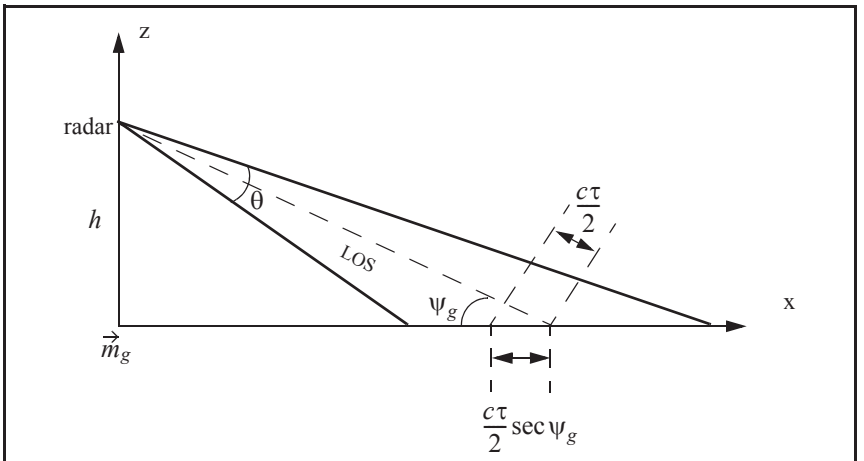


Figure 18.5. Definition of a range cell on the ground.

Furthermore, since the synthetic aperture length L is equal to vT_{ob}, Eq. (18.10) can be rewritten as

$$\Delta A = \frac{\lambda R}{2vT_{ob}}. \qquad \textbf{Eq. (18.11)}$$

The azimuth resolution can be greatly improved by taking advantage of the Doppler variation within a footprint (or a beam). As the radar travels along its flight path, the radial velocity to a ground scatterer (point target) within a footprint varies as a function of the radar radial velocity in the direction of that scatterer. The variation of Doppler frequency for a certain scatterer is called the "Doppler history."

Let $R(t)$ denote the range to a scatterer at time t, and v_r be the corresponding radial velocity; thus the Doppler shift is

$$f_d = -\frac{2\dot{R}(t)}{\lambda} = \frac{2v_r}{\lambda} \qquad \textbf{Eq. (18.12)}$$

where $\dot{R}(t)$ is the range rate to the scatterer. Let t_1 and t_2 be the times when the scatterer enters and leaves the radar beam, respectively, and t_c be the time that corresponds to minimum range. Fig. 18.6 shows a sketch of the corresponding $R(t)$. Since the radial velocity can be computed as the derivative of $R(t)$ with respect to time, one can clearly see that Doppler frequency is maximum at t_1, zero at t_c, and minimum at t_2, as illustrated in Fig. 18.7.

In general, the radar maximum PRF, $f_{r_{max}}$, must be low enough to avoid range ambiguity. Alternatively, the minimum PRF, $f_{r_{min}}$, must be high enough to avoid Doppler ambiguity. SAR unambiguous range must be at least as wide as the extent of a footprint. More precisely, since target returns from maximum range due to the current pulse must be received by the radar before the next pulse is transmitted, it follows that SAR unambiguous range is given by

$$R_u = R_{max} - R_{min}. \qquad \textbf{Eq. (18.13)}$$

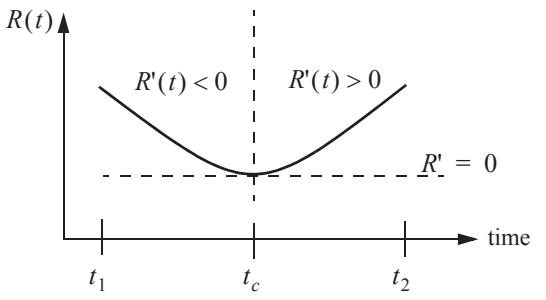


Figure 18.6. Sketch of range versus time for a scatterer.

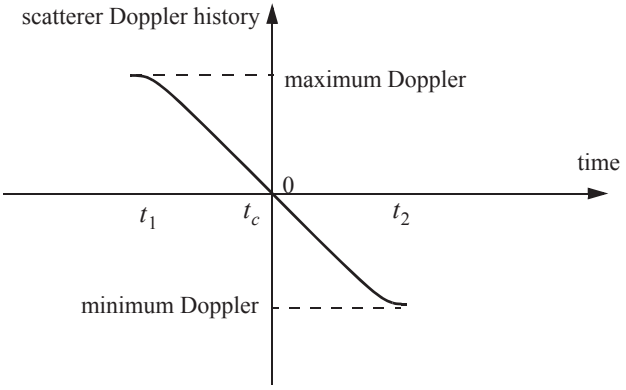


Figure 18.7. Point scatterer Doppler history.

An expression for unambiguous range was derived in Chapter 1, and is repeated here as Eq. (18.14),

$$R_u = \frac{c}{2f_r}. \qquad \textbf{Eq. (18.14)}$$

Combining Eq. (18.14) and Eq. (18.13) yields

$$f_{r_{max}} \leq \frac{c}{2(R_{max} - R_{min})}. \qquad \textbf{Eq. (18.15)}$$

SAR minimum PRF, $f_{r_{min}}$, is selected so that Doppler ambiguity is avoided. In other words, $f_{r_{min}}$ must be greater than the maximum expected Doppler spread within a footprint. From the geometry of Fig. 18.8, the maximum and minimum Doppler frequencies are, respectively, given by

$$f_{d_{max}} = \frac{2v}{\lambda} \sin\left(\frac{\theta}{2}\right) \sin\beta \;\; ; \;\; at \;\; t_1 \qquad \textbf{Eq. (18.16)}$$

$$f_{d_{min}} = -\frac{2v}{\lambda} \sin\left(\frac{\theta}{2}\right) \sin\beta \;\; ; \;\; at \;\; t_2 \,. \qquad \textbf{Eq. (18.17)}$$

It follows that the maximum Doppler spread is

$$\Delta f_d = f_{d_{max}} - f_{d_{min}} \,. \qquad \textbf{Eq. (18.18)}$$

Substituting Eqs. (18.16) and (18.17) into Eq. (18.18) and applying the proper trigonometric identities yield

$$\Delta f_d = \frac{4v}{\lambda} \sin\frac{\theta}{2} \sin\beta \,. \qquad \textbf{Eq. (18.19)}$$

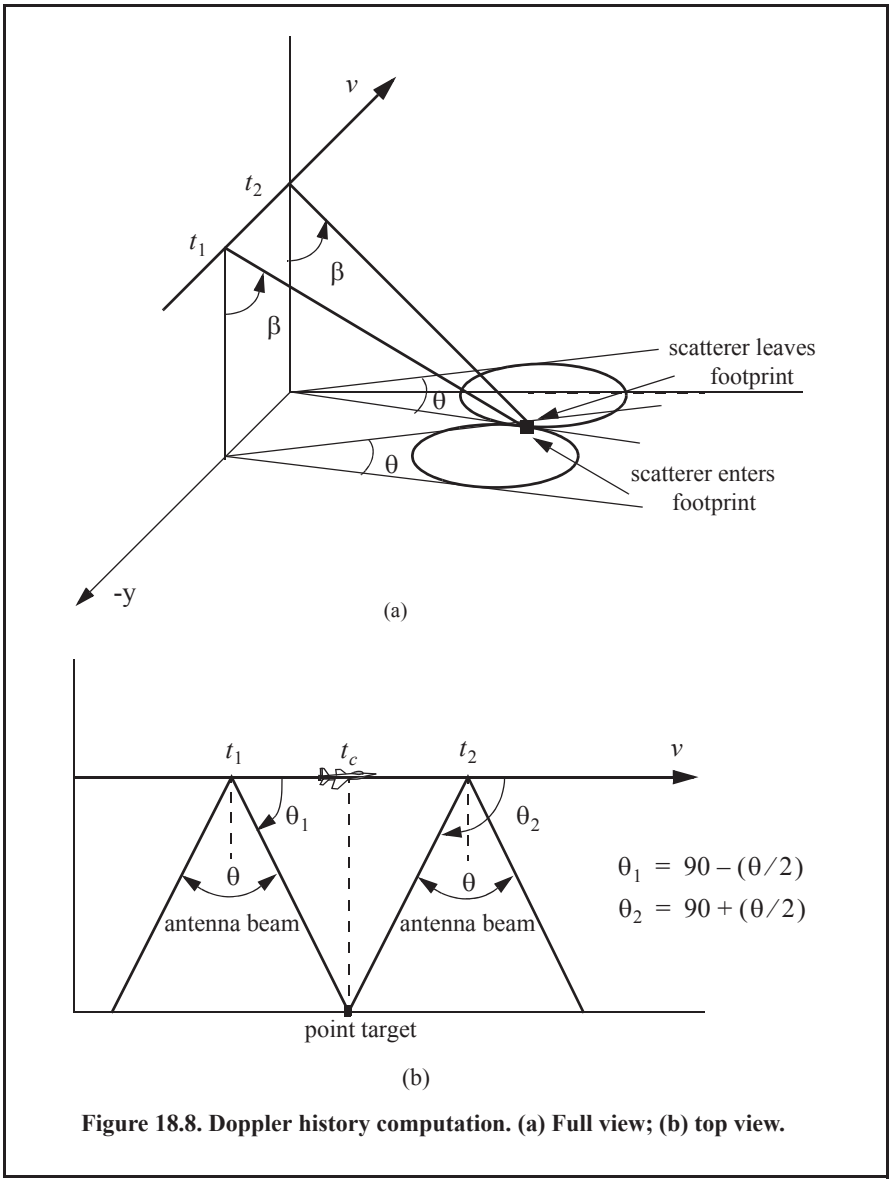


Figure 18.8. Doppler history computation. (a) Full view; (b) top view.

Finally, by using the small angle approximation we get

$$\Delta f_d \approx \frac{4v}{\lambda}\ \frac{\theta}{2}\sin\beta\ =\ \frac{2v}{\lambda}\ \theta\sin\beta\,. \qquad \textbf{Eq. (18.20)}$$

Therefore, the minimum PRF is

$$f_{r_{min}} \geq \frac{2v}{\lambda}\ \theta\sin\beta\,. \qquad \textbf{Eq. (18.21)}$$

Combining Eqs. (18.15) and (18.21) we get

$$\frac{c}{2(R_{max}-R_{min})} \geq f_r \geq \frac{2v}{\lambda}\ \theta\sin\beta\,. \qquad \textbf{Eq. (18.22)}$$

It is possible to resolve adjacent scatterers at the same range within a footprint based only on the difference of their Doppler histories. For this purpose, assume that the two scatterers are within the kth range bin.

Denote their angular displacement as $\Delta\theta$, and let $\Delta f_{d_{min}}$ be the minimum Doppler spread between the two scatterers such that they will appear in two distinct Doppler filters. Using the same methodology that led to Eq. (18.20), we get

$$\Delta f_{d_{min}} = \frac{2v}{\lambda}\ \Delta\theta\sin\beta_k \qquad \textbf{Eq. (18.23)}$$

where β_k is the elevation angle corresponding to the kth range bin.

The bandwidth of the individual Doppler filters must be equal to the inverse of the coherent integration interval T_{ob} (i.e., $\Delta f_{d_{min}} = 1/T_{ob}$). It follows that

$$\Delta\theta = \frac{\lambda}{2vT_{ob}\sin\beta_k}\,. \qquad \textbf{Eq. (18.24)}$$

Substituting L for vT_{ob} yields

$$\Delta\theta = \frac{\lambda}{2L\sin\beta_k}\,. \qquad \textbf{Eq. (18.25)}$$

Therefore, the SAR azimuth resolution (within the kth range bin) is

$$\Delta A_g = \Delta\theta R_k = R_k\ \frac{\lambda}{2L\sin\beta_k}\,. \qquad \textbf{Eq. (18.26)}$$

Note that when $\beta_k = 90°$, Eq. (18.26) is identical to Eq. (18.10).

18.3. SAR Radar Equation

The single-pulse radar equation was derived in Chapter 2, and is repeated here as Eq. (18.27),

$$SNR = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R_k^4 k T_0 B L_{Loss}} \qquad \textbf{Eq. (18.27)}$$

where P_t is peak power, G is antenna gain, λ is wavelength, σ is radar cross section, R_k is radar slant range to the kth range bin, k is Boltzman's constant, T_0 is receiver noise temperature, B is receiver bandwidth, and L_{Loss} is radar losses. The radar cross section is a function of the radar resolution cell and terrain reflectivity. More precisely,

$$\sigma = \sigma^0 \Delta R_g \Delta A_g = \sigma^0 \Delta A_g \frac{c\tau}{2} \sec\psi_g \qquad \text{Eq. (18.28)}$$

where σ^0 is the clutter scattering coefficient, ΔA_g is the azimuth resolution, and Eq. (18.6) was used to replace the ground range resolution. The number of coherently integrated pulses within an observation interval is

$$n = f_r T_{ob} = \frac{f_r L}{v} \qquad \text{Eq. (18.29)}$$

where L is the synthetic aperture size. Using Eq. (18.26) in Eq. (18.29) and rearranging terms yield

$$n = \frac{\lambda R f_r}{2\Delta A_g v} \csc\beta_k . \qquad \text{Eq. (18.30)}$$

The radar average power over the observation interval is

$$P_{av} = (P_t / B) f_r . \qquad \text{Eq. (18.31)}$$

The SNR for n coherently integrated pulses is then

$$(SNR)_n = nSNR = n \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R_k^4 k T_0 B L_{Loss}} . \qquad \text{Eq. (18.32)}$$

Substituting Eqs. (18.31), (18.30), and (18.28) into Eq. (18.32) and performing some algebraic manipulations give the SAR radar equation,

$$(SNR)_n = \frac{P_{av} G^2 \lambda^3 \sigma^0}{(4\pi)^3 R_k^3 k T_0 L_{Loss}} \frac{\Delta R_g}{2v} \csc\beta_k . \qquad \text{Eq. (18.33)}$$

Eq. (18.33) leads to the conclusion that in SAR systems, the SNR is (1) inversely proportional to the third power of range; (2) independent of azimuth resolution; (3) a function of the ground range resolution; (4) inversely proportional to the velocity v; and (5) proportional to the third power of wavelength.

18.4. SAR Signal Processing

There are two signal processing techniques to sequentially produce a SAR map or image; they are line-by-line processing and Doppler processing. The concept of SAR line-by-line processing is as follows: Through the radar linear motion, a synthetic array is formed, where the elements of the current synthetic array correspond to the position of the antenna transmissions during the last observation interval. Azimuth resolution is obtained by forming narrow synthetic beams through combinations of the last observation interval returns. Fine range resolution is accomplished in real time by utilizing range gating and pulse compression. For each

range bin and each of the transmitted pulses during the last observation interval, the returns are recorded in a two-dimensional array of data that is updated for every pulse. Denote the two-dimensional array of data as MAP.

To further illustrate the concept of line-by-line processing, consider the case where a map of size $N_a \times N_r$ is to be produced, where N_a is the number of azimuth cells and N_r is the number of range bins. Hence, MAP is of size $N_a \times N_r$, where the columns refer to range bins, and the rows refer to azimuth cells. For each transmitted pulse, the echoes from consecutive range bins are recorded sequentially in the first row of MAP. Once the first row is completely filled (i.e., returns from all range bins have been received), all data (in all rows) are shifted downward one row before the next pulse is transmitted. Thus, one row of MAP is generated for every transmitted pulse. Consequently, for the current observation interval, returns from the first transmitted pulse will be located in the bottom row of MAP, and returns from the last transmitted pulse will be in the first row of MAP.

In SAR Doppler processing, the array MAP is updated once every N pulses so that a block of N columns is generated simultaneously. In this case, N refers to the number of transmissions during an observation interval (i.e., size of the synthetic array). From an antenna point of view, this is equivalent to having N adjacent synthetic beams formed in parallel through electronic steering.

18.5. Side Looking SAR Doppler Processing

Consider the geometry shown in Fig. 18.9, and assume that the scatterer C_i is located within the kth range bin. The scatterer azimuth and elevation angles are μ_i and β_i, respectively. The scatterer elevation angle β_i is assumed to be equal to β_k, the range bin elevation angle. This assumption is true if the ground range resolution, ΔR_g, is small; otherwise, $\beta_i = \beta_k + \varepsilon_i$ for some small ε_i; in this chapter $\varepsilon_i = 0$.

The normalized transmitted signal can be represented by

$$s(t) = \cos(2\pi f_0 t - \xi_0) \qquad \textbf{Eq. (18.34)}$$

where f_0 is the radar operating frequency, and ξ_0 denotes the transmitter phase. The returned radar signal from C_i is then equal to

$$s_i(t, \mu_i) = A_i \cos[2\pi f_0 (t - \tau_i(t, \mu_i)) - \xi_0] \qquad \textbf{Eq. (18.35)}$$

where $\tau_i(t, \mu_i)$ is the round-trip delay to the scatterer, and A_i includes scatterer strength, range attenuation, and antenna gain. The round-trip delay is

$$\tau_i(t, \mu_i) = \frac{2r_i(t, \mu_i)}{c} \qquad \textbf{Eq. (18.36)}$$

where c is the speed of light and $r_i(t, \mu_i)$ is the scatterer slant range. From the geometry in Fig. 18.9, one can write the expression for the slant range to the ith scatterer within the kth range bin as

$$r_i(t, \mu_i) = \frac{h}{\cos\beta_i} \sqrt{1 - \frac{2vt}{h} \cos\beta_i \cos\mu_i \sin\beta_i + \left(\frac{vt}{h} \cos\beta_i\right)^2}. \qquad \textbf{Eq. (18.37)}$$

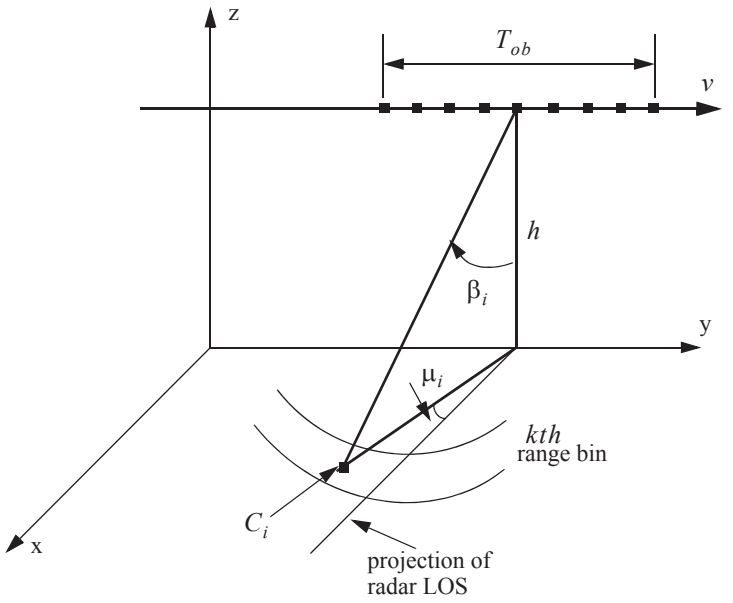


Figure 18.9. A scatterer C_i within the k^{th} range bin.

And by using Eq. (18.36), the round-trip delay can be written as

$$\tau_i(t, \mu_i) = \frac{2}{c}\frac{h}{\cos\beta_i}\sqrt{1 - \frac{2vt}{h}\cos\beta_i\cos\mu_i\sin\beta_i + \left(\frac{vt}{h}\cos\beta_i\right)^2}. \qquad \textbf{Eq. (18.38)}$$

The round-trip delay can be approximated using a two-dimensional second-order Taylor series expansion about the reference state $(t, \mu) = (0, 0)$. Performing this Taylor series expansion yields

$$\tau_i(t, \mu_i) \approx \bar{\tau} + \bar{\tau}_{t\mu}\ \ \mu_i t + \bar{\tau}_{tt}\ \frac{t^2}{2} \qquad \textbf{Eq. (18.39)}$$

where the over-bar indicates evaluation at the state $(0, 0)$, and the subscripts denote partial derivatives. For example, $\bar{\tau}_{t\mu}$ means

$$\bar{\tau}_{t\mu} = \frac{\partial^2}{\partial t \partial \mu}\tau_i(t, \mu_i)\Big|_{(t, \mu) = (0, 0)}. \qquad \textbf{Eq. (18.40)}$$

The Taylor series coefficients are

$$\bar{\tau} = \left(\frac{2h}{c}\right)\frac{1}{\cos\beta_i} \qquad \textbf{Eq. (18.41)}$$

$$\bar{\tau}_{t\mu} = \left(\frac{2v}{c}\right)\sin\beta_i \qquad \textbf{Eq. (18.42)}$$

$$\bar{\tau}_{tt} = \left(\frac{2v^2}{hc}\right)\cos\beta_i \,. \qquad \textbf{Eq. (18.43)}$$

Note that other Taylor series coefficients are either zeros or very small. Hence, they are neglected. Finally, we can rewrite the returned radar signal as

$$\begin{aligned} s_i(t, \mu_i) &= A_i \cos[\psi_i(t, \mu_i) - \xi_0] \\ \hat{\psi}_i(t, \mu_i) &= 2\pi f_0\left[(1 - \bar{\tau}_{t\mu}\mu_i)t - \bar{\tau} - \bar{\tau}_{tt}\frac{t^2}{2}\right] \end{aligned} \,. \qquad \textbf{Eq. (18.44)}$$

Observation of Eq. (18.44) indicates that the instantaneous frequency for the ith scatterer varies as a linear function of time due to the second-order phase term $2\pi f_0(\bar{\tau}_{tt}t^2/2)$ (this confirms the result we concluded about a scatterer Doppler history). Furthermore, since this phase term is range-bin dependent and not scatterer dependent, all scatterers within the same range bin produce this exact second-order phase term. It follows that scatterers within a range bin have identical Doppler histories. These Doppler histories are separated by the time delay required to fly between them, as illustrated in Fig. 18.10.

Suppose that there are I scatterers within the kth range bin. In this case, the combined returns for this cell are the sum of the individual returns due to each scatterer as defined by Eq. (18.44). In other words, superposition holds, and the overall echo signal is

$$s_r(t) = \sum_{i=1}^{I} s_i(t, \mu_i) \,. \qquad \textbf{Eq. (18.45)}$$

A signal processing block diagram for the kth range bin is illustrated in Fig. 18.11. It consists of the following steps. First, heterodyning with the carrier frequency is performed to extract the quadrature components.

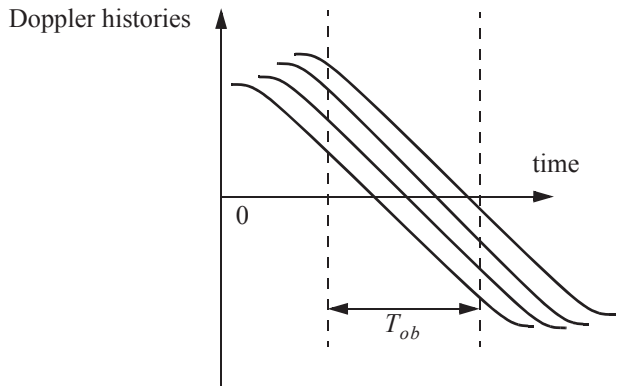


Figure 18.10. Doppler histories for several scatterers within the same range bin.

This is followed by LP filtering and A/D conversion. Next, deramping or focusing to remove the second-order phase term of the quadrature components is carried out using a phase rotation matrix. The last stage of the processing includes windowing, performing an FFT on the windowed quadrature components, and scaling the amplitude spectrum to account for range attenuation and antenna gain.

The discrete quadrature components are

$$\begin{aligned}\tilde{x}_I(t_n) &= \tilde{x}_I(n) = A_i \cos[\tilde{\psi}_i(t_n, \mu_i) - \xi_0] \\ \tilde{x}_Q(t_n) &= \tilde{x}_Q(n) = A_i \sin[\tilde{\psi}_i(t_n, \mu_i) - \xi_0]\end{aligned} \qquad \textbf{Eq. (18.46)}$$

$$\tilde{\psi}_i(t_n, \mu_i) = \hat{\psi}_i(t_n, \mu_i) - 2\pi f_0 t_n \qquad \textbf{Eq. (18.47)}$$

and t_n denotes the nth sampling time (remember that $-T_{ob}/2 \le t_n \le T_{ob}/2$). The quadrature components after deramping (i.e., removal of the phase $\psi = -\pi f_0 \bar{\tau}_{tt} t_n^2$) are given by

$$\begin{bmatrix} x_I(n) \\ x_Q(n) \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \tilde{x}_I(n) \\ \tilde{x}_Q(n) \end{bmatrix}. \qquad \textbf{Eq. (18.48)}$$

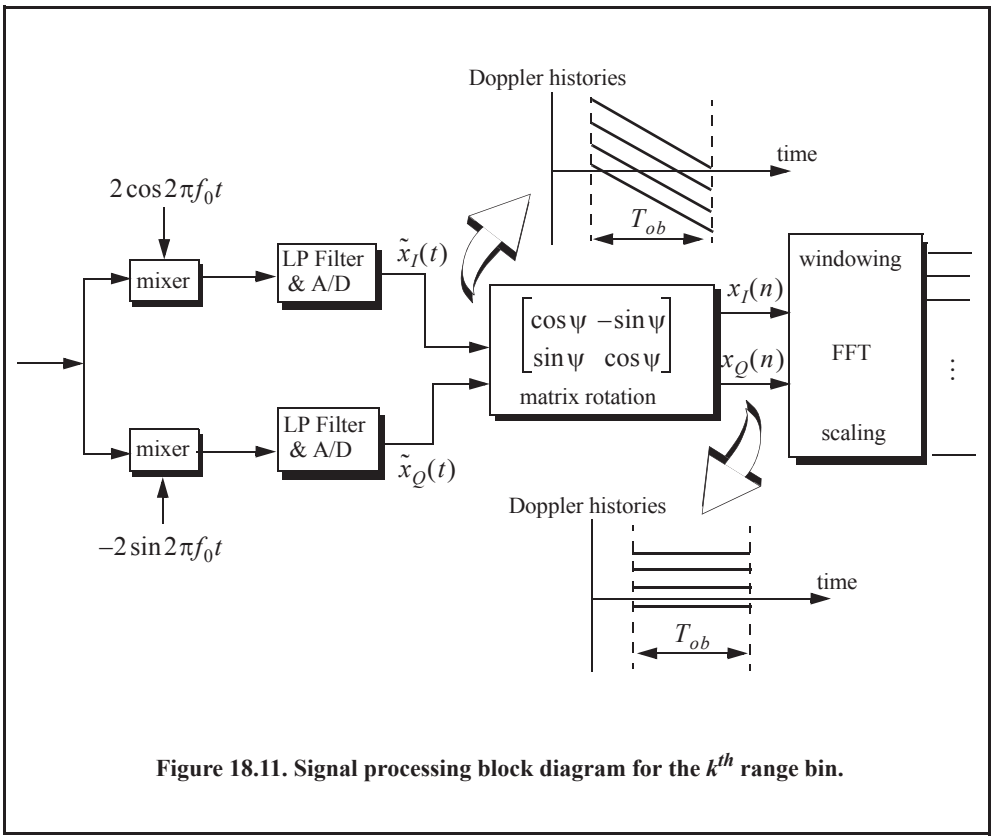


Figure 18.11. Signal processing block diagram for the k^{th} range bin.

18.6. SAR Imaging Using Doppler Processing

It was mentioned earlier that SAR imaging is performed using two orthogonal dimensions (range and azimuth). Range resolution is controlled by the receiver bandwidth and pulse compression. Azimuth resolution is limited by the antenna beamwidth. A one-to-one correspondence between the FFT bins and the azimuth resolution cells can be established by utilizing the signal model described in the previous section. Therefore, the problem of target detection is transformed into a spectral analysis problem, where detection is based on the amplitude spectrum of the returned signal. The FFT frequency resolution Δf is equal to the inverse of the observation interval T_{ob}. It follows that a peak in the amplitude spectrum at $k_1 \Delta f$ indicates the presence of a scatterer at frequency $f_{d1} = k_1 \Delta f$.

For an example, consider the scatterer C_i within the kth range bin. The instantaneous frequency f_{di} corresponding to this scatterer is

$$f_{di} = \frac{1}{2\pi} \frac{d\psi}{dt} = f_0 \dot{\bar{\tau}}_{t\mu} \mu_i = \frac{2v}{\lambda} \sin\beta_i \mu_i . \qquad \textbf{Eq. (18.49)}$$

This is the same result derived in Eq. (18.23), with $\mu_i = \Delta\theta$. Therefore, the scatterers separated in Doppler by more than Δf can then be resolved.

Fig. 18.12 shows a two-dimensional SAR image for three point scatterers located 10Km downrange. In this case, the azimuth and range resolutions are equal to 1m and the operating frequency is 35GHz. Fig. 18.13 is similar to Fig. 18.12, except in this case the resolution cell is equal to 6 inches. One can clearly see the blurring that occurs in the image. Figures 12.12 and 12.13 can be reproduced using the program *"Fig18_12_13.m,"* listed in Appendix 18-A.

18.7. Range Walk

As shown earlier, SAR Doppler processing is achieved in two steps: first, range gating and second, azimuth compression within each bin at the end of the observation interval. For this purpose, azimuth compression assumes that each scatterer remains within the same range bin during the observation interval. However, since the range gates are defined with respect to a radar that is moving, the range gate grid is also moving relative to the ground. As a result, a scatterer appears to be moving within its range bin. This phenomenon is known as range walk. A small amount of range walk does not bother Doppler processing as long as the scatterer remains within the same range bin. However, range walk over several range bins can constitute serious problems, where in this case Doppler processing is meaningless.

18.8. A Three-Dimensional SAR Imaging Technique

This section presents a new three-dimensional (3-D) Synthetic Aperture Radar (SAR) imaging based on Mahafza[1] et al. It utilizes a linear array in transverse motion to synthesize a two-dimensional (2-D) synthetic array. Elements of the linear array are fired sequentially (one element at a time), while all elements receive in parallel. A 2-D information sequence is computed from the equiphase two-way signal returns.

1. Mahafza, B. R. and Sajjadi, M., Three-Dimensional SAR Imaging Using a Linear Array in Transverse Motion, *IEEE - AES Trans.*, Vol. 32, No. 1, January 1996, pp. 499-510.

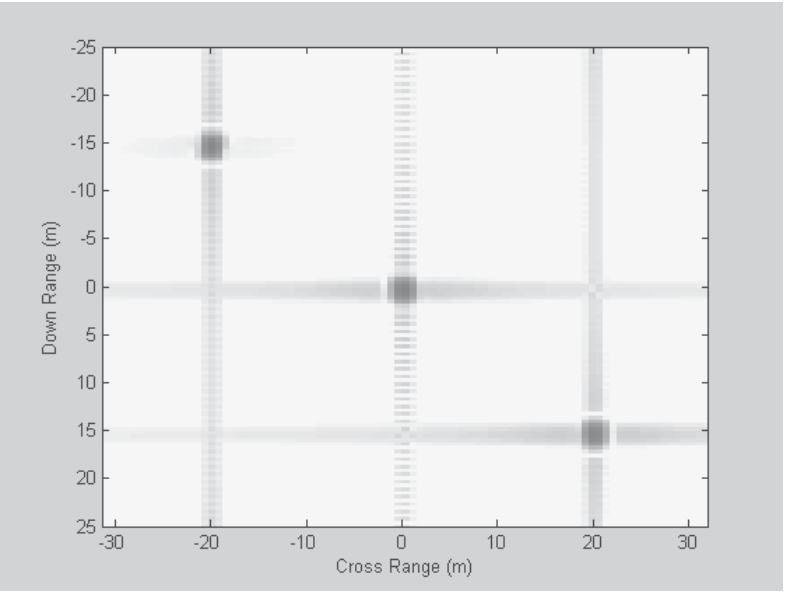


Figure 18.12. Three-point scatterer image. Resolution cell is $1m^2$.

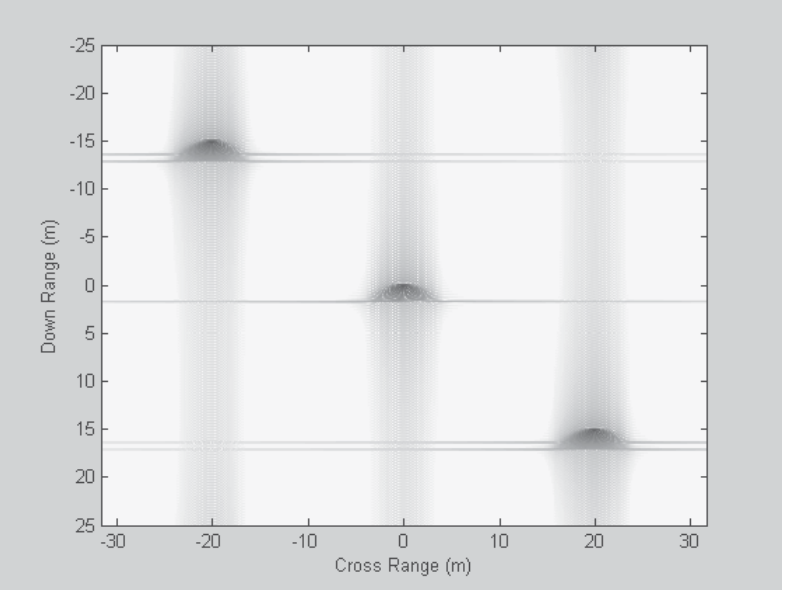


Figure 18.13. Three-point scatterer image. Resolution cell is squared inches.

A signal model based on a third-order Taylor series expansion about incremental relative time, azimuth, elevation, and target height is used. Scatterers are detected as peaks in the amplitude spectrum of the information sequence. Detection is performed in two stages. First, all scatterers within a footprint are detected using an incomplete signal model where target height is set to zero. Then, processing using the complete signal model is performed only on range bins containing significant scatterer returns. The difference between the two images is used to measure target height. Computer simulation shows that this technique is accurate and virtually impulse invariant.

18.8.1. Background

Standard Synthetic Aperture Radar (SAR) imaging systems are generally used to generate high resolution two-dimensional (2-D) images of ground terrain. Range gating determines resolution along the first dimension. Pulse compression techniques are usually used to achieve fine range resolution. Such techniques require the use of wideband receiver and display devices in order to resolve the time structure in the returned signals. The width of azimuth cells provides resolution along the other dimension. Azimuth resolution is limited by the duration of the observation interval.

This section presents a three-dimensional (3-D) SAR imaging technique based on Discrete Fourier Transform (DFT) processing of equiphase data collected in sequential mode (DFTSQM). It uses a linear array in transverse motion to synthesize a 2-D synthetic array. A 2-D information sequence is computed from the equiphase two-way signal returns. To this end, a new signal model based on a third-order Taylor series expansion about incremental relative time, azimuth, elevation, and target height is introduced. Standard SAR imaging can be achieved using an incomplete signal model where target height is set to zero. Detection is performed in two stages. First, all scatterers within a footprint are detected using an incomplete signal model, where target height is set to zero. Then, processing using the complete signal model is performed only on range bins containing significant scatterer returns. The difference between the two images is used as an indication of target height. Computer simulation shows that this technique is accurate and virtually impulse invariant.

18.8.2. DFTSQM Operation and Signal Processing

Linear Arrays

Consider a linear array of size N, uniform element spacing d, and wavelength λ. Assume a far field scatterer P located at direction-sine $\sin\theta_l$. DFTSQM operation for this array can be described as follows. The elements are fired sequentially, one at a time, while all elements receive in parallel. The echoes are collected and integrated coherently on the basis of equal phase to compute a complex information sequence $\{b(m); m = 0, 2N-1\}$. The x-coordinates, in d-units, of the x_n^{th} element with respect to the center of the array is

$$x_n = \left(-\frac{N-1}{2} + n\right); n = 0, \ldots N-1 . \qquad \textbf{Eq. (18.50)}$$

The electric field received by the x_2^{th} element due to the firing of the x_1^{th}, and reflection by the l^{th} far field scatterer P is

$$E(x_1, x_2; s_l) = G^2(s_l)\left(\frac{R_0}{R}\right)^4 \sqrt{\sigma_l}\ exp(j\phi(x_1, x_2; s_l)) \quad \textbf{Eq. (18.51)}$$

$$\phi(x_1, x_2; s_l) = \frac{2\pi}{\lambda}(x_1 + x_2)(s_l) \quad \textbf{Eq. (18.52)}$$

$$s_l = \sin\theta_l \quad \textbf{Eq. (18.53)}$$

where $\sqrt{\sigma_l}$ is the target cross section, $G^2(s_l)$ is the two-way element gain, and $(R_0/R)^4$ is the range attenuation with respect to reference range R_0. The scatterer phase is assumed to be zero, however it could be easily included. Assuming multiple scatterers in the array's FOV, the cumulative electric field in the path $x_1 \Rightarrow x_2$ due to reflections from all scatterers is

$$E(x_1, x_2) = \sum_{all\ l} [E_I(x_1, x_2; s_l) + jE_Q(x_1, x_2; s_l)] \quad \textbf{Eq. (18.54)}$$

where the subscripts (I, Q) denote the quadrature components. Note that the variable part of the phase given in Eq. (18.52) is proportional to the integers resulting from the sums $\{(x_{n1} + x_{n2});\ (n1, n2) = 0, \ldots N-1\}$. In the far field operation, there are a total of $(2N-1)$ distinct $(x_{n1} + x_{n2})$ sums. Therefore, the electric fields with paths of the same $(x_{n1} + x_{n2})$ sums can be collected coherently. In this manner, the information sequence $\{b(m); m = 0, 2N-1\}$ is computed, where $b(2N-1)$ is set to zero. At the same time, one forms the sequence $\{c(m); m = 0, \ldots 2N-2\}$, which keeps track of the number of returns that have the same $(x_{n1} + x_{n2})$ sum. More precisely, for $m = n1 + n2;\ (n1, n2) = \ldots 0, N-1$

$$b(m) = b(m) + E(x_{n1}, x_{n2}) \quad \textbf{Eq. (18.55)}$$

$$c(m) = c(m) + 1\,. \quad \textbf{Eq. (18.56)}$$

It follows that

$$\{c(m); m = 0, \ldots 2N-2\} = \begin{Bmatrix} m+1 & ;\ m = 0, \ldots N-2 \\ N & ;\ m = N-1 \\ 2N-1-m & \quad m = N, \ldots 2N-2 \end{Bmatrix} \quad \textbf{Eq. (18.57)}$$

which is a triangular shape sequence.

The processing of the sequence $\{b(m)\}$ is performed as follows: (1) the weighting takes the sequence $\{c(m)\}$ into account; (2) the complex sequence $\{b(m)\}$ is extended to size N_F, a power integer of two, by zero padding; (3) the DFT of the extended sequence $\{b'(m); m = 0, N_F - 1\}$ is computed,

$$B(q) = \sum_{m=0}^{N_F-1} b'(m) \cdot exp\left(-j\frac{2\pi qm}{N_F}\right); q = 0, \ldots N_F - 1\,; \quad \textbf{Eq. (18.58)}$$

and, (4) after compensation for antenna gain and range attenuation, scatterers are detected as peaks in the amplitude spectrum $|B(q)|$. Note that step (4) is true only when

$$\sin\theta_q = \frac{\lambda q}{2Nd}; q = 0, \ldots 2N-1, \quad \textbf{Eq. (18.59)}$$

where $\sin\theta_q$ denotes the direction-sine of the q^{th} scatterer, and $N_F = 2N$ is implied in Eq. (18.59).

The classical approach to multiple target detection is to use a phased array antenna with phase shifting and tapering hardware. The array beamwidth is proportional to (λ/Nd), and the first sidelobe is at about -13*dB*. On the other hand, multiple target detection using DFTSQM provides a beamwidth proportional to $(\lambda/2Nd)$ as indicated by (Eq. (18.59), which has the effect of doubling the array's resolution. The first sidelobe is at about -27*dB* due to the triangular sequence $\{c(m)\}$. Additionally, no phase shifting hardware is required for detection of targets within a single-element field of view.

Rectangular Arrays

DFTSQM operation and signal processing for 2-D arrays can be described as follows. Consider an $N_x \times N_y$ rectangular array. All N_xN_y elements are fired sequentially, one at a time. After each firing, all the N_xN_y array elements receive in parallel. Thus, N_xN_y samples of the quadrature components are collected after each firing, and a total of $(N_xN_y)^2$ samples will be collected. However, in the far field operation, there are only $(2N_x-1)\times(2N_y-1)$ distinct equiphase returns. Therefore, the collected data can be added coherently to form a 2-D information array of size $(2N_x-1)\times(2N_y-1)$. The two-way radiation pattern is computed as the modulus of the 2-D amplitude spectrum of the information array. The processing includes 2-D windowing, 2-D Discrete Fourier Transformation, antenna gain and range attenuation compensation. The field of view of the 2-D array is determined by the 3*dB* pattern of a single element. All the scatterers within this field will be detected simultaneously as peaks in the amplitude spectrum.

Consider a rectangular array of size $N \times N$, with uniform element spacing $d_x = d_y = d$, and wavelength λ. The coordinates of the n^{th} element, in d-units, are

$$x_n = \left(-\frac{N-1}{2} + n\right) \quad ; n = 0, \ldots N-1 \quad \textbf{Eq. (18.60)}$$

$$y_n = \left(-\frac{N-1}{2} + n\right) \quad ; n = 0, \ldots N-1. \quad \textbf{Eq. (18.61)}$$

Assume a far field point P defined by the azimuth and elevation angles (α, β). In this case, the one-way geometric phase for an element is

$$\varphi'(x,y) = \frac{2\pi}{\lambda}[x\sin\beta\cos\alpha + y\sin\beta\sin\alpha]. \quad \textbf{Eq. (18.62)}$$

Therefore, the two-way geometric phase between the (x_1, y_1) and (x_2, y_2) elements is

$$\varphi(x_1, y_1, x_2, y_2) = \frac{2\pi}{\lambda}\sin\beta[(x_1+x_2)\cos\alpha + (y_1+y_2)\sin\alpha]. \quad \textbf{Eq. (18.63)}$$

The two-way electric field for the l^{th} scatterer at (α_l, β_l) is

$$E(x_1, x_2, y_1, y_2; \alpha_l, \beta_l) = G^2(\beta_l)\left(\frac{R_0}{R}\right)^4 \sqrt{\sigma_l}\ \ exp[j(\varphi(x_1, y_1, x_2, y_2))]\,. \qquad \textbf{Eq. (18.64)}$$

Assuming multiple scatterers within the array's FOV, then the cumulative electric field for the two-way path $(x_1, y_1) \Rightarrow (x_2, y_2)$ is given by

$$E(x_1, x_2, y_1, y_2) = \sum_{all\ \ scatterers} E(x_1, x_2, y_1, y_2; \alpha_l, \beta_l)\,. \qquad \textbf{Eq. (18.65)}$$

All formulas for the 2-D case reduce to those of a linear array case by setting $N_y = 1$ and $\alpha = 0$.

The variable part of the phase given in Eq. (18.63) is proportional to the integers $(x_1 + x_2)$ and $(y_1 + y_2)$. Therefore, after completion of the sequential firing, electric fields with paths of the same (i, j) sums, where

$$\{i = x_{n1} + x_{n2}; i = -(N-1), \ldots (N-1)\} \qquad \textbf{Eq. (18.66)}$$

$$\{j = y_{n1} + y_{n2}; j = -(N-1), \ldots (N-1)\} \qquad \textbf{Eq. (18.67)}$$

can be collected coherently. In this manner, the 2-D information array $\{b(m_x, m_y); (m_x, m_y) = 0, \ldots 2N-1\}$ is computed. The coefficient sequence $\{c(m_x, m_y); (m_x, m_y) = 0, \ldots 2N-2\}$ is also computed. More precisely,

$$\begin{array}{c} for\ \ m_x = n1 + n2\ \ and\ \ m_y = n1 + n2; \\ n1 = 0, \ldots N-1 \quad , and\ \ n2 = 0, \ldots N-1 \end{array} \qquad \textbf{Eq. (18.68)}$$

$$b(m_x, m_y) = b(m_x, m_y) + E(x_{n1}, y_{n1}, x_{n2}, y_{n2})\,. \qquad \textbf{Eq. (18.69)}$$

It follows that

$$c(m_x, m_y) = (N_x - |m_x - (N_x - 1)|) \times (N_y - |m_y - (N_y - 1)|)\,. \qquad \textbf{Eq. (18.70)}$$

The processing of the complex 2-D information array $\{b(m_x, m_y)\}$ is similar to that of the linear case with the exception that one should use a 2-D DFT. After antenna gain and range attenuation compensation, scatterers are detected as peaks in the 2-D amplitude spectrum of the information array. A scatterer located at angles (α_l, β_l) will produce a peak in the amplitude spectrum at DFT indexes (p_l, q_l), where

$$\alpha_l = \text{atan}\left(\frac{q_l}{p_l}\right) \qquad \textbf{Eq. (18.71)}$$

$$\sin\beta_l = \frac{\lambda p_l}{2Nd\cos\alpha_l} = \frac{\lambda q_l}{2Nd\sin\alpha_l}\,. \qquad \textbf{Eq. (18.72)}$$

Derivation of Eq. (18.71) is in Section 12.9.7.

18.8.3. Geometry for DFTSQM SAR Imaging

Fig. 18.14 shows the geometry of the DFTSQM SAR imaging system. In this case, t_c denotes the central time of the observation interval, D_{ob}. The aircraft maintains both constant

velocity v, and height h. The origin for the relative system of coordinates is denoted as $\vec{O}$. The vector $\vec{O}M$ defines the radar location at time t_c. The transmitting antenna consists of a linear real array operating in the sequential mode. The real array is of size N, element spacing d, and the radiators are circular dishes of diameter $D = d$. Assuming that the aircraft scans M transmitting locations along the flight path, then a rectangular array of size $N \times M$ is synthesized, as illustrated in Fig. 18.15.

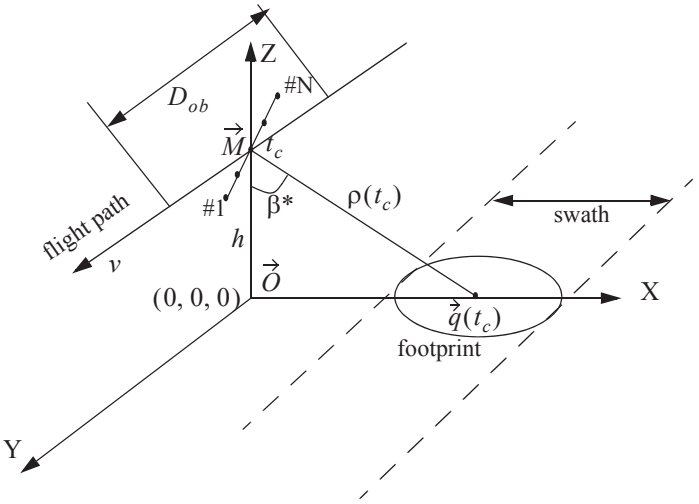


Figure 18.14. Geometry for DFTSQM imaging system.

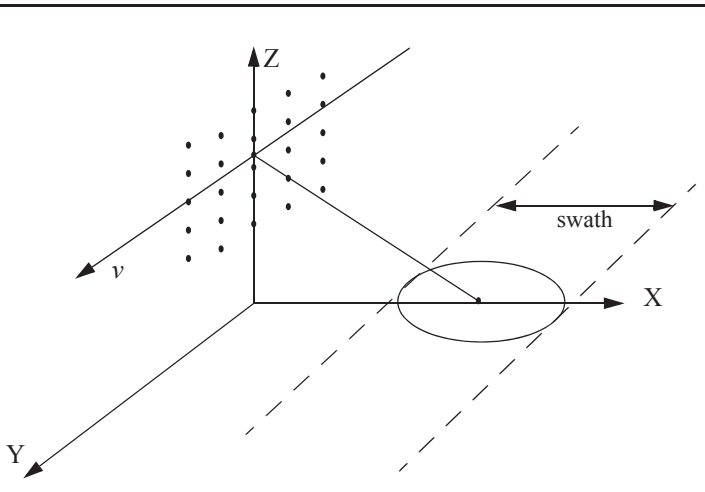


Figure 18.15. Synthesized 2-D array.

The vector $\vec{q}(t_c)$ defines the center of the $3dB$ footprint at time t_c. The center of the array coincides with the flight path, and it is assumed to be perpendicular to both the flight path and the line of sight $\rho(t_c)$. The unit vector $\vec{a}$, along the real array is

$$\vec{a} = \cos\beta^* \vec{a}_x + \sin\beta^* \vec{a}_z \qquad \textbf{Eq. (18.73)}$$

where β^* is the elevation angle, or the compliment of the depression angle, for the center of the footprint at central time t_c.

18.8.4. Slant Range Equation

Consider the geometry shown in Fig. 18.16 and assume that there is a scatterer $\vec{C}_i$ within the k^{th} range cell. This scatterer is defined by

$$\{ampltiude, phase, elevation, azimuth, height\} = \{a_i, \phi_i, \beta_i, \mu_i, \tilde{h}_i\}. \qquad \textbf{Eq. (18.74)}$$

the scatterer $\vec{C}_i$ (assuming rectangular coordinates) is given by

$$\vec{C}_i = h\tan\beta_i \cos\mu_i \vec{a}_x + h\tan\beta_i \sin\mu_i \vec{a}_y + \tilde{h}_i \vec{a}_z \qquad \textbf{Eq. (18.75)}$$

$$\beta_i = \beta_k + \varepsilon \qquad \textbf{Eq. (18.76)}$$

where β_k denotes the elevation angle for the k^{th} range cell at the center of the observation interval and ε is an incremental angle. Let $\vec{O}e_n$ refer to the vector between the n^{th} array element and the point $\vec{O}$, then

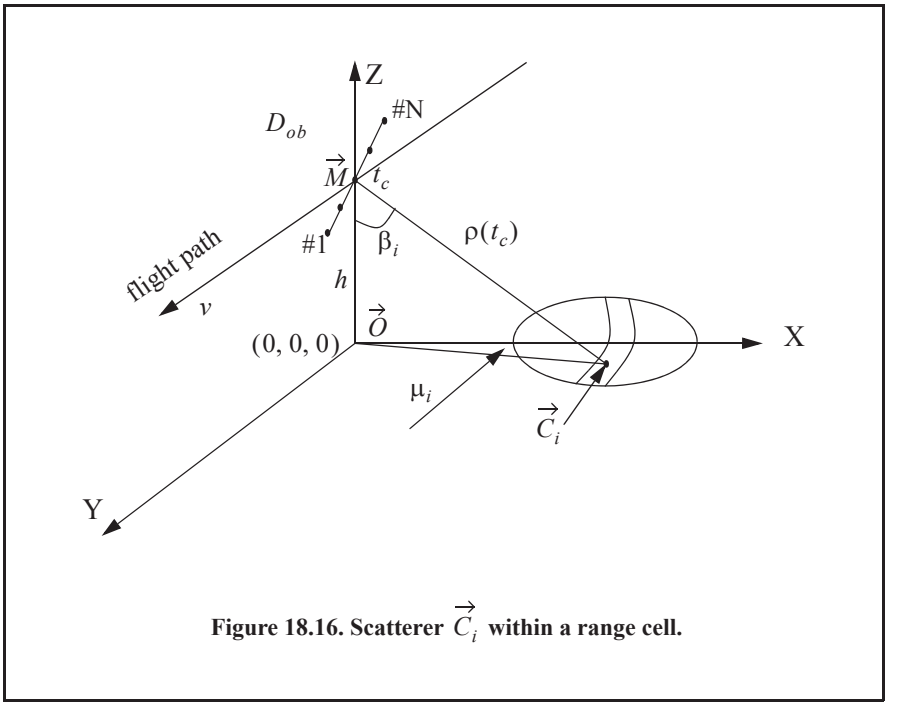


Figure 18.16. Scatterer $\vec{C}_i$ within a range cell.

$$\vec{O}e_n = D_n \cos\beta^* \vec{a}_x + vt\vec{a}_y + (D_n \sin\beta^* + h)\vec{a}_z \qquad \textbf{Eq. (18.77)}$$

$$D_n = \left(\frac{1-N}{2} + n\right)d \quad ; n = 0, \ldots N-1 . \qquad \textbf{Eq. (18.78)}$$

The range between a scatterer $\vec{C}$ within the k^{th} range cell, and the n^{th} element of the real array is

$$r_n^2(t, \varepsilon, \mu, \tilde{h}; D_n) = D_n^2 + v^2t^2 + (h-\tilde{h})^2 + 2D_n \sin\beta^*(h-\tilde{h}) + \quad . \qquad \textbf{Eq. (18.79)}$$
$$h\tan(\beta_k + \varepsilon)[h\tan(\beta_k + \varepsilon) - 2D_n \cos\beta^* \cos\mu - 2vt\sin\mu]$$

It is more practical to use the scatterer's elevation and azimuth direction-sines rather than the corresponding increments. Therefore, define the scatterer's azimuth and elevation direction-sines as

$$s = \sin\mu \qquad \textbf{Eq. (18.80)}$$

$$u = \sin\varepsilon . \qquad \textbf{Eq. (18.81)}$$

Then, one can rewrite Eq. (18.79) as

$$r_n^2(t, s, u, \tilde{h}; D_n) = D_n^2 + v^2t^2 + (h-\tilde{h})^2 + h^2 f^2(u) + \qquad \textbf{Eq. (18.82)}$$
$$2D_n \sin\beta^*(h-\tilde{h}) - (2D_n h\cos\beta^* f(u)\sqrt{1-s^2} - 2vhtf(u)s)$$

$$f(u) = \tan(\beta_k + \operatorname{asin} u) \qquad \textbf{Eq. (18.83)}$$

Expanding r_n as a third-order Taylor series expansion about incremental $(t, s, u, \tilde{h})$ yields

$$r(t, s, u, \tilde{h}; D_n) = \bar{r} + \bar{r}_{\tilde{h}}\tilde{h} + \bar{r}_u u + \bar{r}_{\tilde{h}\tilde{h}}\frac{\tilde{h}^2}{2} + \bar{r}_{\tilde{h}u}\tilde{h}u + \bar{r}_{ss}\frac{s^2}{2} + \bar{r}_{st}st + \qquad \textbf{Eq. (18.84)}$$
$$\bar{r}_{tt}\frac{t^2}{2} + \bar{r}_{uu}\frac{u^2}{2} + \bar{r}_{\tilde{h}\tilde{h}\tilde{h}}\frac{\tilde{h}^3}{6} + \bar{r}_{\tilde{h}\tilde{h}u}\frac{\tilde{h}^2 u}{2} + \bar{r}_{\tilde{h}st}\tilde{h}st + \bar{r}_{\tilde{h}uu}\frac{\tilde{h}u^2}{2} +$$
$$\bar{r}_{\tilde{h}ss}\frac{\tilde{h}s^2}{2} + \bar{r}_{uss}\frac{us^2}{2} + \bar{r}_{stu}stu + \bar{r}_{suu}\frac{su^2}{2} + \bar{r}_{t\tilde{h}\tilde{h}}\frac{t\tilde{h}^2}{2} + \bar{r}_{utt}\frac{ut^2}{2} + \bar{r}_{uuu}\frac{u^3}{6}$$

where subscripts denote partial derivations, and the over-bar indicates evaluation at the state $(t, s, u, \tilde{h}) = (0, 0, 0, 0)$. Note that

$$\{\bar{r}_s = \bar{r}_t = \bar{r}_{\tilde{h}s} = \bar{r}_{\tilde{h}t} = \bar{r}_{su} = \bar{r}_{tu} = \bar{r}_{\tilde{h}\tilde{h}s} = \bar{r}_{\tilde{h}\tilde{h}t} = \bar{r}_{\tilde{h}su} = \bar{r}_{\tilde{h}tu} = \quad . \qquad \textbf{Eq. (18.85)}$$
$$\bar{r}_{sss} = \bar{r}_{sst} = \bar{r}_{stt} = \bar{r}_{ttt} = \bar{r}_{tsu} = 0\}$$

Section 12.9.8 has detailed expressions of all non-zero Taylor series coefficients for the k^{th} range cell.

Even at the maximum increments $t_{mx}, s_{mx}, u_{mx}, \tilde{h}_{mx}$, the terms

$$\bar{r}_{\tilde{h}\tilde{h}\tilde{h}}\frac{\tilde{h}^3}{6}, \bar{r}_{\tilde{h}\tilde{h}u}\frac{\tilde{h}^2 u}{2}, \bar{r}_{\tilde{h}uu}\frac{\tilde{h}u^2}{2}, \bar{r}_{\tilde{h}ss}\frac{\tilde{h}s^2}{2}, \bar{r}_{uss}\frac{us^2}{2}, \bar{r}_{stu}stu, \bar{r}_{suu}\frac{su^2}{2}, \bar{r}_{t\tilde{h}\tilde{h}}\frac{t\tilde{h}^2}{2}, \bar{r}_{utt}\frac{ut^2}{2}, \bar{r}_{uuu}\frac{u^3}{6} \qquad \textbf{Eq. (18.86)}$$

are small and can be neglected. Thus, the range r_n is approximated by

$$r(t, s, u, \tilde{h}; D_n) = \bar{r} + \bar{r}_{\tilde{h}}\tilde{h} + \bar{r}_u u + \bar{r}_{\tilde{h}\tilde{h}}\frac{\tilde{h}^2}{2} + \bar{r}_{\tilde{h}u}\tilde{h}u + \quad . \qquad \text{Eq. (18.87)}$$
$$\bar{r}_{ss}\frac{s^2}{2} + \bar{r}_{st}st + \bar{r}_{tt}\frac{t^2}{2} + \bar{r}_{uu}\frac{u^2}{2} + \bar{r}_{\tilde{h}st}\tilde{h}st$$

Consider the following two-way path: the n_1^{th} element transmitting, scatterer $\vec{C}_i$ reflecting, and the n_2^{th} element receiving. It follows that the round-trip delay corresponding to this two-way path is

$$\tau_{n_1 n_2} = \frac{1}{c}(r_{n_1}(t, s, u, \tilde{h}; D_{n_1}) + r_{n_2}(t, s, u, \tilde{h}; D_{n_2})) \qquad \text{Eq. (18.88)}$$

where c is the speed of light.

18.8.5. Signal Synthesis

The observation interval is divided into M subintervals of width $\Delta t = (D_{ob} \div M)$. During each subinterval, the real array is operated in sequential mode, and an array length of $2N$ is synthesized. The number of subintervals M is computed such that Δt is large enough to allow sequential transmission for the real array without causing range ambiguities. In other words, if the maximum range is denoted as R_{mx}, then

$$\Delta t > N\frac{2R_{mx}}{c}. \qquad \text{Eq. (18.89)}$$

Each subinterval is then partitioned into N sampling subintervals of width $2R_{mx}/c$. The location t_{mn} represents the sampling time at which the n^{th} element is transmitting during the m^{th} subinterval.

The normalized transmitted signal during the m^{th} subinterval for the n^{th} element is defined as

$$s_n(t_{mn}) = \cos(2\pi f_o t_{mn} + \zeta) \qquad \text{Eq. (18.90)}$$

where ζ denotes the transmitter phase, and f_o is the system operating frequency. Assume that there is only one scatterer, $\vec{C}_i$ within the k^{th} range cell defined by $(a_i, \phi_i, s_i, u_i, \tilde{h}_i)$. The returned signal at the n_2^{th} element due to firing from the n_1^{th} element and reflection from the $\vec{C}_i$ scatterer is

$$s_i(n_1, n_2; t_{mn_1}) = a_i G^2(\sin\beta_i)(\rho_k(t_c)/\rho(t_c))^4 \cos[2\pi f_o(t_{mn_1} - \tau_{n_1 n_2}) + \zeta - \phi_i] \qquad \text{Eq. (18.91)}$$

where G^2 represents the two-way antenna gain, and the term $(\rho_k(t_c)/\rho(t_c))^4$ denotes the range attenuation at the k^{th} range cell. The analysis in this paper will assume that ζ and ϕ_i are both equal to zeroes.

Suppose that there are N_o scatterers within the k^{th} range cell, with angular locations given by

$$\{(a_i, \phi_i, s_i, u_i, \tilde{h}_i); i = 1, \ldots N_o\}. \qquad \text{Eq. (18.92)}$$

The composite returned signal at time t_{mn_1} within this range cell due to the path $(n_1 \Rightarrow all\ \vec{C}_i \Rightarrow n_2)$ is

$$s(n_1, n_2; t_{mn_1}) = \sum_{i=1}^{N_o} s_i(n_1, n_2; t_{mn_1}) . \qquad \textbf{Eq. (18.93)}$$

The platform motion synthesizes a rectangular array of size $N \times M$, where only one column of N elements exists at a time. However, if $M = 2N$ and the real array is operated in the sequential mode, a square planar array of size $2N \times 2N$ is synthesized. The element spacing along the flight path is $d_y = vD_{ob}/M$.

Consider the k^{th} range bin. The corresponding two-dimensional information sequence $\{b_k(n, m); (n, m) = 0, \ldots 2N-2\}$ consists of $2N$ similar vectors. The m^{th} vector represents the returns due to the sequential firing of all N elements during the m^{th} subinterval. Each vector has $(2N-1)$ rows, and it is extended, by adding zeroes, to the next power of two. For example, consider the m^{th} subinterval, and let $M = 2N = 4$. Then, the elements of the extended column $\{b_k(n, m)\}$ are

$$\begin{aligned} \{b_k(0, m), b_k(1, m), b_k(2, m), b_k(3, m), b_k(4, m), b_k(5, m), \\ b_k(6, m), b_k(7, m)\} = \{s(0, 0; t_{mn_0}), s(0, 1; t_{mn_0}) + s(1, 0; t_{mn_1}), \\ s(0, 2; t_{mn_0}) + s(1, 1; t_{mn_1}) + s(2, 0; t_{mn_2}), s(0, 3; t_{mn_0}) + s(1, 2; t_{mn_1}) + \\ s(2, 1; t_{mn_2}) + s(3, 0; t_{mn_3}), s(1, 3; t_{mn_1}) + s(2, 2; t_{mn_2}) + \\ s(3, 1; t_{mn_3}), s(2, 3; t_{mn_2}) + s(3, 2; t_{mn_3}), s(3, 3; t_{mn_3}), 0\} \end{aligned} \qquad \textbf{Eq. (18.94)}$$

18.8.6. Electronic Processing

Consider again the k^{th} range cell during the m^{th} subinterval, and the two-way path: n_1^{th} element transmitting and n_2^{th} element receiving. The analog quadrature components corresponding to this two-way path are

$$s_I^{\perp}(n_1, n_2; t) = B\cos\psi^{\perp} \qquad \textbf{Eq. (18.95)}$$

$$s_Q^{\perp}(n_1, n_2; t) = B\sin\psi^{\perp} \qquad \textbf{Eq. (18.96)}$$

$$\begin{aligned} \psi^{\perp} = 2\pi f_0 \Bigg\{ t - \frac{1}{c}\Bigg[& 2\bar{r} + (\bar{r}_{\tilde{h}}(D_{n_1}) + \bar{r}_{\tilde{h}}(D_{n_2}))\tilde{h} + (\bar{r}_u(D_{n_1}) + \bar{r}_u(D_{n_2}))u + \\ & (\bar{r}_{\tilde{h}\tilde{h}}(D_{n_1}) + \bar{r}_{\tilde{h}\tilde{h}}(D_{n_2}))\frac{\tilde{h}^2}{2} + (\bar{r}_{\tilde{h}u}(D_{n_1}) + \bar{r}_{\tilde{h}u}(D_{n_2}))\tilde{h}u + \\ & (\bar{r}_{ss}(D_{n_1}) + \bar{r}_{ss}(D_{n_2}))\frac{s^2}{2} + 2\bar{r}_{st}st + 2\bar{r}_{tt}\frac{t^2}{2} + \\ & (\bar{r}_{uu}(D_{n_1}) + \bar{r}_{uu}(D_{n_2}))\frac{u^2}{2} + (\bar{r}_{\tilde{h}st}(D_{n_1}) + \bar{r}_{\tilde{h}st}(D_{n_2}))\tilde{h}st\Bigg] \Bigg\} \end{aligned} \qquad \textbf{Eq. (18.97)}$$

where B denotes antenna gain, range attenuation, and scatterers' strengths. The subscripts for t have been dropped for notation simplicity. Rearranging Eq. (18.97) and collecting terms yields

$$\psi^{\perp} = \frac{2\pi f_0}{c}\Bigg\{\{tc - [2\bar{r}_{st}s + (\bar{r}_{\tilde{h}st}(D_{n_1}) + \bar{r}_{\tilde{h}st}(D_{n_2}))\tilde{h}s]t - \bar{r}_{tt}t^2\} - \quad \textbf{Eq. (18.98)}$$

$$\Big[2\bar{r} + (\bar{r}_{\tilde{h}}(D_{n_1}) + \bar{r}_{\tilde{h}}(D_{n_2}))\tilde{h} + (\bar{r}_u(D_{n_1}) + \bar{r}_u(D_{n_2}))u +$$

$$(\bar{r}_{uu}(D_{n_1}) + \bar{r}_{uu}(D_{n_2}))\frac{u^2}{2} + (\bar{r}_{\tilde{h}\tilde{h}}(D_{n_1}) + \bar{r}_{\tilde{h}\tilde{h}}(D_{n_2}))\frac{\tilde{h}^2}{2} +$$

$$(\bar{r}_{\tilde{h}u}(D_{n_1}) + \bar{r}_{\tilde{h}u}(D_{n_2}))\tilde{h}u + (\bar{r}_{ss}(D_{n_1}) + \bar{r}_{ss}(D_{n_2}))\frac{s^2}{2}\Big]\Bigg\}$$

After analog-to-digital (A/D) conversion, deramping of the quadrature components to cancel the quadratic phase $(-2\pi f_0 \bar{r}_{tt} t^2/c)$ is performed. Then, the digital quadrature components are

$$s_I(n_1, n_2; t) = B\cos\psi \quad \textbf{Eq. (18.99)}$$

$$s_Q(n_1, n_2; t) = B\sin\psi \quad \textbf{Eq. (18.100)}$$

$$\psi = \psi^{\perp} - 2\pi f_0 t + 2\pi f_0 \bar{r}_{tt}\frac{t^2}{c}. \quad \textbf{Eq. (18.101)}$$

The instantaneous frequency for the i^{th} scatterer within the k^{th} range cell is computed as

$$f_{di} = \frac{1}{2\pi}\frac{\mathrm{d}\psi}{\mathrm{d}t} = -\frac{f_0}{c}[2\bar{r}_{st}s + (\bar{r}_{\tilde{h}st}(D_{n_1}) + \bar{r}_{\tilde{h}st}(D_{n_2}))\tilde{h}s]. \quad \textbf{Eq. (18.102)}$$

Substituting the actual values for $\bar{r}_{st}$, $\bar{r}_{\tilde{h}st}(D_{n_1})$, $\bar{r}_{\tilde{h}st}(D_{n_2})$ and collecting terms yields

$$f_{di} = -\left(\frac{2v\sin\beta_k}{\lambda}\right)\left(\frac{\tilde{h}s}{\rho_k^2(t_c)}(h + (D_{n_1} + D_{n_2})\sin\beta^*) - s\right). \quad \textbf{Eq. (18.103)}$$

Note that if $\tilde{h} = 0$, then

$$f_{di} = \frac{2v}{\lambda}\sin\beta_k \sin\mu, \quad \textbf{Eq. (18.104)}$$

which is the Doppler value corresponding to a ground patch (see Eq. (18.49)).

The last stage of the processing consists of three steps: (1) two-dimensional windowing; (2) performing a two-dimensional DFT on the windowed quadrature components; and (3) scaling to compensate for antenna gain and range attenuation.

18.8.7. Derivation of Eq. (18.71)

Consider a rectangular array of size $N \times N$, with uniform element spacing $d_x = d_y = d$, and wavelength λ. Assume sequential mode operation where elements are fired sequentially, one at a time, while all elements receive in parallel. Assume far field observation defined by azimuth and elevation angles (α, β). The unit vector $\hat{u}$ on the line of sight, with respect to $\vec{O}$, is given by

$$\hat{u} = \sin\beta\cos\alpha\ \hat{a}_x + \sin\beta\sin\alpha\ \hat{a}_y + \cos\beta\ \hat{a}_z . \qquad \textbf{Eq. (18.105)}$$

The $(n_x, n_y)^{th}$ element of the array can be defined by the vector

$$\hat{e}(n_x, n_y) = \left(n_x - \frac{N-1}{2}\right)d\ \hat{a}_x + \left(n_y - \frac{N-1}{2}\right)d\ \hat{a}_y \qquad \textbf{Eq. (18.106)}$$

where $(n_x, n_y = 0, \ldots N-1)$. The one-way geometric phase for this element is

$$\varphi'(n_x, n_y) = k(\hat{u} \bullet \hat{e}(n_x, n_y)) \qquad \textbf{Eq. (18.107)}$$

where $k = 2\pi/\lambda$ is the wavenumber, and the operator ($\bullet$) indicates dot product. Therefore, the two-way geometric phase between the (n_{x1}, n_{y1}) and (n_{x2}, n_{y2}) elements is

$$\varphi(n_{x1}, n_{y1}, n_{x2}, n_{y2}) = k[\hat{u} \bullet \{\hat{e}(n_{x1}, n_{y1}) + \hat{e}(n_{x2}, n_{y2})\}] . \qquad \textbf{Eq. (18.108)}$$

The cumulative two-way normalized electric field due to all transmissions in the direction (α, β) is

$$E(\hat{u}) = E_t(\hat{u})E_r(\hat{u}) \qquad \textbf{Eq. (18.109)}$$

where the subscripts t and r, respectively refer to the transmitted and received electric fields. More precisely,

$$E_t(\hat{u}) = \sum_{n_{xt}=0}^{N-1} \sum_{n_{yt}=0}^{N-1} w(n_{xt}, n_{yt}) exp[jk\{\hat{u} \bullet \hat{e}(n_{xt}, n_{yt})\}] \qquad \textbf{Eq. (18.110)}$$

$$E_r(\hat{u}) = \sum_{n_{xr}=0}^{N-1} \sum_{n_{yr}=0}^{N-1} w(n_{xr}, n_{yr}) exp[jk\{\hat{u} \bullet \hat{e}(n_{xr}, n_{yr})\}] . \qquad \textbf{Eq. (18.111)}$$

In this case, $w(n_x, n_y)$ denotes the tapering sequence. Substituting Eqs. (18.108), (18.110), and (18.111) into Eq. (18.109) and grouping all fields with the same two-way geometric phase yields

$$E(\hat{u}) = e^{j\delta} \sum_{m=0}^{N_a-1} \sum_{n=0}^{N_a-1} w'(m, n) exp[jkd\sin\beta(m\cos\alpha + n\sin\alpha)] \qquad \textbf{Eq. (18.112)}$$

$$N_a = 2N - 1 \qquad \textbf{Eq. (18.113)}$$

$$m = n_{xt} + n_{xr}; m = 0, \ldots 2N-2 \qquad \textbf{Eq. (18.114)}$$

$$n = n_{yt} + n_{yr}; n = 0, \ldots 2N-2 \qquad \textbf{Eq. (18.115)}$$

$$\delta = \left(\frac{-d\sin\beta}{2}\right)(N-1)(\cos\alpha + \sin\alpha) . \qquad \textbf{Eq. (18.116)}$$

The two-way array pattern is then computed as

$$|E(\tilde{u})| = \left| \sum_{m=0}^{N_a-1} \sum_{n=0}^{N_a-1} w'(m,n) exp[jkd \sin\beta(m\cos\alpha + n\sin\alpha)] \right| . \quad \text{Eq. (18.117)}$$

Consider the two-dimensional DFT transform, $W'(p, q)$, of the array $w'(n_x, n_y)$

$$W'(p,q) = \sum_{m=0}^{N_a-1} \sum_{n=0}^{N_a-1} w'(m,n) exp\left(-j\frac{2\pi}{N_a}(pm+qn)\right); p, q = 0, \ldots N_a - 1 . \quad \text{Eq. (18.118)}$$

Comparison of Eqs. (18.117) and Eq. (18.118) indicates that $|E(\tilde{u})|$ is equal to $|W'(p, q)|$ if

$$-\left(\frac{2\pi}{N_a}\right)p = \frac{2\pi}{\lambda} d \sin\beta\cos\alpha \quad \text{Eq. (18.119)}$$

$$-\left(\frac{2\pi}{N_a}\right)q = \frac{2\pi}{\lambda} d \sin\beta\sin\alpha . \quad \text{Eq. (18.120)}$$

It follows that

$$\alpha = \tan^{-1}\left(\frac{q}{p}\right) . \quad \text{Eq. (18.121)}$$

18.8.8. Non-Zero Taylor Series Coefficients for the k^th Range Cell

$$\bar{r} = \sqrt{D_n^2 + h^2(1+\tan\beta_k) + 2hD_n \sin\beta^* - 2hD_n \cos\beta^* \tan\beta_k} = \rho_k(t_c) \quad \text{Eq. (18.122)}$$

$$\bar{r}_{\tilde{h}} = \left(\frac{-1}{\bar{r}}\right)(h + D_n \sin\beta^*) \quad \text{Eq. (18.123)}$$

$$\bar{r}_u = \left(\frac{h}{\bar{r}\cos^2\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) \quad \text{Eq. (18.124)}$$

$$\bar{r}_{\tilde{h}\tilde{h}} = \left(\frac{1}{\bar{r}}\right) - \left(\frac{1}{\bar{r}^3}\right)(h + D_n \sin\beta^*) \quad \text{Eq. (18.125)}$$

$$\bar{r}_{\tilde{h}u} = \left(\frac{1}{\bar{r}^3}\right)\left(\frac{h}{\cos^2\beta_k}\right)(h + D_n\tan\beta^*)(h\tan\beta_k - D_n\cos\beta^*) \quad \text{Eq. (18.126)}$$

$$\bar{r}_{ss} = \left(\frac{-1}{4\bar{r}^3}\right) + \left(\frac{1}{\bar{r}}\right)(h\tan\beta_k - D_n\cos\beta^*) \quad \text{Eq. (18.127)}$$

$$\bar{r}_{st} = \left(\frac{-1}{\bar{r}}\right) hv\tan\beta_k \quad \text{Eq. (18.128)}$$

$$\bar{r}_{tt} = \frac{v^2}{\bar{r}} \qquad \textbf{Eq. (18.129)}$$

$$\bar{r}_{uu} = \left(\frac{h}{\bar{r}\cos^3\beta_k}\right)\left\{\left(\frac{h}{\bar{r}^2\cos\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) + h\left(\left(\frac{1}{\cos\beta_k}\right) + 2\tan\beta_k\sin\beta_k\right) - 2\sin\beta_k D_n\cos\beta^*\right\} \qquad \textbf{Eq. (18.130)}$$

$$\bar{r}_{\tilde{h}\tilde{h}\tilde{h}} = \left(\frac{3}{\bar{r}^3}\right)(h + D_n\sin\beta^*)\left[\left(\frac{1}{\bar{r}^2}\right)(h + D_n\sin\beta^*)^2 - 1\right] \qquad \textbf{Eq. (18.131)}$$

$$\bar{r}_{\tilde{h}\tilde{h}u} = \left(\frac{h}{\bar{r}^3\cos^2\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*)\left[\left(\frac{-3}{\bar{r}^2}\right)(h + D_n\sin\beta^*)^2 + 1\right] \qquad \textbf{Eq. (18.132)}$$

$$\bar{r}_{\tilde{h}st} = \left(\frac{hv\tan\beta_k}{\bar{r}^3}\right)(h + D_n\sin\beta^*) \qquad \textbf{Eq. (18.133)}$$

$$\bar{r}_{\tilde{h}uu} = \left(\frac{-3}{\bar{r}^5}\right)\left(\frac{h^2}{\cos^4\beta_k}\right)(h + D_n\sin\beta^*)(h\tan\beta_k - D_n\cos\beta^*) \qquad \textbf{Eq. (18.134)}$$

$$\bar{r}_{\tilde{h}ss} = \left(\frac{-1}{\bar{r}^3}\right)(h\tan\beta_k - D_n\cos\beta^*)(h + D_n\sin\beta^*) \qquad \textbf{Eq. (18.135)}$$

$$\bar{r}_{uss} = \left(\frac{h}{\bar{r}\cos^2\beta_k}\right)(D_n\cos\beta^*)\left[\left(\frac{1}{\bar{r}^2}\right)(h\tan\beta_k - D_n\cos\beta^*)(h\tan\beta_k) + 1\right] \qquad \textbf{Eq. (18.136)}$$

$$\bar{r}_{stu} = \left(\frac{-h\tan\beta_k}{\bar{r}^3\cos^2\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) \qquad \textbf{Eq. (18.137)}$$

$$\bar{r}_{suu} = \left(\frac{hD_n\cos\beta^*}{\bar{r}\cos^2\beta_k}\right)\left[\left(\frac{h\tan\beta_k}{\bar{r}^2}\right)(h\tan\beta_k - D_n\cos\beta^*) + 1\right] \qquad \textbf{Eq. (18.138)}$$

$$\bar{r}_{\tilde{h}tt} = \left(\frac{v^2h}{\bar{r}^3\cos^2\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) \qquad \textbf{Eq. (18.139)}$$

$$\bar{r}_{uuu} = \qquad \textbf{Eq. (18.140)}$$

$$\left(\frac{h}{\bar{r}\cos^4\beta_k}\right)[8h\tan\beta_k + \sin^2\beta_k(h - D_n\cos\beta^*) - 2D_n\cos\beta^*] + \left(\frac{3h^2}{\bar{r}^3\cos^5\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) + \left[\left(\frac{3h^2}{\bar{r}^3\cos^5\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*)\left(\frac{1}{2\cos\beta_k} + (h\tan\beta_k - D_n\cos\beta^*)\right)\right] + \left(\frac{3h^3}{\bar{r}^5\cos^6\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*)$$

Problems

18.1. A side looking SAR is traveling at an altitude of $15Km$; the elevation angle is $\beta = 15°$. If the aperture length is $L = 5m$, the pulse width is $\tau = 20\mu s$ and the wavelength is $\lambda = 3.5cm$, (a) calculate the azimuth resolution, (b) calculate the range and ground range resolutions.

18.2. An MMW side looking SAR has the following specifications: radar velocity $v = 70m/s$, elevation angle $\beta = 35°$, operating frequency $f_0 = 94GHz$, and antenna $3dB$ beamwidth $\theta_{3dB} = 65mrad$. (a) Calculate the footprint dimensions. (b) Compute the minimum and maximum ranges. (c) Compute the Doppler frequency span across the footprint. (d) Calculate the minimum and maximum PRFs.

18.3. A side looking SAR takes on eight positions within an observation interval. In each position, the radar transmits and receives one pulse. Let the distance between any two consecutive antenna positions be d, and define $\delta = 2\pi\frac{d}{\lambda}(\sin\beta - \sin\beta_0)$ to be the one-way phase difference for a beam steered at angle β_0. (a) In each of the eight positions a sample of the phase pattern is obtained after heterodyning. List the phase samples. (b) How will you process the sequence of samples using an FFT (do not forget windowing)? (c) Give a formula for the angle between the grating lobes.

18.4. Consider a synthetic aperture radar. You are given the following Doppler history for a scatterer: $\{1000Hz, 0, -1000HZ\}$, which corresponds to times $\{-10ms, 0, 10ms\}$. Assume that the observation interval is $T_{ob} = 20ms$, and a platform velocity $v = 200m/s$. (a) Show the Doppler history for another scatterer which is identical to the first one except that it is located in azimuth $1m$ earlier. (b) How will you perform deramping on the quadrature components (show only the general approach)? (c) Show the Doppler history for both scatterers after deramping.

18.5. You want to design a side looking synthetic aperture ultrasonic radar operating at $f_0 = 60KHz$ and peak power $P_t = 2W$. The antenna beam is conical with $3dB$ beamwidth $\theta_{3dB} = 5°$. The maximum gain is 16. The radar is at a constant altitude $h = 15m$ and is moving at a velocity of $10m/s$. The elevation angle defining the footprint is $\beta = 45°$. (a) Give an expression for the antenna gain assuming a Gaussian pattern. (b) Compute the pulse width corresponding to range resolution of $10mm$. (c) What are the footprint dimensions? (d) Compute and plot the Doppler history for a scatterer located on the central range bin. (e) Calculate the minimum and maximum PRFs. Do you need to use more than one PRF? (f) How will you design the system in order to achieve an azimuth resolution of $10mm$?

18.6. Validate Eq. (18.46).

18.7. In Section 18.7 we assumed the elevation angle increment ε is equal to zero. Develop an equivalent to Eq. (18.43) for the case when $\varepsilon \neq 0$. You need to use a third-order three-dimensional Taylor series expansion about the state $(t, \mu, \varepsilon) = (0, 0, 0)$ in order to compute the new round-trip delay expression.

Appendix 18-A: Chapter 18 MATLAB Code Listings

The MATLAB code provided in this chapter was designed as an academic standalone tool and is not adequate for other purposes. The code was written in a way to assist the reader in gaining a better understanding of the theory. The code was not developed, nor is it intended to be used as part of an open-loop or a closed-loop simulation of any kind. The MATLAB code found in this textbook can be downloaded from this book's web page on the CRC Press website. Simply use your favorite web browser, go to www.crcpress.com, and search for keyword *"Mahafza"* to locate this book's web page.

MATLAB Program "Fig18_12-13.m" Listing

```
%                    Figures 18.12 and 18.13
%   Program to do Spotlight SAR using the rectangular format and
%   HRR for range compression.
%                   13 June 2003
%                Dr. Brian J. Smith
clear all;
%%%%%%%%% SAR Image Resolution %%%%
dr = .50;
da = .10;
% dr = 6*2.54/100;
% da = 6*2.54/100;
%%%%%%%%% Scatter Locations %%%%%%%%
xn = [10000 10015 9985];  % Scatter Location, x-axis
yn = [0 -20 20];          % Scatter Location, y-axis
Num_Scatter = 3;          % Number of Scatters
Rnom = 10000;
%%%%%%%%% Radar Parameters %%%%%%%%
f_0 =   35.0e9;    % Lowest Freq. in the HRR Waveform
df =    3.0e6;    % Freq. step size for HRR, Hz
c =      3e8;    % Speed of light, m/s
Kr = 1.33;
Num_Pulse = 2^(round(log2(Kr*c/(2*dr*df))));
Lambda = c/(f_0 + Num_Pulse*df/2);
%%%%%%%%% Synthetic Array Parameters %%%%%%%%
du = 0.2;
L = round(Kr*Lambda*Rnom/(2*da));
U = -(L/2):du:(L/2);
Num_du = length(U);
%%%%%%%%% This section generates the target returns %%%%%%
Num_U = round(L/du);
I_Temp = 0;
Q_Temp = 0;
for I = 1:Num_U
   for J = 1:Num_Pulse
      for K = 1:Num_Scatter
         Yr = yn(K) - ((I-1)*du - (L/2));
         Rt = sqrt(xn(K)^2 + Yr^2);
         F_ci = f_0 + (J -1)*df;
         PHI = -4*pi*Rt*F_ci/c;
         I_Temp = cos(PHI) + I_Temp;
         Q_Temp = sin(PHI) + Q_Temp;
```

```
      end;
      IQ_Raw(J,I) = I_Temp + i*Q_Temp;
      I_Temp = 0.0;
      Q_Temp = 0.0;
   end;
end;
%%%%%%%%%% End target return section %%%%%
%%%%%%%%%% Range Compression %%%%%%%%%%%%%%
Num_RB = 2*Num_Pulse;
WR = hamming(Num_Pulse);
for I = 1:Num_U
   Range_Compressed(:,I) = fftshift(ifft(IQ_Raw(:,I).*WR,Num_RB));
end;
%%%%%%%%%% Focus Range Compressed Data %%%%%
dn = (1:Num_U)*du - L/2;
PHI_Focus = -2*pi*(dn.^2)/(Lambda*xn(1));
for I = 1:Num_RB
   Temp = angle(Range_Compressed(I,:)) - PHI_Focus;
   Focused(I,:) = abs(Range_Compressed(I,:)).*exp(i*Temp);
end;
%Focused = Range_Compressed;
%%%%%%%%%% Azimuth Compression %%%%%%%%%%%%%
WA = hamming(Num_U);
for I = 1:Num_RB
  AZ_Compressed(I,:) = fftshift(ifft(Focused(I,:).*WA'));
end;
 SAR_Map = 10*log10(abs(AZ_Compressed));
Y_Temp = (1:Num_RB)*(c/(2*Num_RB*df));
Y = Y_Temp - max(Y_Temp)/2;
X_Temp = (1:length(IQ_Raw))*(Lambda*xn(1)/(2*L));
X = X_Temp - max(X_Temp)/2;
image(X,Y,20-SAR_Map);  %
%image(X,Y,5-SAR_Map);  %
axis([-25 25 -25 25]); axis equal; colormap(gray(64));
xlabel('Cross Range (m)'); ylabel('Down Range (m)');
grid
%print -djpeg .jpg
```