Chapter 8 *Phased Arrays*

8.1. 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 *antenna radiation pattern*. The directivity of a transmitting antenna can be defined by

$$G_D = \frac{maximum\ \ radiation\ \ intensity}{average\ \ radiation\ \ intensity} \tag{8.1}$$

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. (8.1) can be written as

$$G_D = \frac{4\pi(maximum\ \ radiated\ \ power/unit\ \ solid\ \ angle)}{total\ \ radiated\ \ power} \tag{8.2}$$

It follows that

$$G_D = 4\pi\ \frac{P(\theta, \phi)_{max}}{\int_0^{2\pi}\int_0^{\pi} P(\theta, \phi)\, d\theta\, d\phi} \tag{8.3}$$

As an approximation, it is customary to rewrite Eq. (8.3) as

$$G_D \approx \frac{4\pi}{\theta_3 \phi_3} \tag{8.4}$$

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

$$G = \rho_r G_D \tag{8.5}$$

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. (8.1). The antenna effective aperture A_e is related to gain by

$$A_e = \frac{G\lambda^2}{4\pi} \tag{8.6}$$

where λ is the wavelength. The relationship between the antenna's effective aperture A_e and the physical aperture A is

$$\begin{aligned} A_e &= \rho A \\ 0 &\le \rho \le 1 \end{aligned} \tag{8.7}$$

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. (8.4) through (8.6) (assuming that $\rho_r = 1$) yields

$$G = \frac{4\pi A_e}{\lambda^2} \approx \frac{4\pi}{\theta_3 \phi_3} \tag{8.8}$$

Consequently, the angular cross section of the beam is

$$\theta_3 \phi_3 \approx \frac{\lambda^2}{A_e} \tag{8.9}$$

Eq. (8.9) indicates that the antenna beamwidth decreases as $\sqrt{A_e}$ increases. It follows that, in surveillance operations, the number of beam positions an antenna will take on to cover a volume V is

$$N_{Beams} > \frac{V}{\theta_3 \phi_3} \tag{8.10}$$

and when V represents the entire hemisphere, Eq. (8.10) is modified to

$$N_{Beams} > \frac{2\pi}{\theta_3 \phi_3} \approx \frac{2\pi A_e}{\lambda^2} \approx \frac{G}{2} \tag{8.11}$$

8.2. 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 (equi-phase fronts). In the Fraunhofer regions 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. 8.1. 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

$$\delta r = \overline{AO} - \overline{OB} = \sqrt{r^2 + \left(\frac{d}{2}\right)^2} - r \tag{8.12}$$

and since in the far field $r \gg d$, Eq. (8.12) is approximated via binomial expansion by

$$\delta r = r\left(\sqrt{1 + \left(\frac{d}{2r}\right)^2} - 1\right) \approx \frac{d^2}{8r} \tag{8.13}$$

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

$$\delta r = d^2/8r \le \lambda/16 \tag{8.14}$$

then a useful expression for far field is

$$r \geq 2d^2/\lambda \tag{8.15}$$

Note that far field is a function of both the antenna size and the operating wavelength.

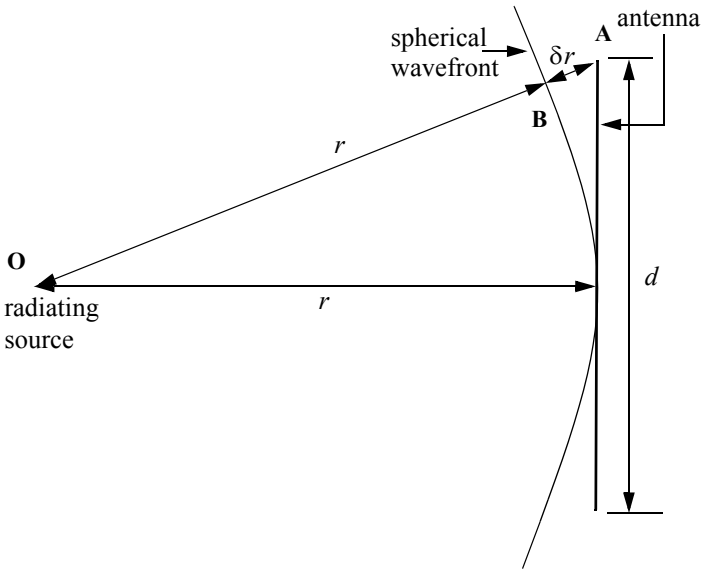

Figure 8.1. Construction of far field criterion.

8.3. 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 multi-function radar systems have made phased array antennas attractive for radar applications.

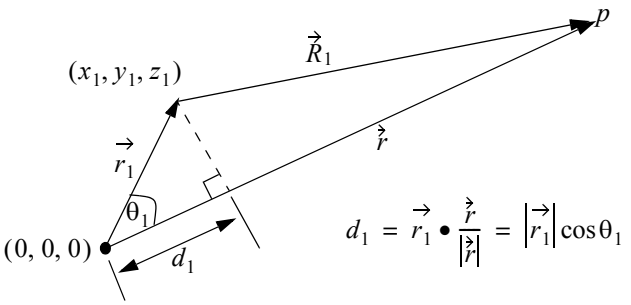

Figure 8.2 Geometry for an array antenna. Single element

Fig. 8.2 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

$$E(\theta, \phi) = I_0 \frac{e^{-jkR_1}}{R_1} f(\theta, \phi) \tag{8.16}$$

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. 8.3. 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

$$\vec{r}_i = \hat{a}_x x_i + \hat{a}_y y_i + \hat{a}_z z_i \tag{8.17}$$

The far field components that constitute the total electric field are

$$E_i(\theta, \phi) = I_i \frac{e^{-jkR_i}}{R_i} f(\theta_i, \phi_i) \tag{8.18}$$

where

$$\begin{aligned} R_i &= \left|\vec{R}_i\right| = \left|\vec{r} - \vec{r}_i\right| = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \\ &= r\sqrt{1 + (x_i^2 + y_i^2 + z_i^2)/r^2 - 2(xx_i + yy_i + zz_i)/r^2} \end{aligned} \tag{8.19}$$

Using spherical coordinates, where $x = r\sin\theta\cos\varphi$, $y = r\sin\theta\sin\varphi$, and $z = r\cos\theta$ yields

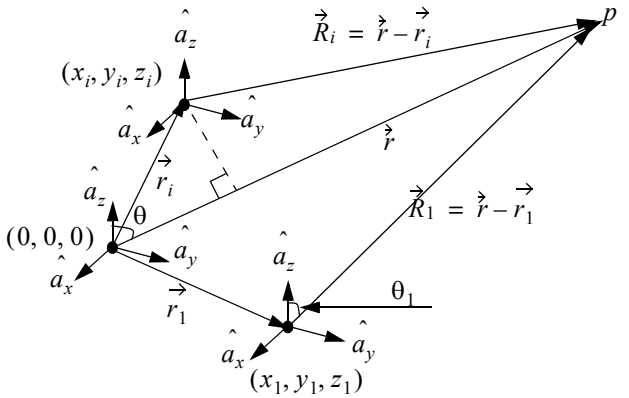

Figure 8.3 Geometry for an array antenna.

$$\frac{(x_i^2 + y_i^2 + z_i^2)}{r^2} = \frac{|\vec{r}_i|}{r^2} \ll 1 \tag{8.20}$$

Thus, a good approximation (using binomial expansion) for Eq. (8.19) is

$$R_i = r - r(x_i \sin\theta\cos\phi + y_i \sin\theta\sin\phi + z_i \cos\theta) \tag{8.21}$$

It follows that the phase contribution at the far field point from the ith radiator with respect to the phase reference is

$$e^{-jkR_i} = e^{-jkr}\ e^{jk(x_i \sin\theta\cos\phi + y_i \sin\theta\sin\phi + z_i \cos\theta)} \tag{8.22}$$

Remember, however, that the unit vector $\vec{r}_0$ along the vector $\vec{r}$ is

$$\vec{r}_0 = \frac{\vec{r}}{|\vec{r}|} = \hat{a}_x \sin\theta\cos\phi + \hat{a}_y \sin\theta\sin\phi + \hat{a}_z \cos\theta \tag{8.23}$$

Hence, we can rewrite Eq. (8.22) as

$$e^{-jkR_i} = e^{-jkr}\ e^{jk(\vec{r}_i \bullet \vec{r}_0)} = e^{-jkr} e^{j\Psi_i(\theta, \phi)} \tag{8.24}$$

Finally, by virtue of superposition, the total electric field is

$$E(\theta, \phi) = \sum_{i=1}^{N} I_i e^{j\Psi_i(\theta, \phi)} \tag{8.25}$$

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 lobes' locations.

8.4. Linear Arrays

Fig. 8.4 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 $kd\sin\psi$, where $k = 2\pi/\lambda$. The combined phase at the far field observation point P is independent of ϕ and is computed from Eq. (8.24) as

$$\Psi(\psi, \phi) = k(\vec{r}_i \bullet \vec{r}_0) = (n-1)kd\sin\psi \tag{8.26}$$

Thus, from Eq. (8.25), the electric field at a far field observation point with direction-sine equal to $\sin\psi$ (assuming isotropic elements) is

$$E(\sin\psi) = \sum_{n=1}^{N} e^{j(n-1)(kd\sin\psi)} \tag{8.27}$$

Expanding the summation in Eq. (8.27) yields

$$E(\sin\psi) = 1 + e^{jkd\sin\psi} + \ldots + e^{j(N-1)(kd\sin\psi)} \tag{8.28}$$

The right-hand side of Eq. (8.29) is a geometric series, which can be expressed in the form

$$1 + a + a^2 + a^3 + \ldots + a^{(N-1)} = \frac{1-a^N}{1-a} \tag{8.29}$$

Replacing a by $e^{jkd\sin\psi}$ yields

$$E(\sin\psi) = \frac{1-e^{jNkd\sin\psi}}{1-e^{jkd\sin\psi}} = \frac{1-(\cos Nkd\sin\psi)-j(\sin Nkd\sin\psi)}{1-(\cos kd\sin\psi)-j(\sin kd\sin\psi)} \tag{8.30}$$

The far field array intensity pattern is then given by

$$|E(\sin\psi)| = \sqrt{E(\sin\psi)E^*(\sin\psi)} \tag{8.31}$$

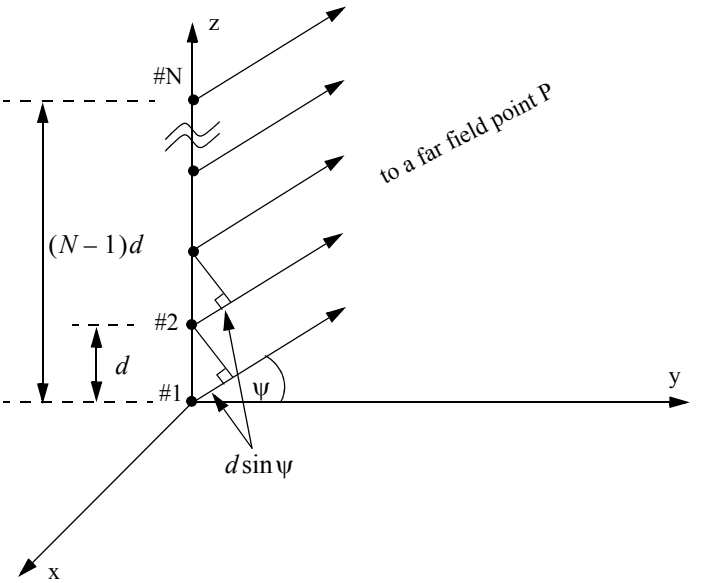

Figure 8.4. Linear array of equally spaced elements.

Substituting Eq. (8.30) into Eq. (8.31) and collecting terms yield

$$|E(\sin\psi)| = \sqrt{\frac{(1-\cos Nkd\sin\psi)^2 + (\sin Nkd\sin\psi)^2}{(1-\cos kd\sin\psi)^2 + (\sin kd\sin\psi)^2}} = \sqrt{\frac{1-\cos Nkd\sin\psi}{1-\cos kd\sin\psi}} \tag{8.32}$$

and using the trigonometric identity $1-\cos\theta = 2(\sin\theta/2)^2$ yields

$$|E(\sin\psi)| = \left|\frac{\sin(Nkd\sin\psi/2)}{\sin(kd\sin\psi/2)}\right| \tag{8.33}$$

which is a periodic function of $kd\sin\psi$, with a period equal to 2π.

The maximum value of $|E(\sin\psi)|$, which occurs at $\psi = 0$, is equal to N. It follows that the normalized intensity pattern is equal to

$$|E_n(\sin\psi)| = \frac{1}{N}\left|\frac{\sin((Nkd\sin\psi)/2)}{\sin((kd\sin\psi)/2)}\right| \tag{8.34}$$

The normalized two-way array pattern (radiation pattern) is given by

$$G(\sin\psi) = \left|E_n(\sin\psi)\right|^2 = \frac{1}{N^2}\left(\frac{\sin((Nkd\sin\psi)/2)}{\sin((kd\sin\psi)/2)}\right)^2 \qquad \textbf{(8.35)}$$

Fig. 8.5 shows a plot of Eq. (8.35) versus $\sin\theta$ for $N = 8$. The radiation pattern $G(\sin\psi)$ has cylindrical symmetry about its axis $(\sin\psi = 0)$, and is independent of the azimuth angle. Thus, it is completely determined by its values within the interval $(0 < \psi < \pi)$. This plot can be reproduced using MATLAB program *"fig8_5.m"* given in Listing 8.1 in Section 8.8.

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 $\sin\psi_0$ is accomplished by making the phase difference between any two adjacent elements equal to $kd\sin\psi_0$. In this case, the normalized radiation pattern can be written as

$$G(\sin\psi) = \frac{1}{N^2}\left(\frac{\sin[(Nkd/2)(\sin\psi - \sin\psi_0)]}{\sin[(kd/2)(\sin\psi - \sin\psi_0)]}\right)^2 \qquad \textbf{(8.36)}$$

If $\psi_0 = 0$ 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. (8.35) are zeros. More precisely,

$$\frac{kd\sin\psi}{2} = \pm m\pi \qquad ; \; m = 0, 1, 2, \ldots \qquad \textbf{(8.37)}$$

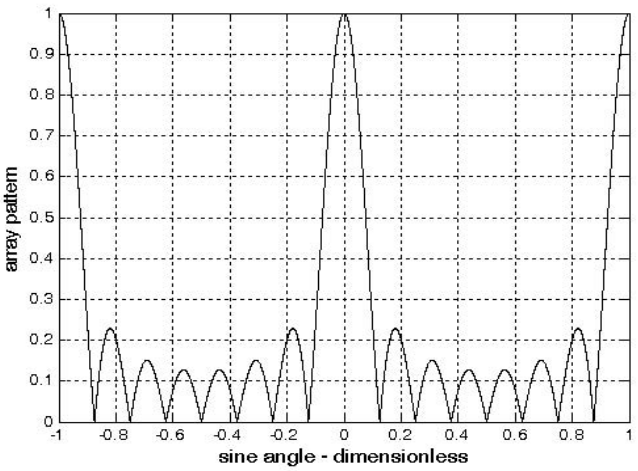

Figure 8.5a. Normalized radiation pattern for a linear array; $N = 8$; $d = \lambda$.

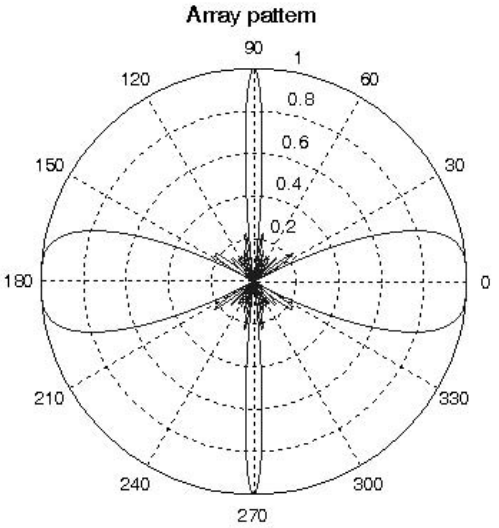

Figure 8.5b. Polar plot for the array pattern in Fig. 8.5a.

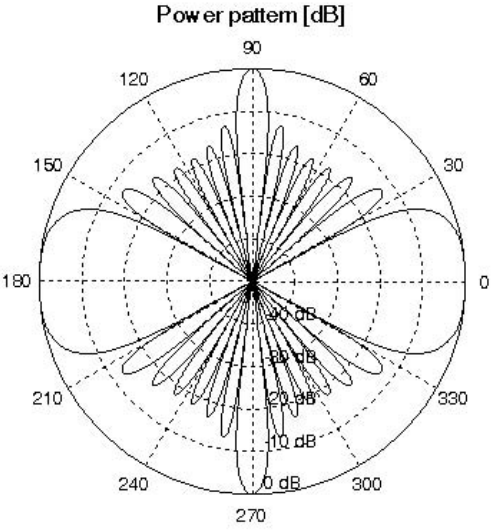

Figure 8.5c. Polar plot for the power pattern in Fig. 8.5a.

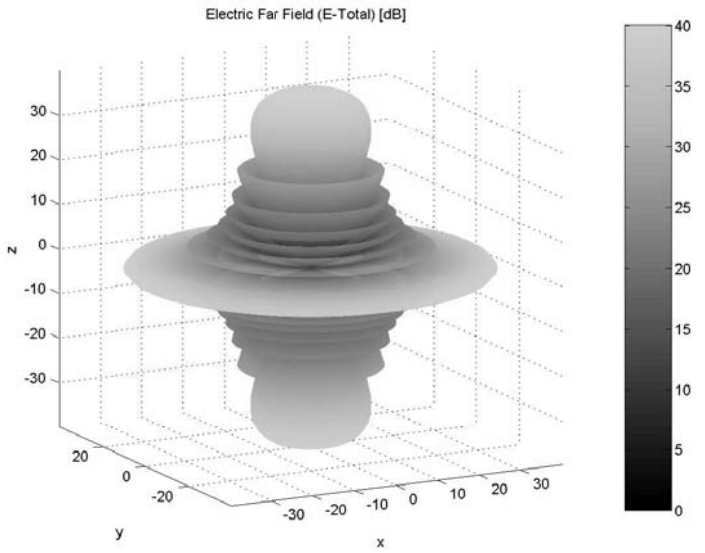

Figure 8.5d. Three-dimensional plot for the radiation pattern in Fig. 8.5a.

Solving for ψ yields

$$\psi_m = \text{asin}\left(\pm\frac{\lambda m}{d}\right) \qquad ; \; m = 0, 1, 2, \ldots \tag{8.38}$$

where the subscript m is used as a maxima indicator. The first maximum occurs at $\psi_0 = 0$, 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. (8.38) is greater than unity; it follows that $d < \lambda$. Under this condition, the main lobe is assumed to be at $\psi = 0$ (broadside array). Alternatively, when electronic beam steering is considered, the grating lobes occur at

$$\left|\sin\psi - \sin\psi_0\right| = \pm\frac{\lambda n}{d} \qquad ; \; n = 1, 2, \ldots \tag{8.39}$$

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. (8.35) is maximum, or equivalently

$$\frac{Nkd\sin\psi}{2} = \pm(2l+1)\frac{\pi}{2} \qquad ;\ l = 1, 2, \ldots \tag{8.40}$$

Solving for ψ yields

$$\psi_l = \operatorname{asin}\left(\pm\frac{\lambda}{2d}\frac{2l+1}{N}\right) \qquad ;\ l = 1, 2, \ldots \tag{8.41}$$

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. (8.36) is zero. More precisely,

$$\frac{N}{2}kd\sin\psi = \pm n\pi \qquad ;\ \begin{matrix} n = 1, 2, \ldots \\ n \neq N, 2N, \ldots \end{matrix} \tag{8.42}$$

Again solving for ψ yields

$$\psi_n = \operatorname{asin}\left(\pm\frac{\lambda}{d}\frac{n}{N}\right) \qquad ;\ \begin{matrix} n = 1, 2, \ldots \\ n \neq N, 2N, \ldots \end{matrix} \tag{8.43}$$

where the subscript n is used as a null indicator. Define the angle which corresponds to the half power point as ψ_h. It follows that the half power (3 dB) beamwidth is $2\left|\psi_m - \psi_h\right|$. This occurs when

$$\frac{N}{2}kd\sin\psi_h = 1.391 \ \ radians \Rightarrow \psi_h = \operatorname{asin}\left(\frac{\lambda}{2\pi d}\frac{2.782}{N}\right) \tag{8.44}$$

8.4.1. Array Tapering

Fig. 8.6a 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 about $13.46\ \ dB$ below the main lobe, and for most radar applications this may not be sufficient. Fig. 8.6b shows the 3-D plot for the radiation pattern shown in Fig. 8.6.a.

In order to reduce the sidelobe levels, the array must be designed to radiate more power towards 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 MATLAB signal processing toolbox provides users with a wide variety of built-in windows. This list includes: *"Bartlett, Barthannwin, Blackmanharris, Bohmanwin, Chebwin, Gausswin, Hamming, Hann, Kaiser, Nuttallwin, Rectwin, Triang, and Tukeywin."*

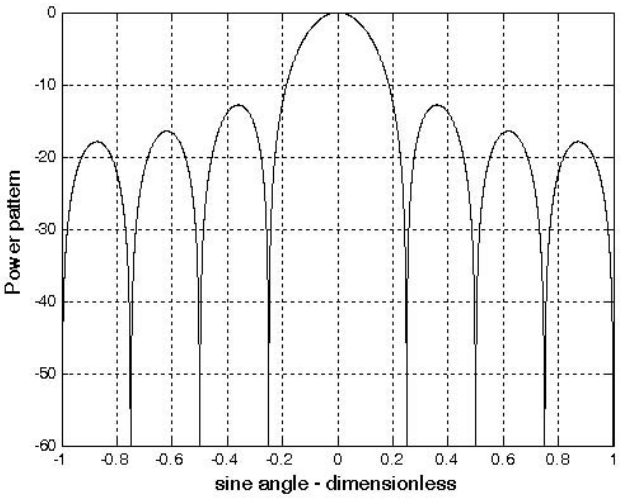

Figure 8.6a. Normalized pattern for a linear array. $N = 8$, $d = \lambda/2$.

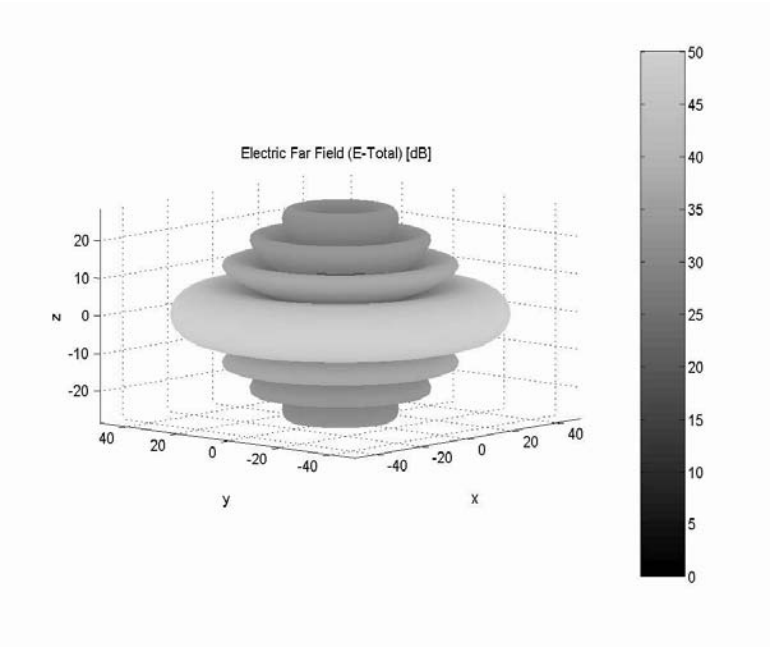

Figure 8.6b. Three-dimensional plot for the radiation pattern in Fig. 8.6a.

Table 8.1 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. 8.7.

TABLE 8.1. 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*

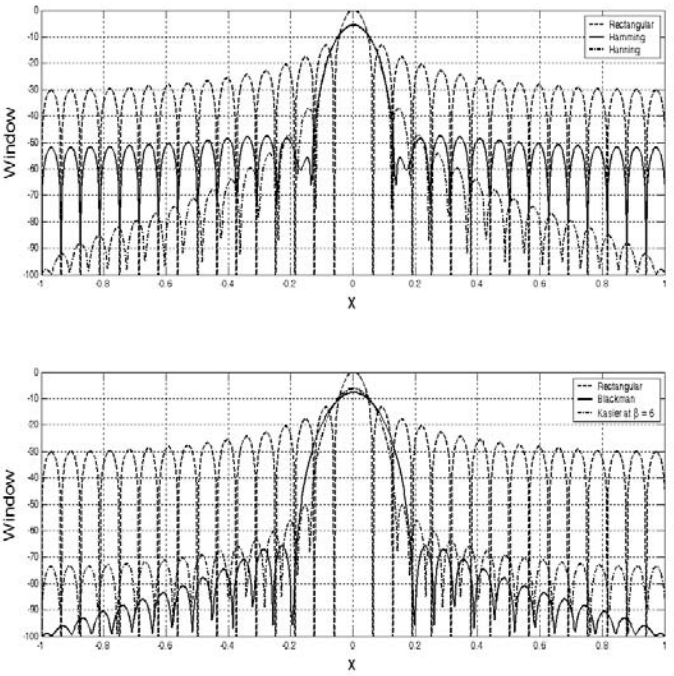

Figure 8.7. Most common windows. This figure can be reproduced using MATLAB program *"fig8_7.m"* given in Listing 8.2 in Section 8.8.

8.4.2. Computation of the Radiation Pattern via the DFT

Fig. 8.8 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 $\sin\psi$ is

$$E(\sin\psi) = \sum_{n=0}^{N-1} w(n) e^{j\Delta\phi\left(n-\left(\frac{N-1}{2}\right)\right)} \tag{8.45}$$

where in this case the phase reference is taken as the physical center of the array, and

$$\Delta\phi = \frac{2\pi d}{\lambda}\sin\psi \tag{8.46}$$

Expanding Eq. (8.45) and factoring the common phase term $\exp[j(N-1)\Delta\phi/2]$ yield

$$E(\sin\psi) = e^{j(N-1)\Delta\phi/2}\{w(0)e^{-j(N-1)\Delta\phi} + w(1)e^{-j(N-2)\Delta\phi} + \ldots + w(N-1)\} \tag{8.47}$$

By using the symmetry property of a window sequence (remember that a window must be symmetrical about its central point), we can rewrite Eq. (8.47) as

$$E(\sin\psi) = e^{j\phi_0}\{w(N-1)e^{-j(N-1)\Delta\phi} + w(N-2)e^{-j(N-2)\Delta\phi} + \ldots + w(0)\} \tag{8.48}$$

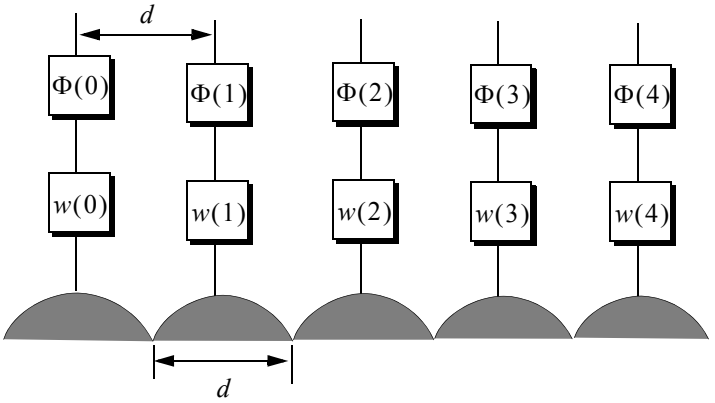

Figure 8.8. Linear array of size 5, with tapering and phase shifting hardware.

where $\phi_0 = (N-1)\Delta\phi/2$.

Define $\{V_1^n = \exp(-j\Delta\phi n); n = 0, 1, \ldots, N-1\}$. It follows that

$$E(\sin\psi) = e^{j\phi_0}[w(0) + w(1)V_1^1 + \ldots + w(N-1)V_1^{N-1}] \tag{8.49}$$

$$= e^{j\phi_0}\sum_{n=0}^{N-1} w(n)V_1^n$$

The discrete Fourier transform of the sequence $w(n)$ is defined as

$$W(q) = \sum_{n=0}^{N-1} w(n)e^{-\frac{(j2\pi nq)}{N}} \quad ; \; q = 0, 1, \ldots, N-1 \tag{8.50}$$

The set $\{\sin\psi_q\}$ which makes V_1 equal to the DFT kernel is

$$\sin\psi_q = \frac{\lambda q}{Nd} \quad ; \; q = 0, 1, \ldots, N-1 \tag{8.51}$$

Then by using Eq. (8.51) in Eq. (8.50) yields

$$E(\sin\psi) = e^{j\phi_0}W(q) \tag{8.52}$$

The one-way array pattern is computed as the modulus of Eq. (8.52). It follows that the one-way radiation pattern of a tapered linear array of circular dishes is

$$G(\sin\psi) = G_e\,|W(q)| \tag{8.53}$$

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 (sine the steering angle). It is given in Listing 8.3 in Section 8.8. 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 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*
theta	*real angle available for steering*	*degrees*	*output*
patternr	*array pattern*	*dB*	*output*
patterng	*gain pattern*	*dB*	*output*

A MATLAB based GUI workspace called *"linear_array_gui.m"*[1] was developed for this function. It shown in Fig. 8.9.

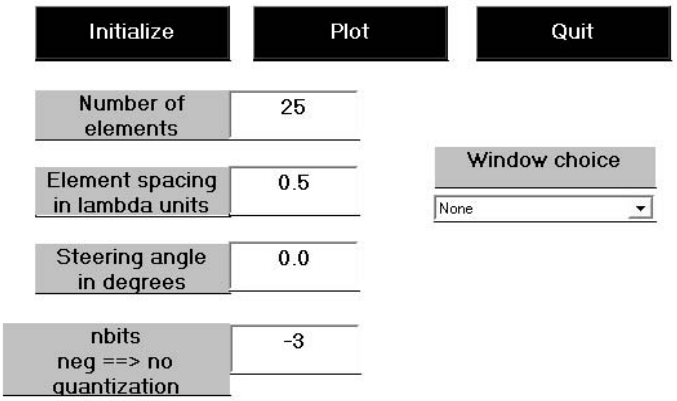

Figure 8.9. MATLAB GUI workspace associated with the function *"linear_array.m".*

1. The MATLAB "Signal Processing" Toolbox is required to execute this program.

Figs. 8.10 through 8. 18 respectively show plots of the array gain pattern versus steering angle for the following cases:

[theta, patternr, patterng] = linear_array(25, 0.5, 0, -1, -1, -3);

[theta, patternr, patterng] = linear_array(25, 0.5, 0, 1, 'Hamming', -3);

[theta, patternr, patterng] = linear_array(25, 0.5, 5, -1, -1, 3);

[theta, patternr, patterng] = linear_array(25, 0.5, 5, 1, 'Hamming', 3);

[theta, patternr, patterng] = linear_array(25, 0.5, 25, 1, 'Hamming', 3);

[theta, patternr, patterng] = linear_array(25, 1.5, 40, -1, -1, -3);

[theta, patternr, patterng] = linear_array(25, 1.5, 40, 1, 'Hamming', -3);

[theta, patternr, patterng] = linear_array(25, 1.5, -40, -1, -1, 3);

[theta, patternr, patterng] = linear_array(25, 1.5, -40, 1, 'Hamming', 3);

Users are advised to utilize the GUI developed for this function and test a few cases of their own.

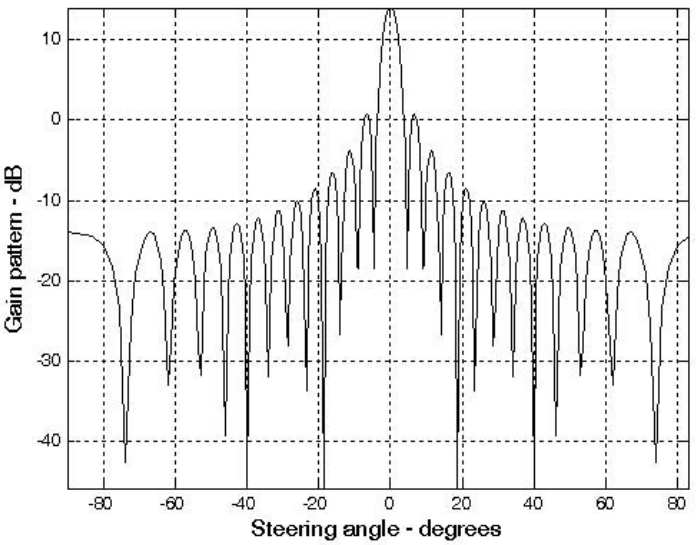

Figure 8.10. Array gain pattern: $Nr = 25;\ dolr = 0.5;\ \theta_0 = 0°;$ $win = none;\ nbits = -3.$

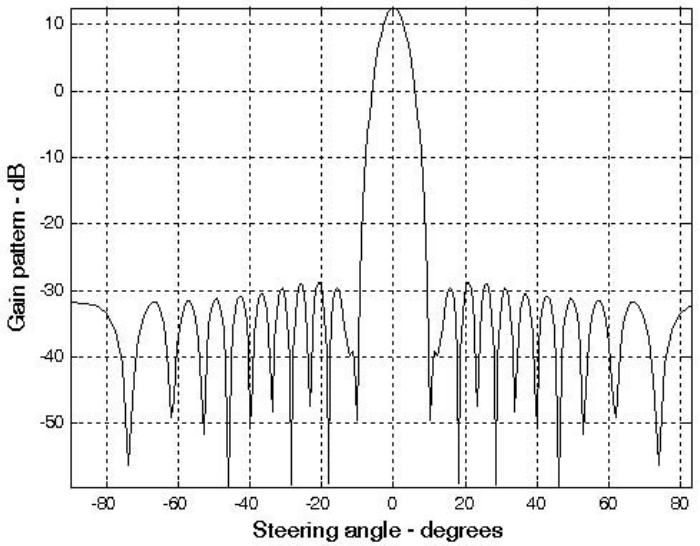

Figure 8.11. Array gain pattern: $Nr = 25;\ dolr = 0.5;\ \theta_0 = 0°;$ $win = Hamming;\ nbits = -3$

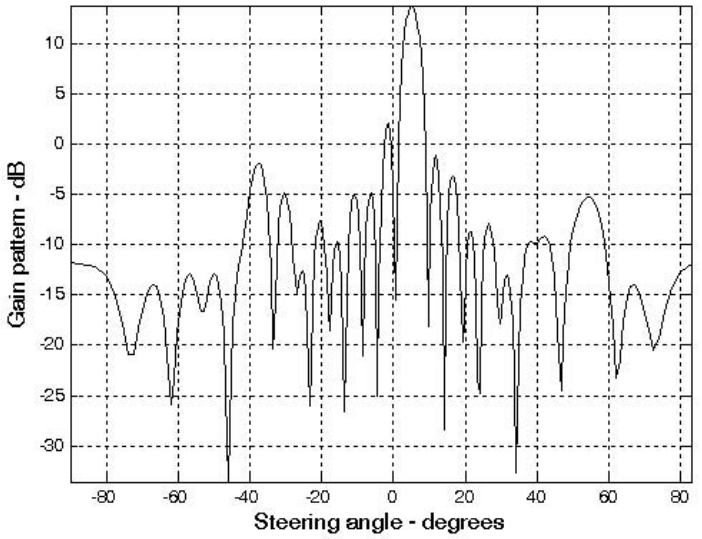

Figure 8.12. Array gain pattern: $Nr = 25;\ dolr = 0.5;\ \theta_0 = 5°;$ $win = none;\ nbits = 3$

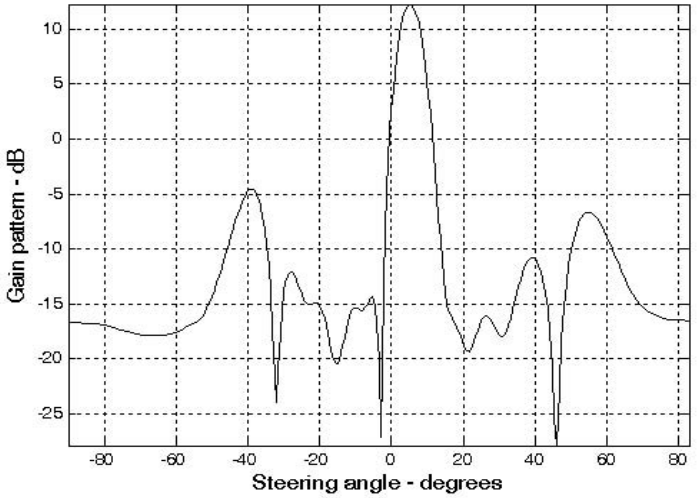

Figure 8.13. Array gain pattern: $Nr = 25;\ dolr = 0.5;\ \theta_0 = 5°;$ $win = Hamming;\ nbits = 3$

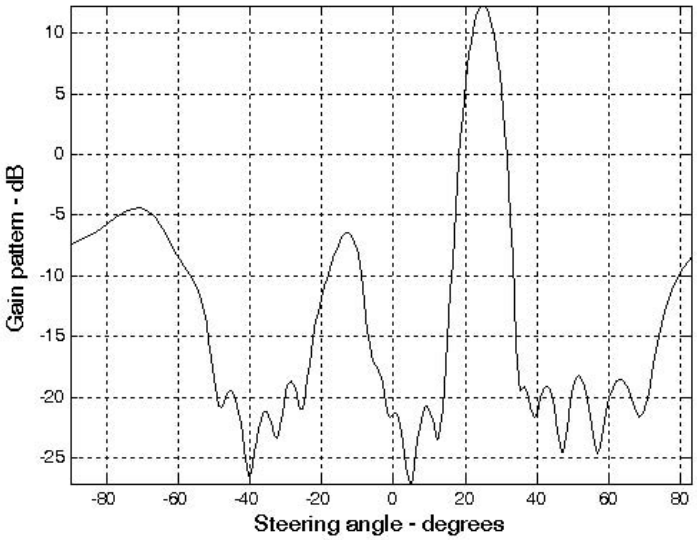

Figure 8.14. Array gain pattern: $Nr = 25;\ dolr = 0.5;\ \theta_0 = 25°;$ $win = Hamming;\ nbits = 3$

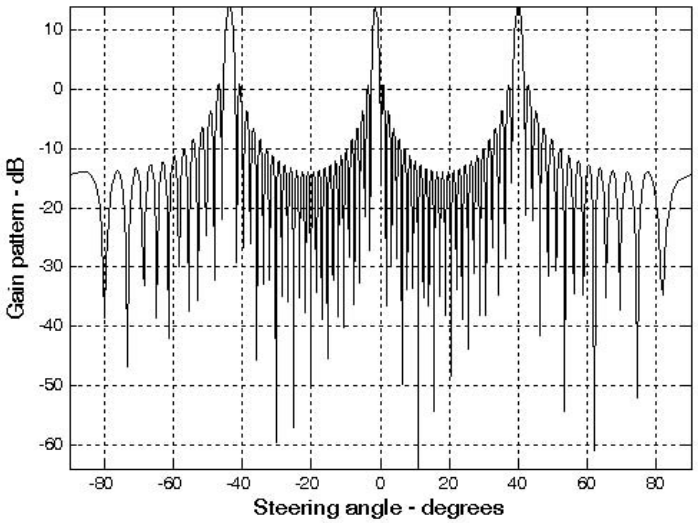

Figure 8.15. Array gain pattern: $Nr = 25;\ dolr = 1.5;\ \theta_0 = 40°$ **;** $win = none;\ nbits = -3$

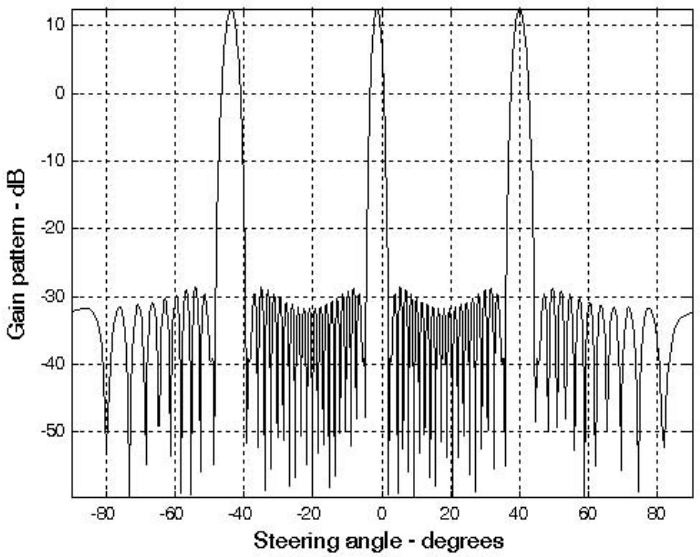

Figure 8.16. Array gain pattern: $Nr = 25;\ dolr = 1.5;\ \theta_0 = 40°$ **;** $win = Hamming;\ nbits = -3$

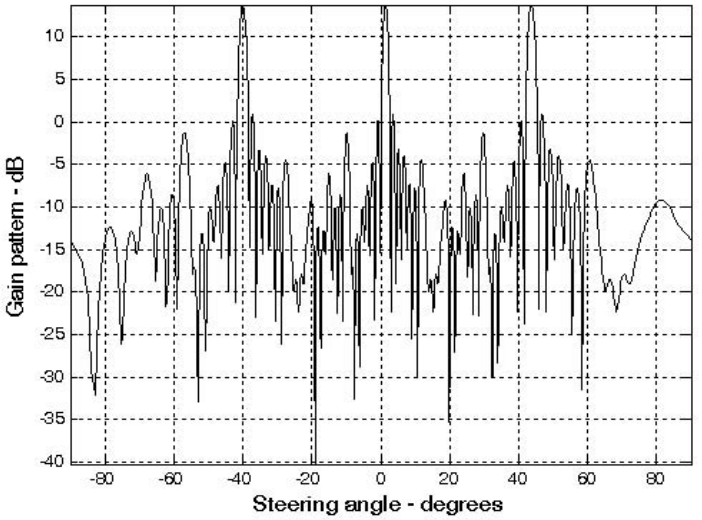

Figure 8.17. Array gain pattern: $Nr = 25$; $dolr = 1.5$; $\theta_0 = -40°$; $win = none$; $nbits = 3$

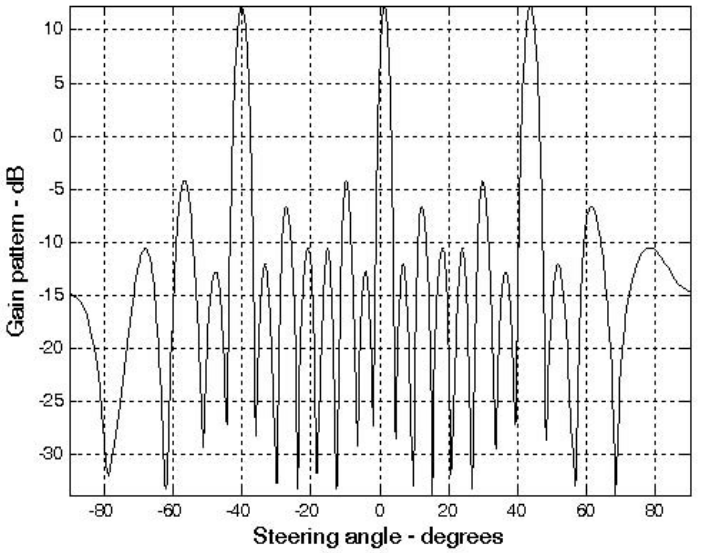

Figure 8.18. Array gain pattern: $Nr = 25$; $dolr = 1.5$; $\theta_0 = -40°$; $win = Hamming$; $nbits = 3$

8.5. *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. 8.19.

Planar arrays can be steered in elevation and azimuth $((\theta, \phi)$, as illustrated in Fig. 8.20 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. (8.24) and (8.25).

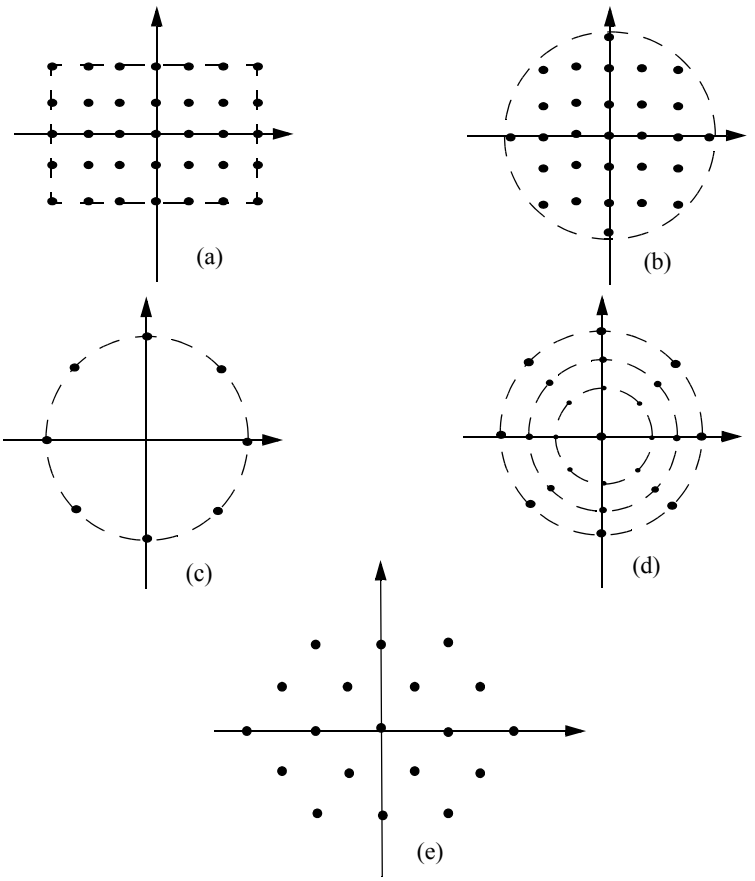

Figure 8.19. Planar array grids. (a) Rectangular; (b) Rectangular with circular boundary; (c) Circular; (d) Concentric circular; and (e) Hexagonal.

Rectangular Grid Arrays

Consider the $N \times M$ rectangular grid as shown in Fig. 8.20. 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, given by

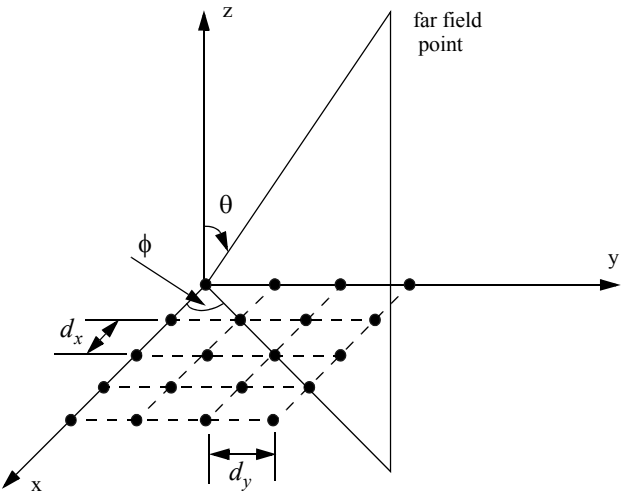

Figure 8.20. Rectangular array geometry.

$$E_x(\theta, \phi) = \sum_{n=1}^{N} I_{x_n} e^{j(n-1)kd_x \sin\theta \cos\phi} \tag{8.54}$$

$$E_y(\theta, \phi) = \sum_{m=1}^{N} I_{y_m} e^{j(m-1)kd_y \sin\theta \sin\phi} \tag{8.55}$$

The total electric field at the far field observation point is then given by

$$E(\theta, \phi) = E_x(\theta, \phi) E_y(\theta, \phi) = \tag{8.56}$$

$$\left(\sum_{m=1}^{N} I_{y_m} e^{j(m-1)kd_y \sin\theta \sin\phi} \right) \left(\sum_{n=1}^{N} I_{x_n} e^{j(n-1)kd_x \sin\theta \cos\phi} \right)$$

Eq. (8.56) can be expressed in terms of the directional cosines

$$\begin{aligned} u &= \sin\theta\cos\phi \\ v &= \sin\theta\sin\phi \end{aligned} \tag{8.57a}$$

$$\phi = \operatorname{atan}\left(\frac{u}{v}\right) \tag{8.57b}$$

$$\theta = \operatorname{asin}\sqrt{u^2 + v^2}$$

The visible region is then defined by

$$\sqrt{u^2 + v^2} \le 1 \tag{8.58}$$

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. Fig. 8.21 shows how a beam steered in a certain θ, ϕ direction is translated into U, V space.

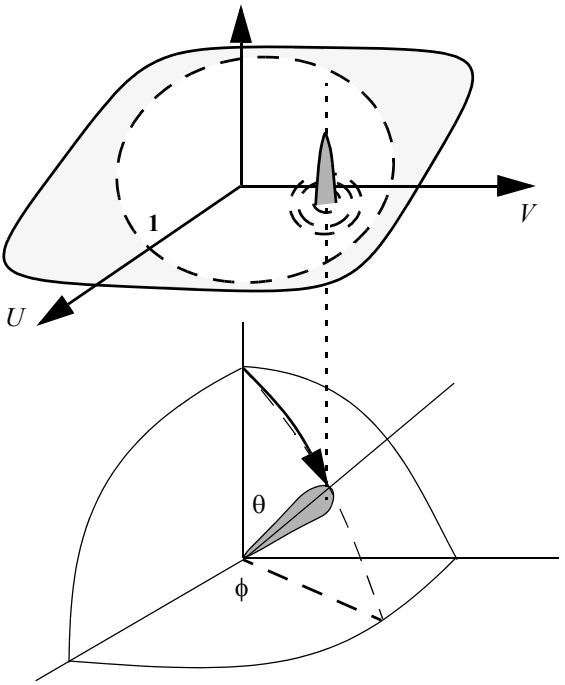

Figure 8.21. Translation from spherical coordinates into U,V space.

The rectangular array one-way intensity pattern is then equal to the product of the individual patterns. More precisely for a uniform excitation ($I_{y_m} = I_{x_n} = const$),

$$E(\theta, \phi) = \left|\frac{\sin((Nkd_x \sin\theta\cos\phi)/2)}{\sin((kd_x \sin\theta\cos\phi)/2)}\right| \left|\frac{\sin((Nkd_y \sin\theta\sin\phi)/2)}{\sin((kd_y \sin\theta\sin\phi)/2)}\right| \quad \textbf{(8.59)}$$

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.

Circular Grid Arrays

The geometry of interest is shown in Fig. 8.19c. 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. 8.22. From the geometry

$$\Phi_n = \frac{2\pi}{N} n \qquad ; n = 1, 2, \ldots, N \quad \textbf{(8.60)}$$

The coordinates of the nth element are

$$\begin{aligned} x_n &= a \cos\Phi_n \\ y_n &= a \sin\Phi_n \\ z_n &= 0 \end{aligned} \quad \textbf{(8.61)}$$

It follows that

$$k(\hat{r}_n \bullet \hat{r}_0) = \Psi_n = k(a\sin\theta\cos\phi\cos\Phi_n + a\sin\theta\sin\phi\sin\Phi_n + 0) \quad \textbf{(8.62)}$$

which can be rearranged as

$$\Psi_n = ak\sin\theta(\cos\phi\cos\Phi_n + \sin\phi\sin\Phi_n) \quad \textbf{(8.63)}$$

Then by using the identity $\cos(A - B) = \cos A\cos B + \sin A\sin B$, Eq.(8.63) collapses to

$$\Psi_n = ak\sin\theta\cos(\Phi_n - \phi) \quad \textbf{(8.64)}$$

Finally by using Eq. (8.25), the far field electric field is then given by

$$E(\theta, \phi; a) = \sum_{n=1}^{N} I_n \exp\left\{j\frac{2\pi a}{\lambda}\sin\theta\cos(\Phi_n - \phi)\right\} \quad \textbf{(8.65)}$$

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. (8.65) takes on the following form

$$E(\theta, \phi; a) = \sum_{n=1}^{N} I_n \exp\left\{ j\frac{2\pi a}{\lambda}[\sin\theta\cos(\Phi_n - \phi) - \sin\theta_0\cos(\Phi_n - \phi_0)] \right\} \quad \textbf{(8.66)}$$

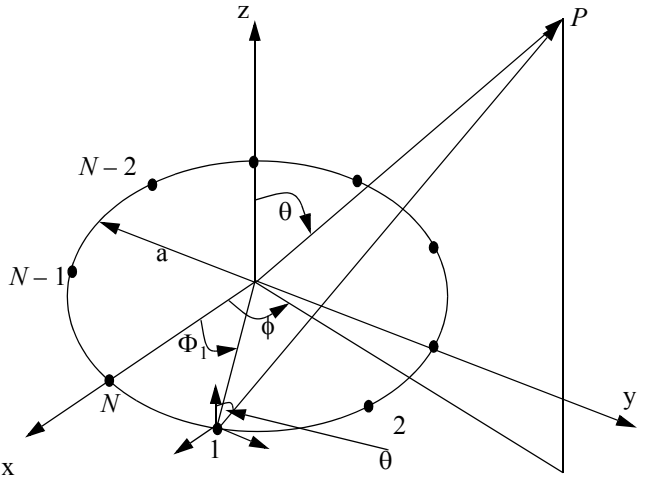

Figure 8.22. Geometry for a circular array.

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. It is given in Listing 8.4 in Section 8.8. The input parameters to this program include:

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*

Symbol	Description	Units
phid	*constant* ϕ *plane*	*degrees*
thetad	*constant* θ *plane*	*degrees*

Consider the case when the inputs are:

a	*1.5*
N	*10 dipole antennas*
theta0	$\theta_0 = 45°$
phi0	$\phi_0 = 60°$
Variations	*'Theta'*
phid	$\phi_d = 60°$
thetad	$\theta_d = 45°$

Fig.s 8.23 and 8.24 respectively show the array pattern in relative amplitude and the power pattern versus the angle θ. Figs. 8.25 and 8.26 are similar to Figs. 8.23 and 8.24 except in this case the patterns are plotted in polar coordinates.

Fig. 8.27 shows a plot of the normalized single element pattern (upper left corner), the normalized array factor (upper right corner), and the total array pattern (lower left corner). Fig. 8.28 shows the 3-D pattern for this example in the θ, ϕ space.

Figs. 8.29 through 8.33 are similar to those in Figs. 8.23 through 8.27, except in this case the input parameters are given by:

a	*1.5*
N	*10 dipole antennas*
theta0	$\theta_0 = 45°$
phi0	$\phi_0 = 60°$
Variations	*'Phi'*
phid	$\phi_d = 60°$
thetad	$\theta_d = 45°$

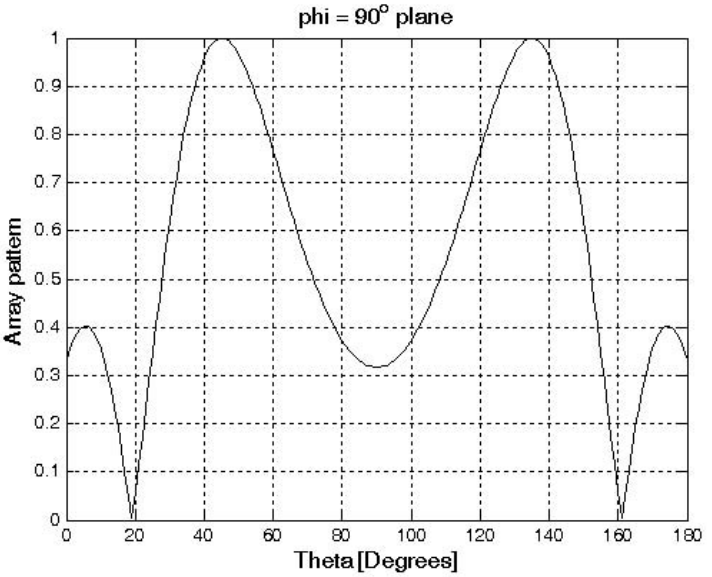

Figure 8.23. Array factor pattern for a circular array, using the parameters defined in the table on top of page 346 (rectangular coordinates).

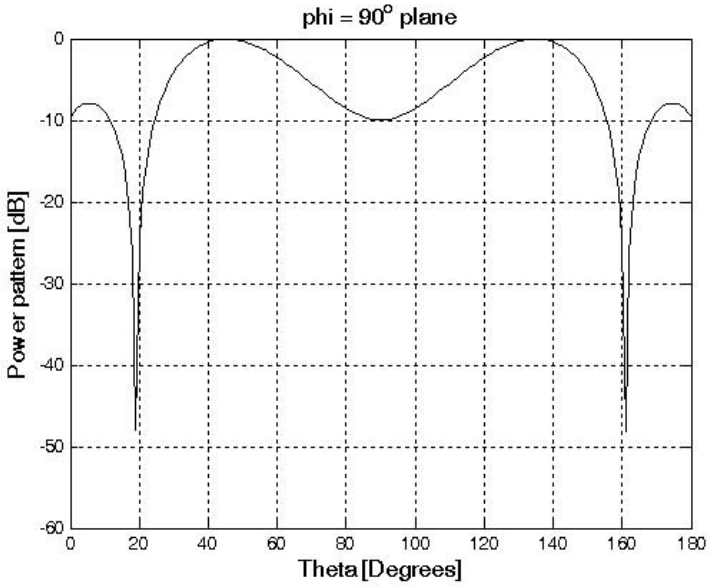

Figure 8.24. Same as Fig. 8.23 using dB scale.

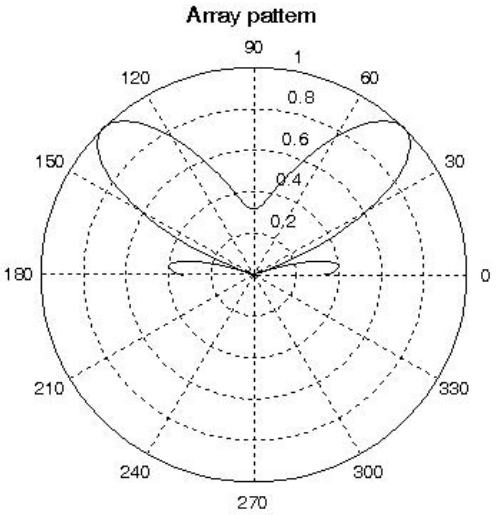

Figure 8.25. Array factor pattern for a circular array, using the parameters defined in the table on top of page 346 (polar coordinates).

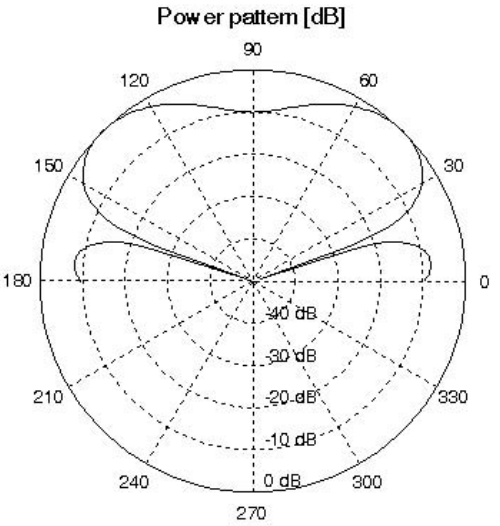

Figure 8.26. Same as Fig. 8.25 using dB scale.

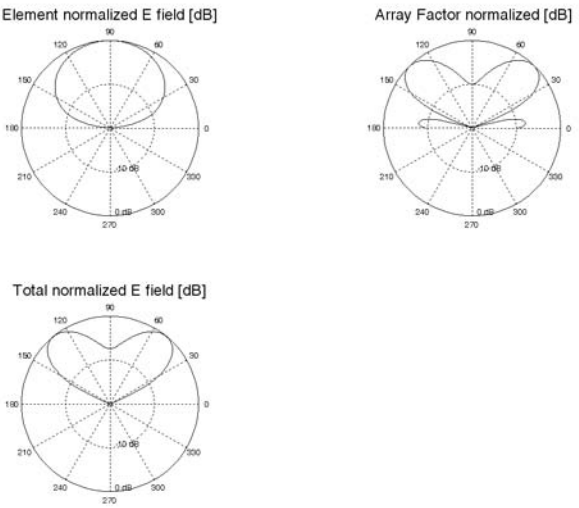

Figure 8.27. Element, array factor, and total pattern for the circular array defined in the table on top of page 346.

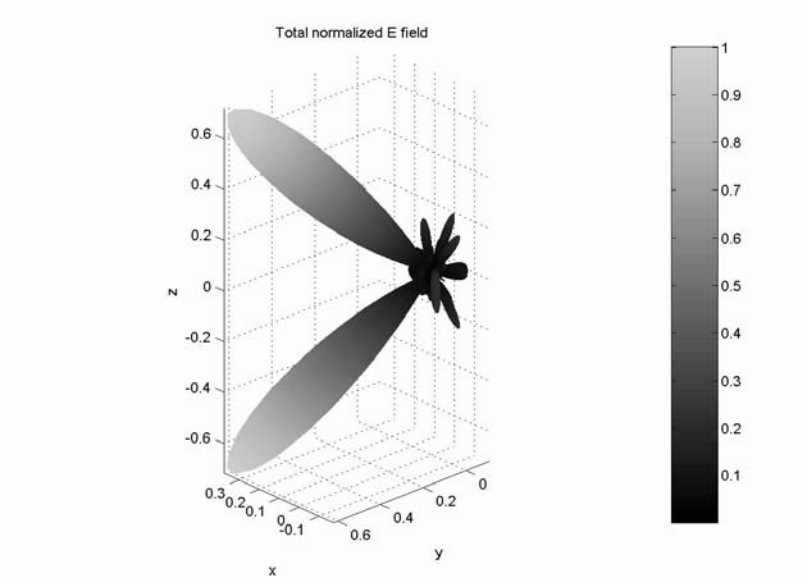

Figure 8.28. 3-D total array pattern (in θ, ϕ space) for the circular array defined in the table on top of page 346.

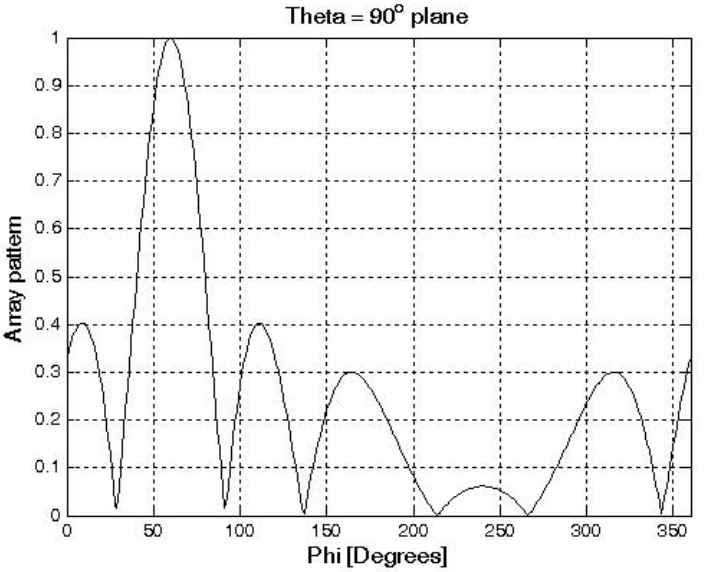

Figure 8.29. Array factor pattern for a circular array, using the parameters defined in the table on bottom of page 346 (rectangular coordinates).

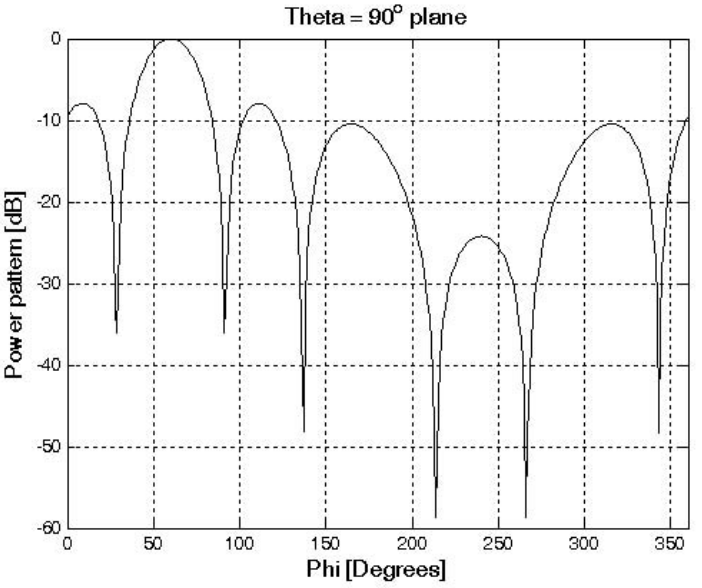

Figure 8.30. Same as Fig. 8.29 using dB scale.

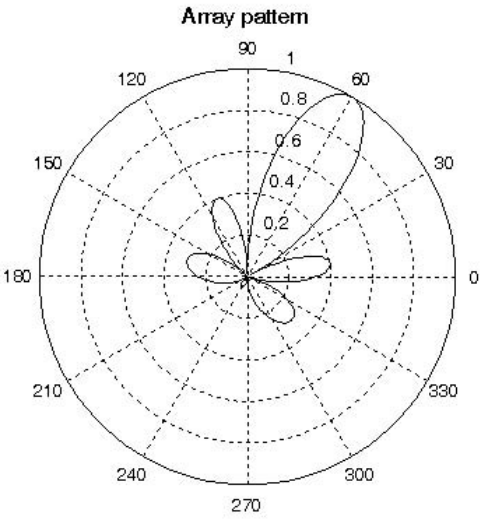

Figure 8.31. Array factor pattern for a circular array, using the parameters defined in the table on bottom of page 346 (polar coordinates).

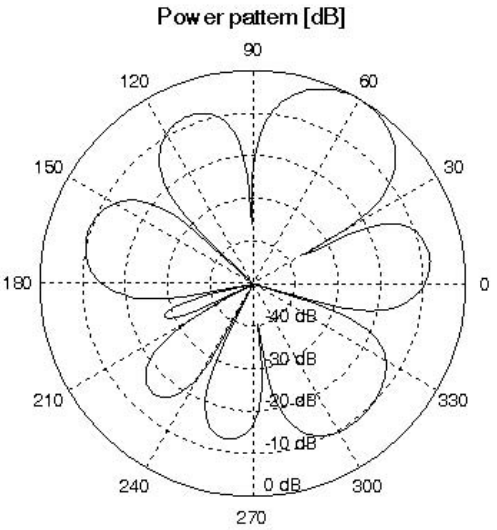

Figure 8.32. Same as Fig. 8.31 using dB scale.

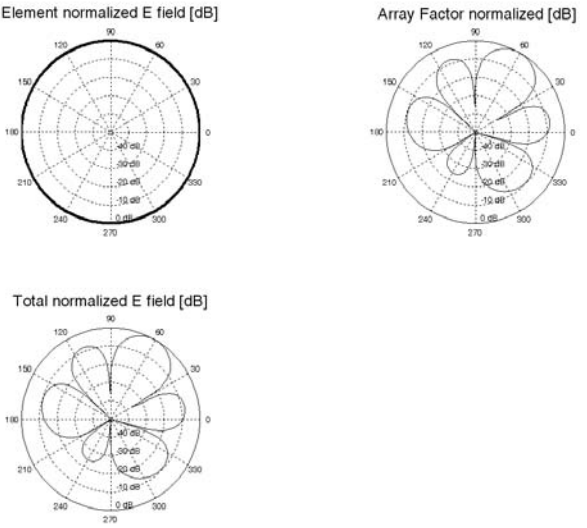

Figure 8.33. Element, array factor, and total pattern for the circular array defined in the table on bottom of page 346.

Concentric Grid Circular Arrays

The geometry of interest is shown in Fig. 8.19d and Fig. 8.34. In this case, N_2 elements are distributed equally on the outer circle whose radius is a_2, while other 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 pattern is derived in two steps. First, the array pattern corresponding to the linearly distributed concentric circular arrays with N_1 and N_2 elements and the center element are computed separately. Second, the overall array pattern corresponding to the two concentric arrays and the center element are added. The element pattern of the identical antenna elements are considered in the first step. Thus, the total pattern becomes,

$$E(\theta, \phi) = E_0(\theta, \phi) + E_1(\theta, \phi; a_1) + E_2(\theta, \phi; a_2) \qquad \textbf{(8.67)}$$

Fig. 8.35 shows a 3-D plot for concentric circular array in the θ, ϕ space for the following parameters:

a_1	N_1	a_2	N_2
1 λ	*8 (*λ/2 *dipoles)*	*2*λ	*8 (*λ/2 *dipoles)*

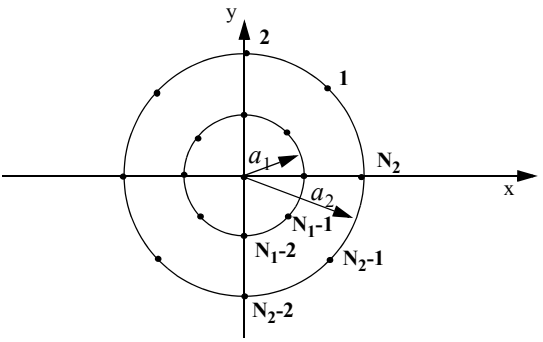

Figure 8.34. Concentric circular array geometry.

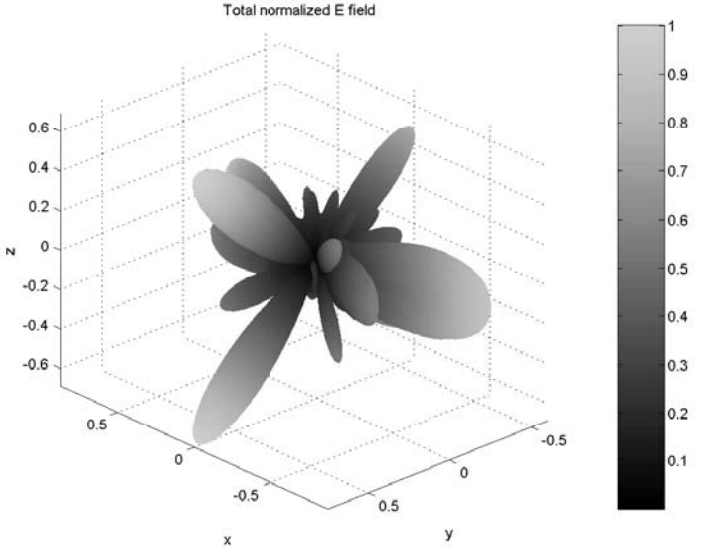

Figure 8.35. 3-D array pattern for a concentric circular array; $\theta = 45°$ **and** $\phi = 90°$

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

$$r_d = \frac{N_d - 1}{2} + \Delta x \tag{8.68}$$

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

$$a(m, n) = \begin{cases} 1 & , \; if \;\; dis \;\; to \;\; (m, n)th \;\; element < r_d \\ 0 & ; \; elsewhere \end{cases} \tag{8.69}$$

where distance, dis, is measured from the center of the circle. This is illustrated in Fig. 8.36.

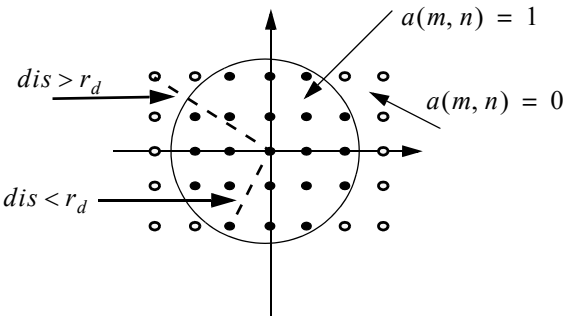

Figure 8.36. Elements with solid dots have $a(m, n) = 1$; other elements have $a(m, n) = 0$.

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

$$d_y = \frac{\sqrt{3}}{2} \; d_x \tag{8.70}$$

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

$$N_r = N_x - |m| \tag{8.71}$$

The electric field at a far field observation point is computed using Eq. (8.24) and (8.25). The phase associated with $(m, n)th$ location is

$$\psi_{m,n} = \frac{2\pi d_x}{\lambda} \sin\theta \left[\left(m + \frac{n}{2} \right) \cos\phi + n \frac{\sqrt{3}}{2} \sin\phi \right] \tag{8.72}$$

MATLAB Function "rect_array.m"

The function *"rect_array.m"* computes and plots the rectangular antenna gain pattern in the visible *U,V* space. This function is given in Listing 8.5 in Section 8.8. 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 shown in Fig. 8.37. The user is advised to use this MATLAB GUI[1] workspace to generate array gain patterns that match this requirement.

Fig.s 8.38 through 8.43 respectively show plots of the array gain pattern in the *U-V* space, for the following cases:

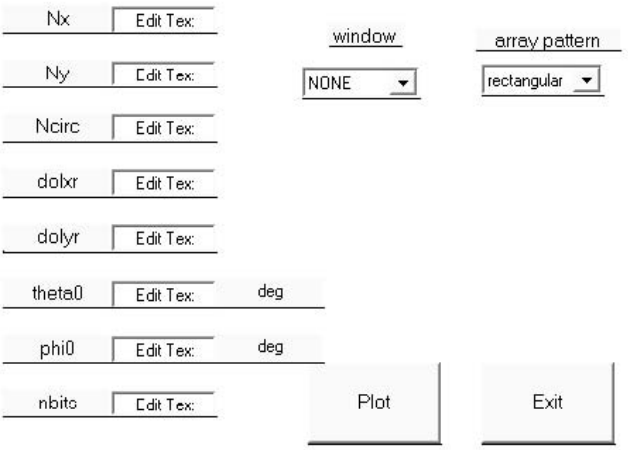

Figure 8.37. MATLAB GUI workspace *"array.m."*

$$[pattern] = rect\_array(15, 15, 0.5, 0.5, 0, 0, -1, -1, -3) \quad \textbf{(8.73)}$$

$$[pattern] = rect\_array(15, 15, 0.5, 0.5, 20, 30, -1, -1, -3) \quad \textbf{(8.74)}$$

$$[pattern] = rect\_array(15, 15, 0.5, 0.5, 30, 30, 1, \text{'Hamming'}, -3) \quad \textbf{(8.75)}$$

$$[pattern] = rect\_array(15, 15, 0.5, 0.5, 30, 30, -1, -1, 3) \quad \textbf{(8.76)}$$

$$[pattern] = rect\_array(15, 15, 1, 0.5, 10, 30, -1, -1, -3) \quad \textbf{(8.77)}$$

$$[pattern] = rect\_array(15, 15, 1, 1, 0, 0, -1, -1, -3) \quad \textbf{(8.78)}$$

1. This GUI was developed by Mr. David J. Hall, Consultant to Decibel Research, Inc., Huntsville, Alabama.

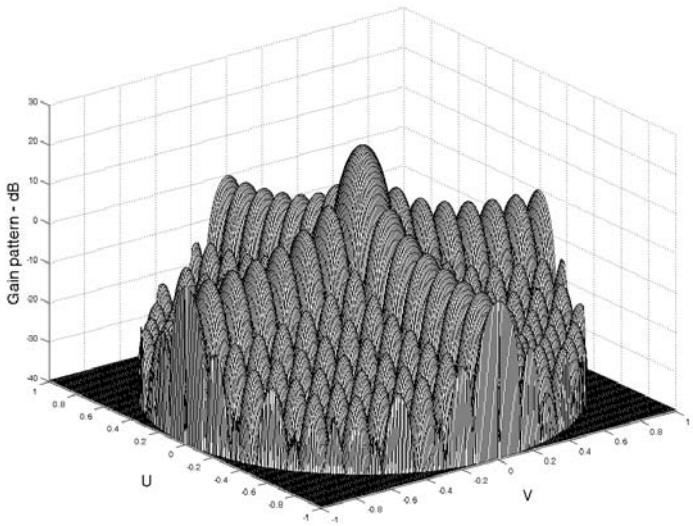

Figure 8.38a. 3-D gain pattern corresponding to Eq. (8.73).

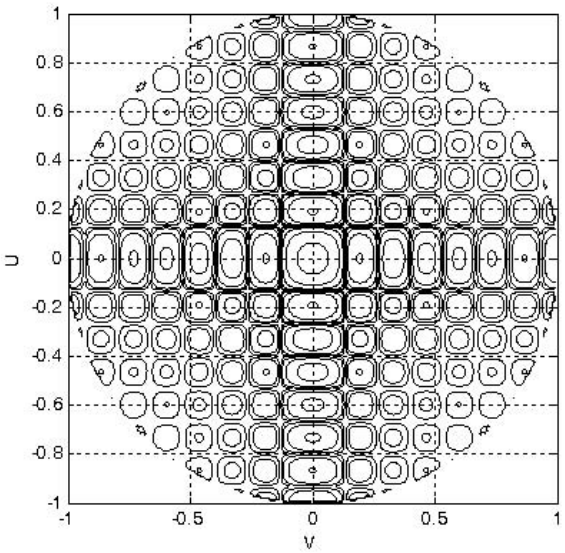

Figure 8.38b. Contour plot corresponding to Eq. (8.73).

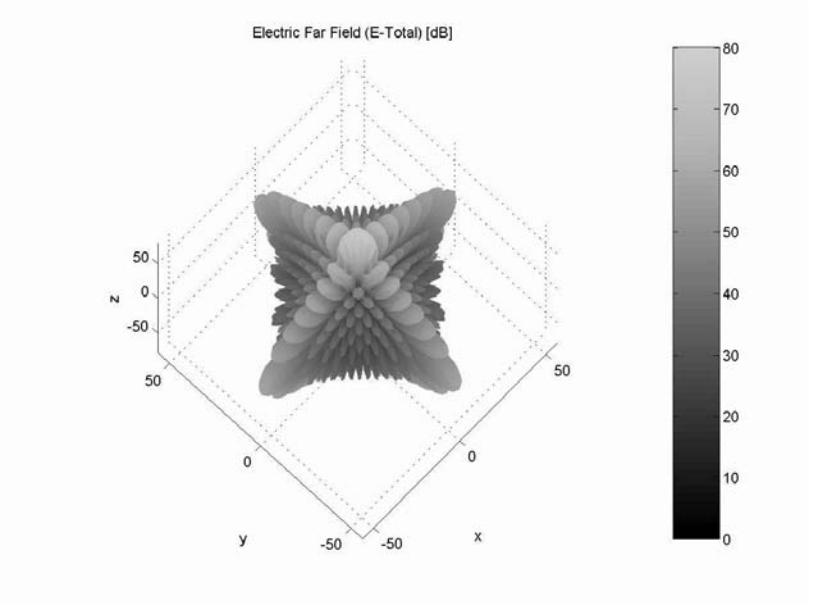

Figure 8.38c. Three-dimensional plot (θ, ϕ space) corresponding to Eq. (8.73).

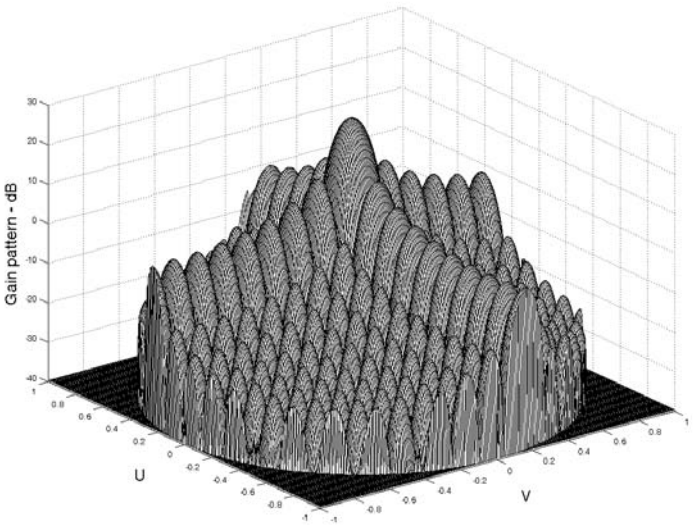

Figure 8.39a. 3-D gain pattern corresponding to Eq. (8.74).

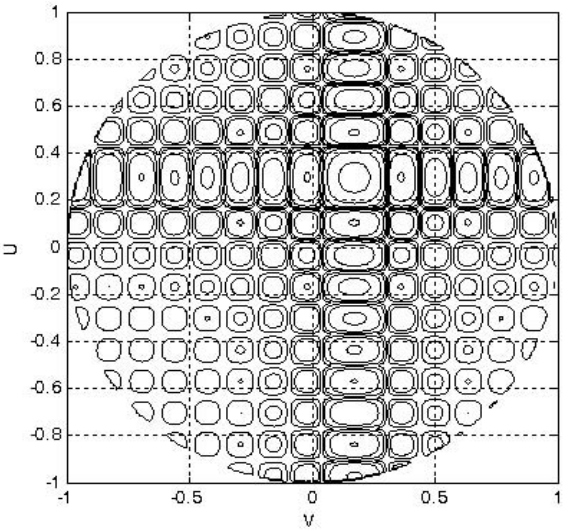

Figure 8.39b. Contour plot corresponding to Eq. (8.74).

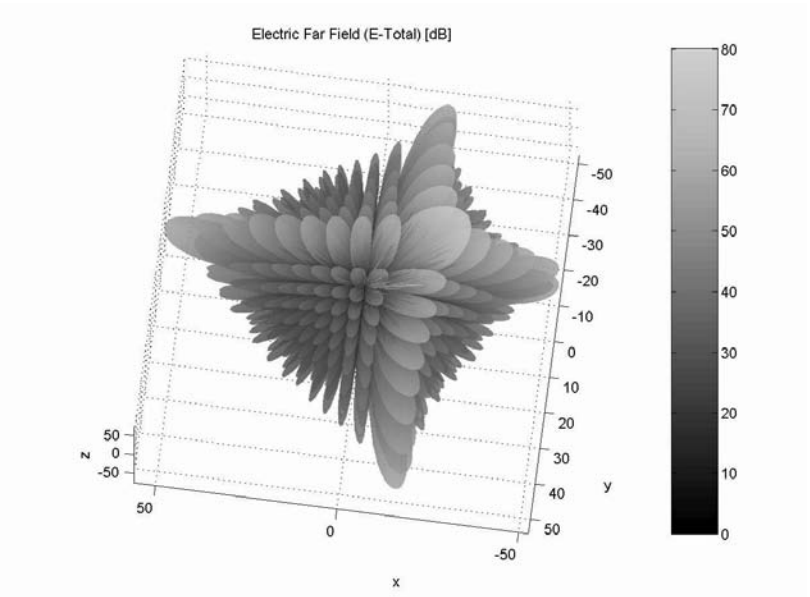

Figure 8.39c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.74).

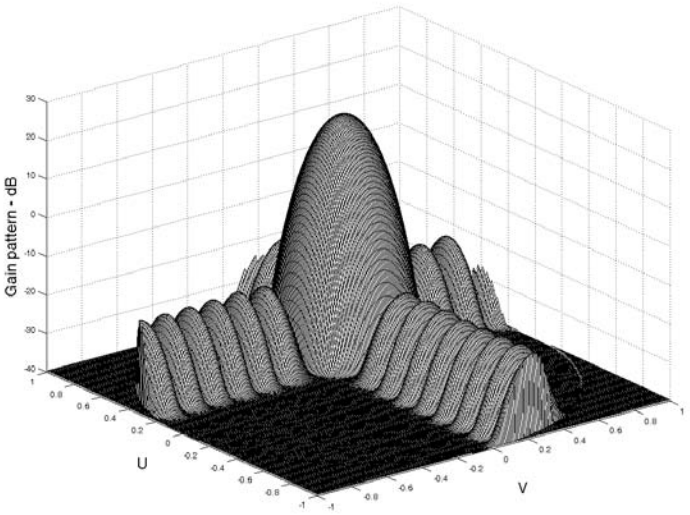

Figure 8.40a. 3-D gain pattern corresponding to Eq. (8.75).

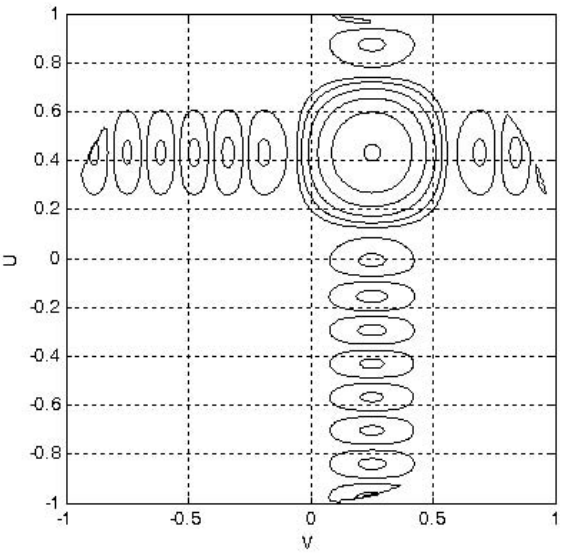

Figure 8.40b. Contour plot corresponding to Eq. (8.75).

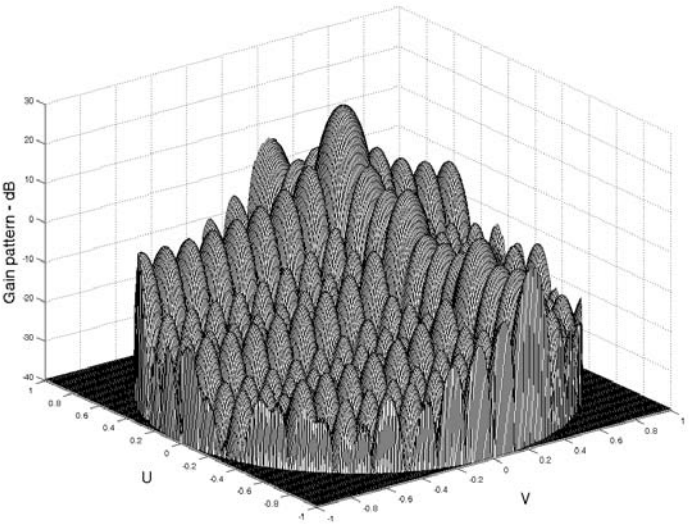

Figure 8.41a. 3-D gain pattern corresponding to Eq. (8.76).

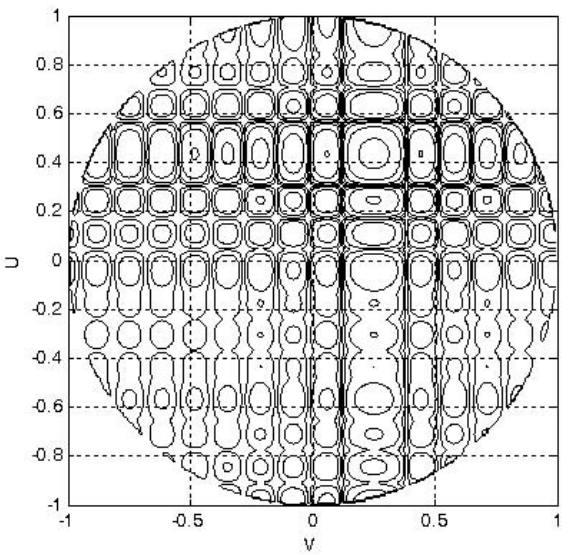

Figure 8.41b. Contour plot corresponding to Eq. (8.76).

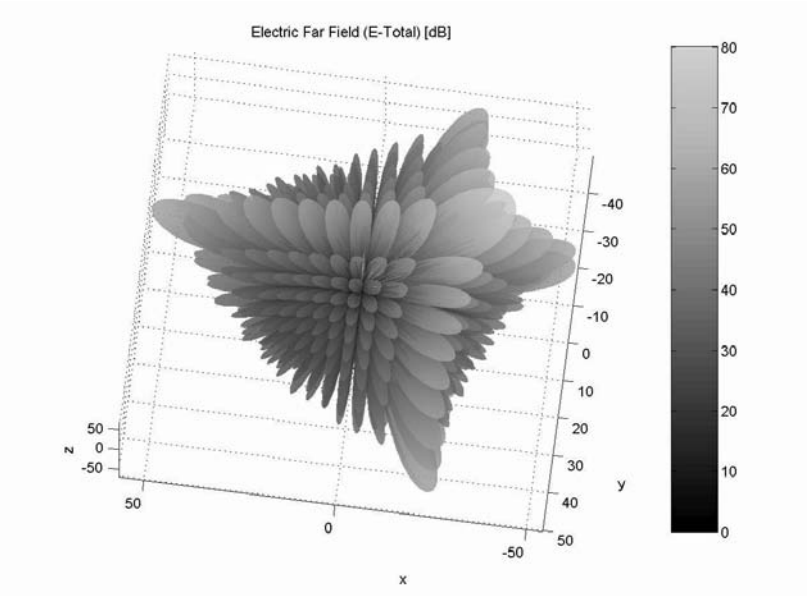

Figure 8.41c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.76).

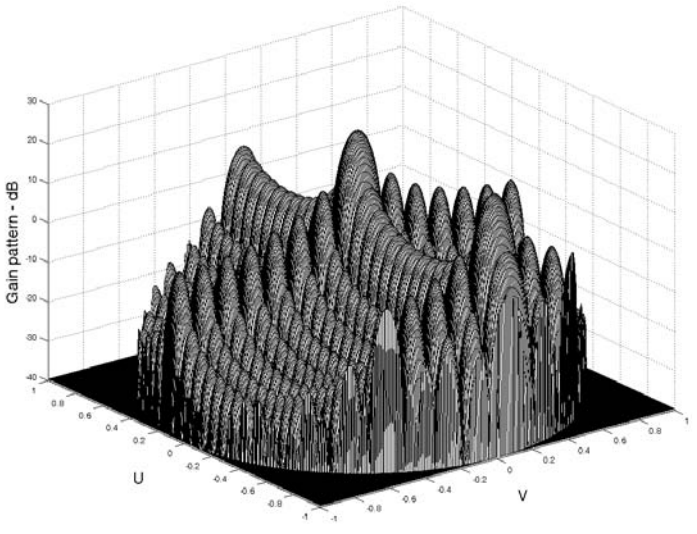

Figure 8.42a. 3-D gain pattern corresponding to Eq. (8.77).

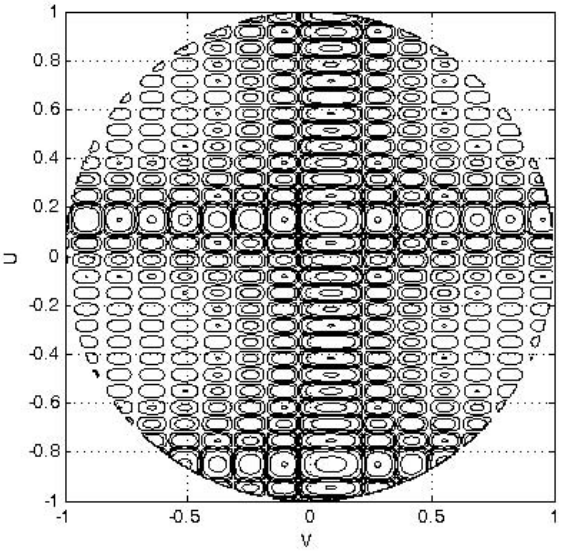

Figure 8.42b. Contour plot corresponding to Eq. (8.77).

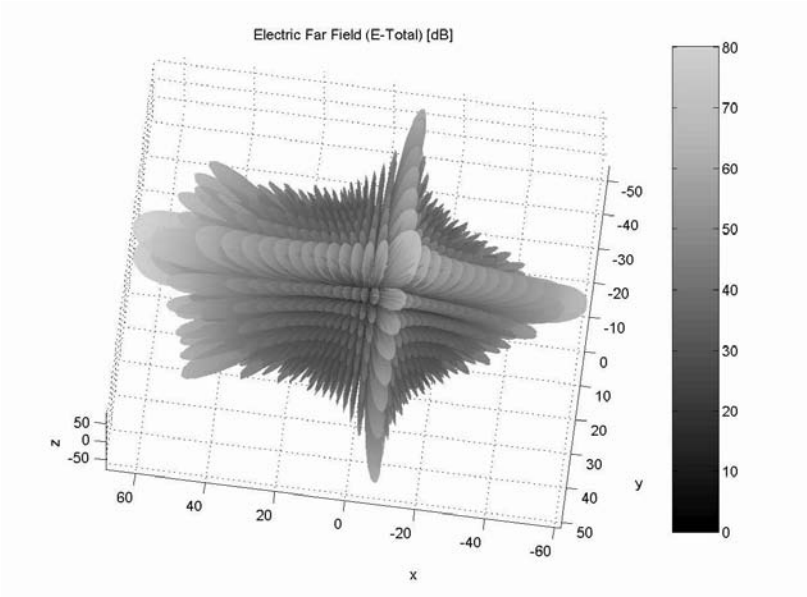

Figure 8.42c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.77).

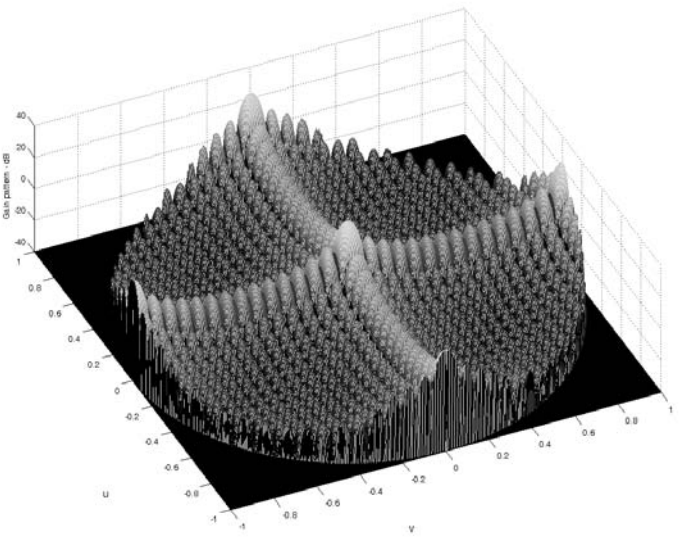

Figure 8.43a. 3-D gain pattern corresponding to Eq. (8.78).

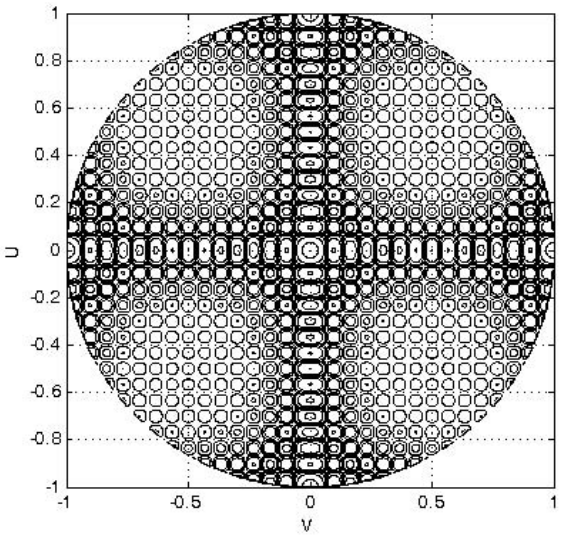

Figure 8.43b. Contour plot corresponding to Eq. (8.78).

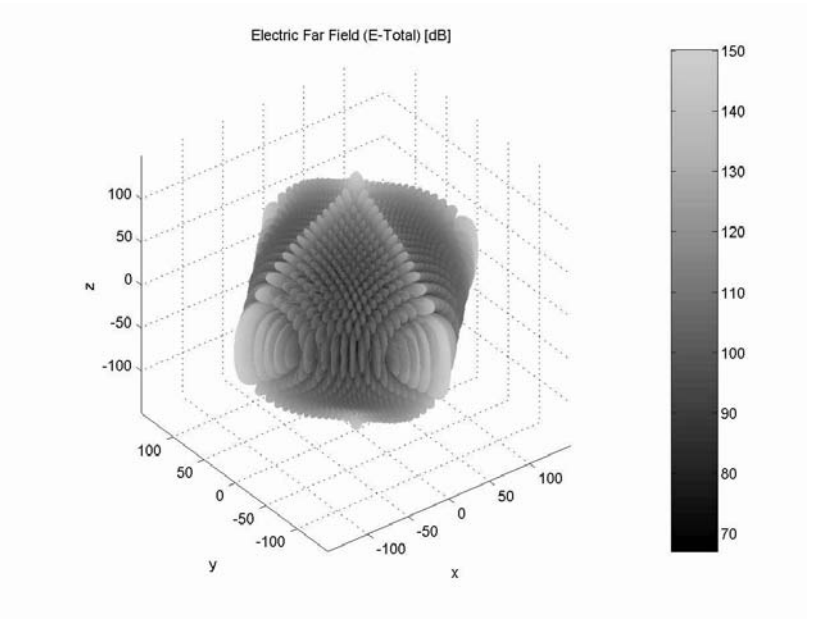

Figure 8.43c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.78).

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. This function is given in Listing 8.6 in Section 8.8. 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*

Symbol	Description	Units	Status
win	*window for sidelobe control*	*none*	*input*
nbits	*negative #: perfect quantization* *positive #: use* 2^{nbits} *quantization levels*	*none*	*input*
patterng	*gain pattern*	*dB*	*output*
amn	*a(m,n) sequence defined in Eq. (8.68)*	*none*	*output*

Figs. 8.44 through 8.49 respectively show plots of the array gain pattern versus steering for the following cases:

[pattern, amn] = circ_array(15, 0.5, 0.5, 0, 0, -1, -1, -3) **(8.79)**

[pattern, amn] = circ_array(15, 0.5, 0.5, 20, 30, -1, -1, -3) **(8.80)**

[pattern, amn] = circ_array(15, 0.5, 0.5, 30, 30, 1, 'Hamming', -3) **(8.81)**

[pattern, amn] = circ_array(15, 0.5, 0.5, 30, 30, -1, -1, 3) **(8.82)**

[pattern, amn] = circ_array(15, 1, 0.5, 10, 30, -1, -1, -3) **(8.83)**

[pattern, amn] = circ_array(15, 1, 1, 0, 0, -1, -1, -3) **(8.84)**

Note the function *"circ_array.m"* uses the function *"rec_to_circ.m",* which computes the array $a(m,n)$. It is given in Listing 8.7 in Section 8.8.

The MATLAB GUI workspace defined in *"array.m"* can be used to execute this function.

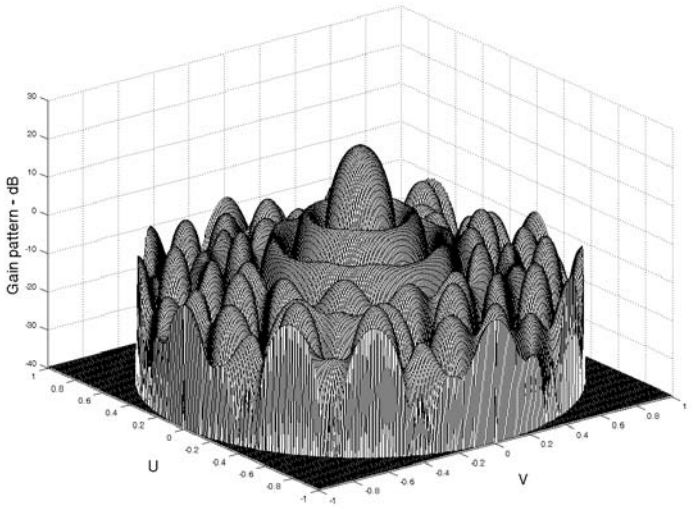

Figure 8.44a. 3-D gain pattern corresponding to Eq. (8.79).

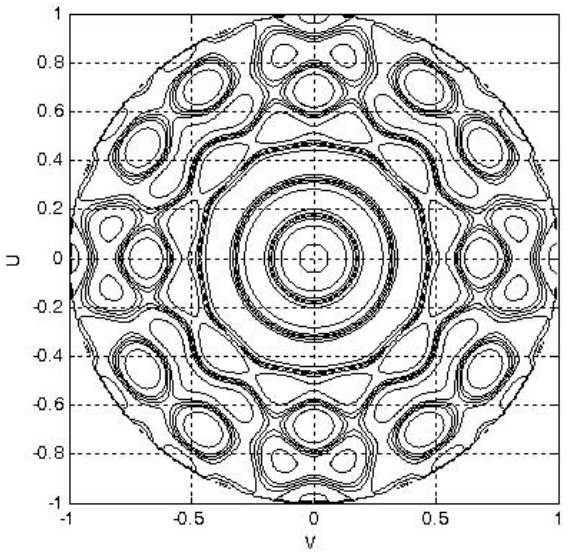

Figure 8.44b. Contour plot corresponding to Eq. (8.79).

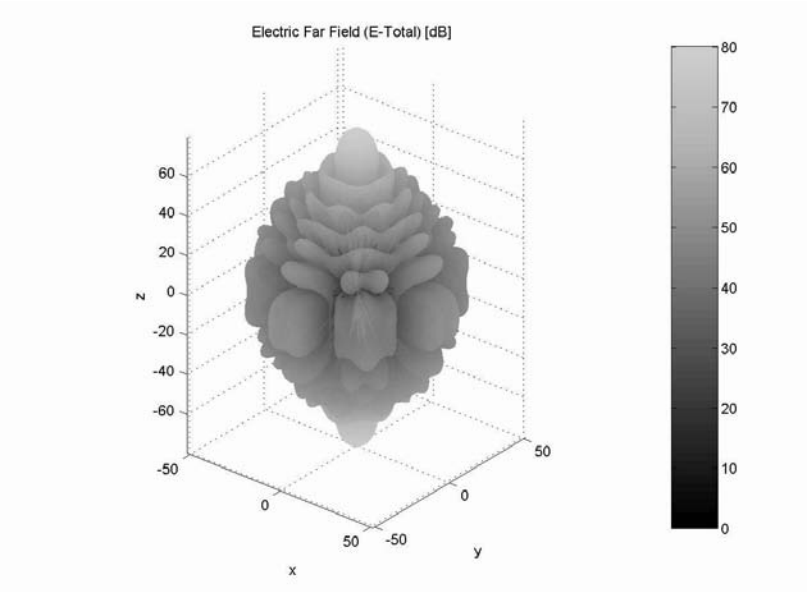

Figure 8.44c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.79).

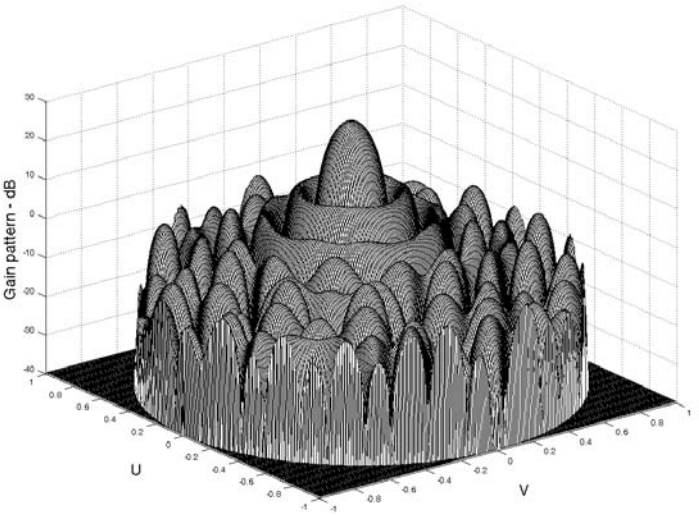

Figure 8.45a. 3-D gain pattern corresponding to Eq. (8.80).

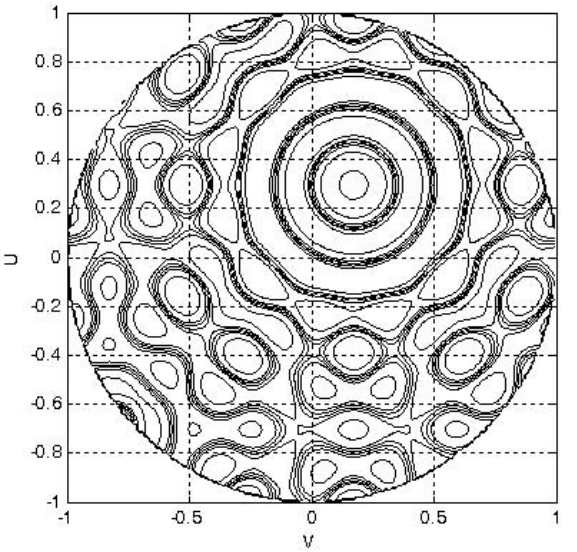

Figure 8.45b. Contour plot corresponding to Eq. (8.80).

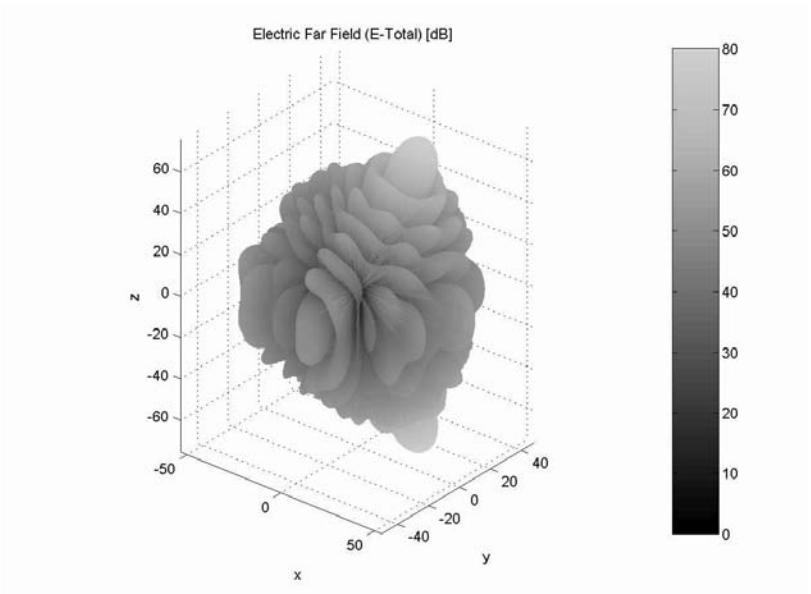

Figure 8.45c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.80).

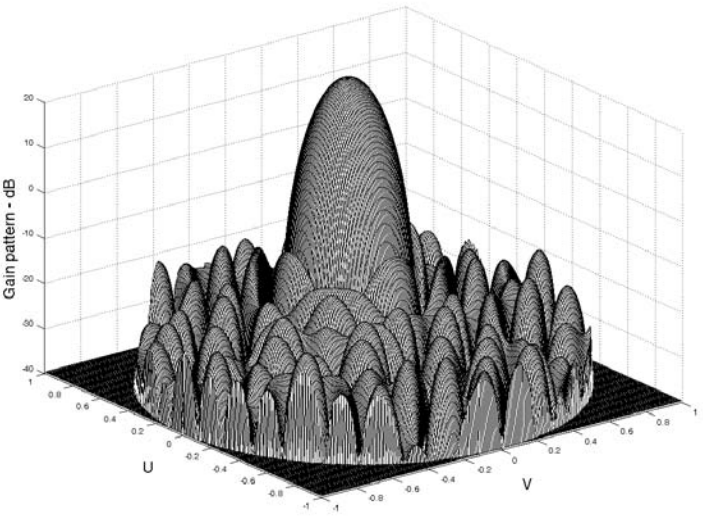

Figure 8.46a. 3-D gain pattern corresponding to Eq. (8.81).

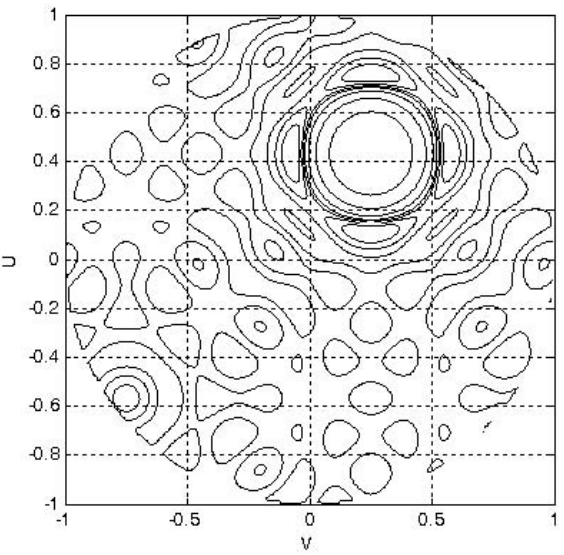

Figure 8.46b. Contour plot corresponding to Eq. (8.81).

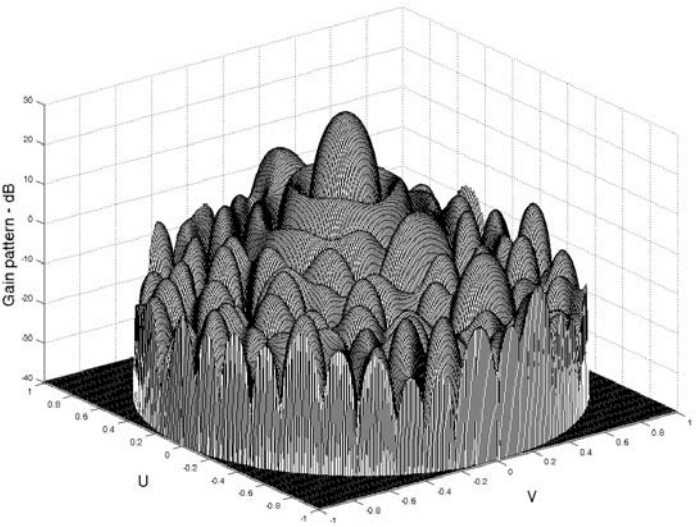

Figure 8.47a. 3-D gain pattern corresponding to Eq. (8.82).

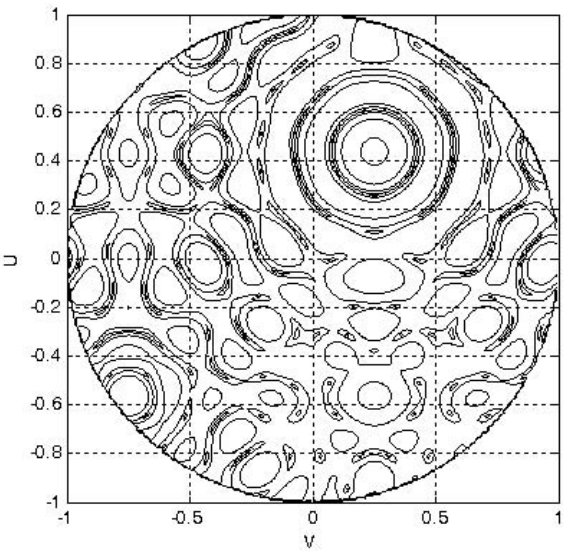

Figure 8.47b. Contour plot corresponding to Eq. (8.82).

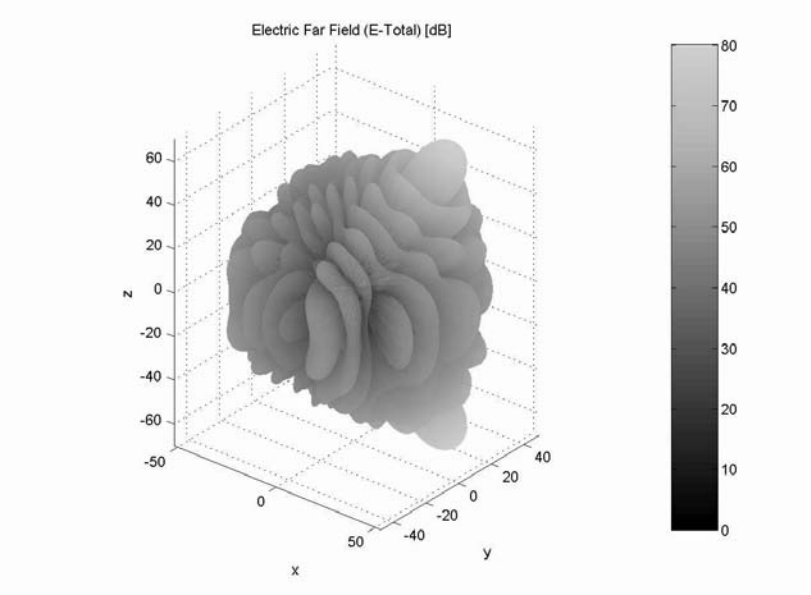

Figure 8.47c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.82).

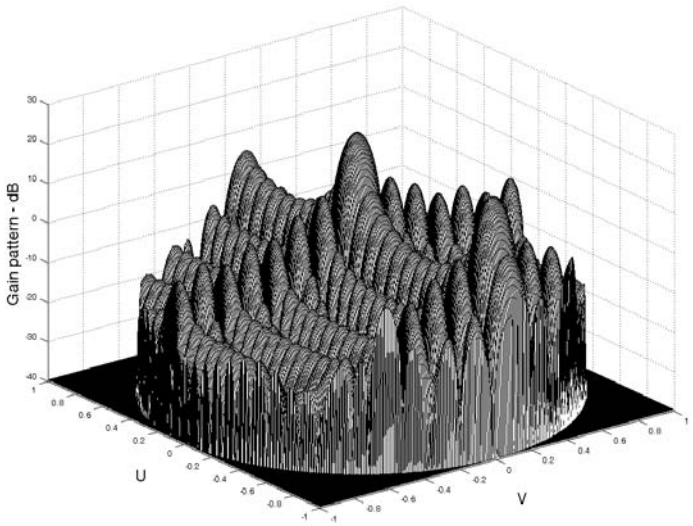

Figure 8.48a. 3-D gain pattern corresponding to Eq. (8.83).

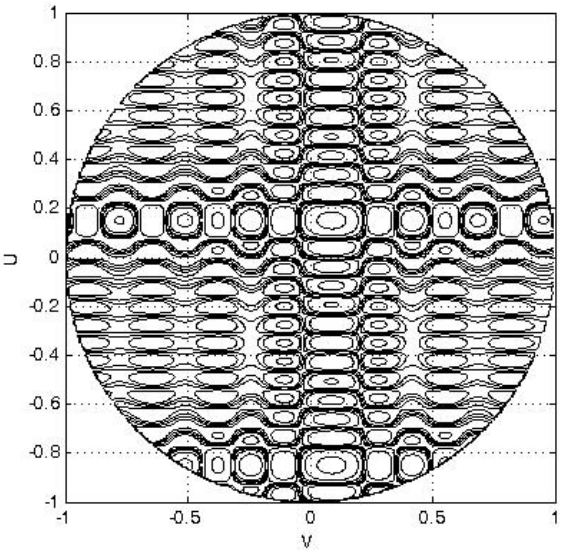

Figure 8.48b. Contour plot corresponding to Eq. (8.83).

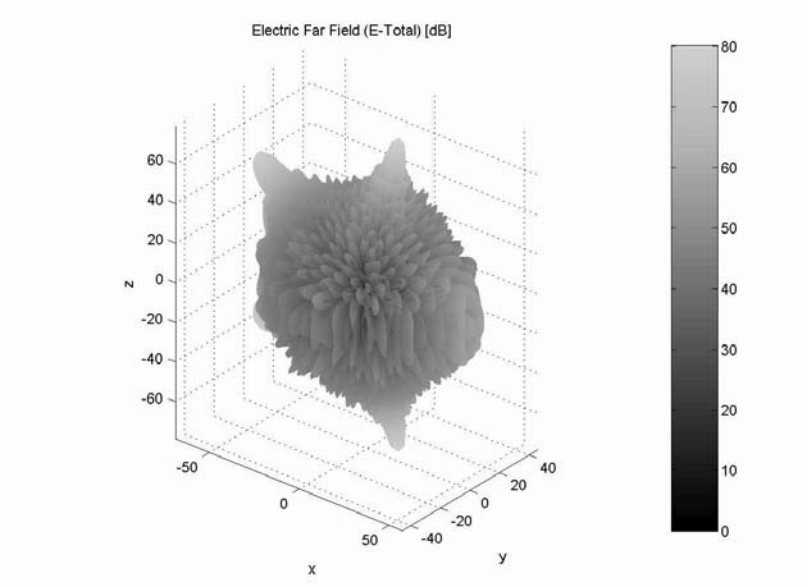

Figure 8.48c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.83).

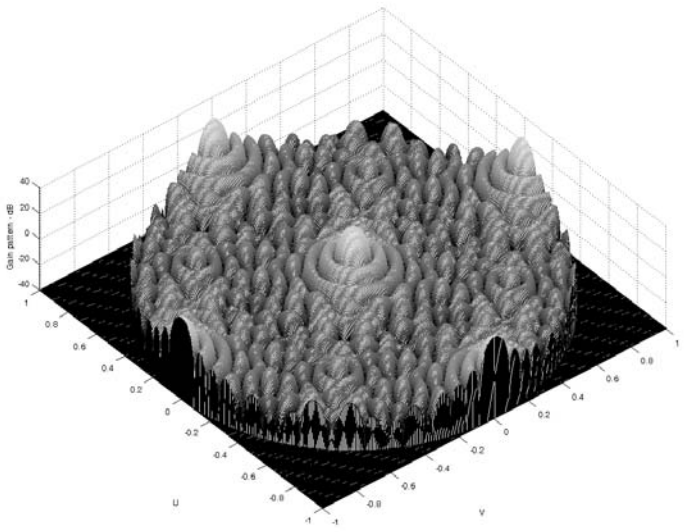

Figure 8.49a. 3-D gain pattern corresponding to Eq. (8.84).

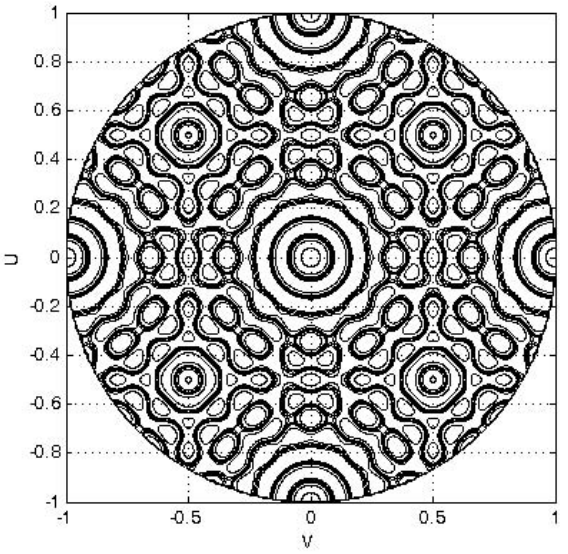

Figure 8.49b. Contour plot corresponding to Eq. (8.84).

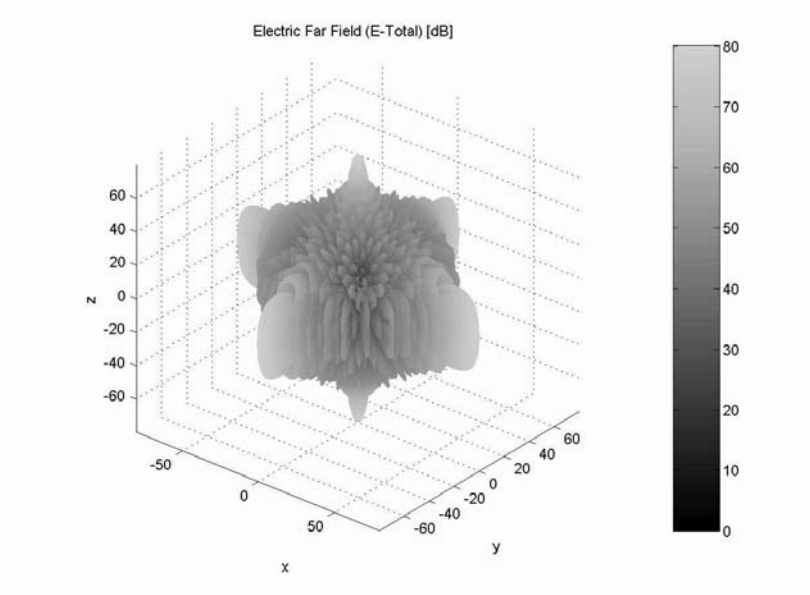

Figure 8.49c. 3-D plot (θ, ϕ space) corresponding to Eq. (8.84).

The program *"array.m"* also plots the array's element spacing pattern. Figs. 8.50a and 8.50b 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, Fig. 8.50a shows a rectangular grid with circular boundary as defined in Eqs. (8.67) and (8.68) with $d_x = d_y = 0.5\lambda$ and $a = 0.35\lambda$. Fig. 8.50b shows a similar configuration except that an element spacing $d_x = 1.5\lambda$ and $d_y = 0.5\lambda$.

8.6. 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. 8.51. This loss in antenna gain is called scan loss, L_{scan}, where

$$L_{scan} = \left(\frac{A}{A_\theta}\right)^2 = \left(\frac{G}{G_\theta}\right)^2 \qquad \textbf{(8.85)}$$

A_θ is effective aperture area at scan angle θ, and G_θ is effective array gain at the same angle.

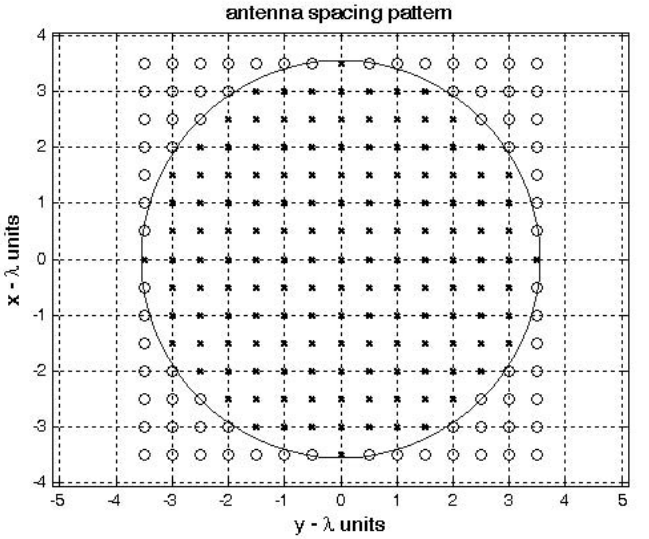

Figure 8.50a. A 15 element circular array made from a rectangular array with circular boundary. Element spacing $d_x = 0.5\lambda = d_y$.

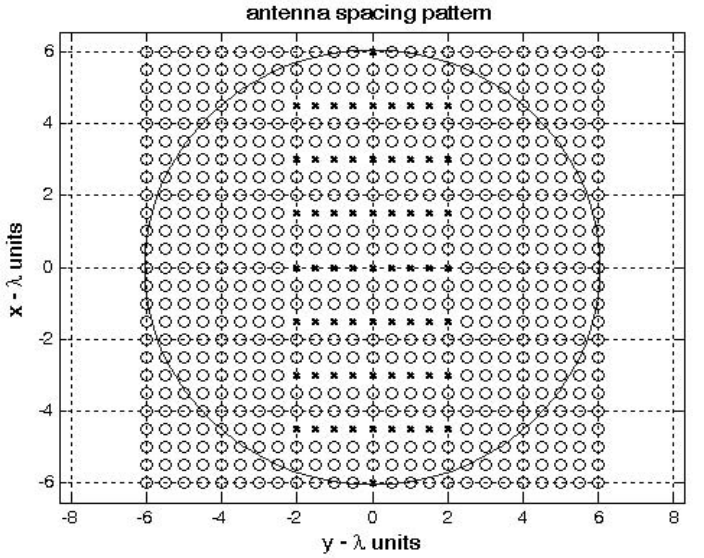

Figure 8.50b. A 15 element circular array made from a rectangular array with circular boundary. Element spacing $d_y = 0.5\lambda$ and $d_x = 1.5\lambda$.

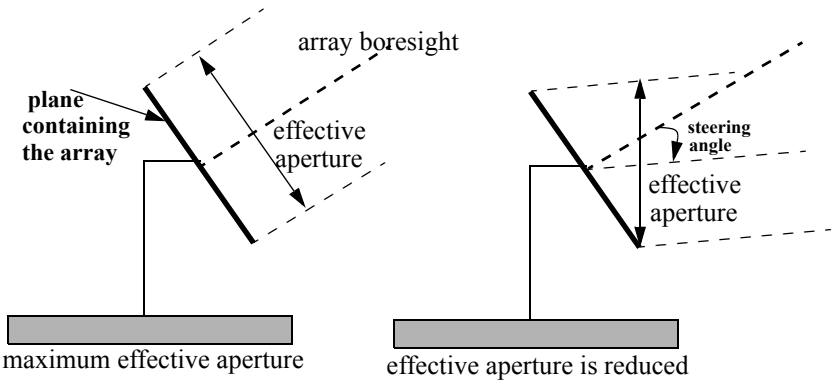

Figure 8.51. Reduction in array effective aperture due to electronic scanning.

The beamwidth at scan angle θ is

$$\Theta_\theta = \frac{\Theta_{broadside}}{\cos\theta} \tag{8.86}$$

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). FFOV arrays employ element spacing of 0.6λ or less to avoid grating lobes. FFOV array scan loss is approximated by

$$L_{scan} \approx (\cos\theta)^{2.5} \tag{8.87}$$

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

$$L_{scan} = \left[\frac{\sin\left(\frac{\pi d}{\lambda}\sin\theta\right)}{\frac{\pi d}{\lambda}\sin\theta}\right]^{-4} \tag{8.88}$$

Fig. 8.52 shows a plot for scan loss versus scan angle. This figure can be reproduced using MATLAB program *"fig8_52.m"* given in Listing 8.8.

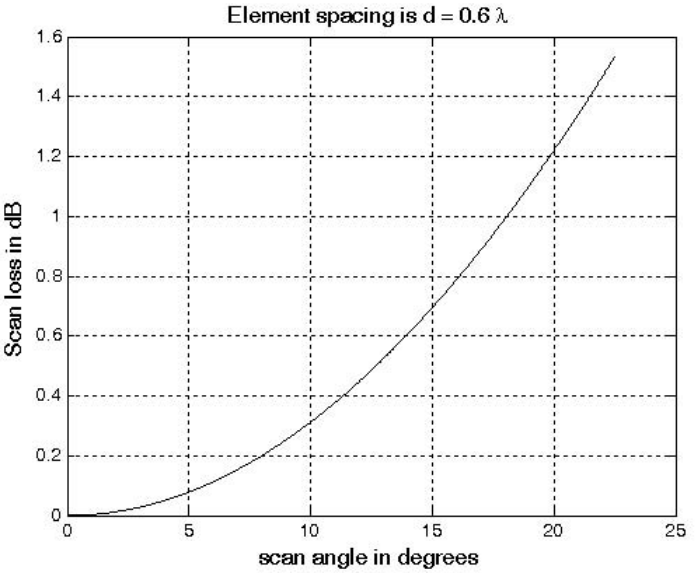

Figure 8.52. Scan loss versus scan angle, based on Eq. (8.87).

8.7. "MyRadar" Design Case Study - Visit 8

8.7.1. Problem Statement

Modify the "MyRadar" design case study such that we employ a phased array antenna. For this purpose, modify the design requirements such that the search volume is now defined by $\Theta_e = 10^\circ$ *and* $\Theta_a \leq 45^\circ$. *Assume X-band, if possible. Design an electronically steered radar (ESR). Non-coherent integration of a few pulses may be used, if necessary. Size the radar so that it can fulfill this mission. Calculate the antenna gain, aperture size, missile and aircraft detection range, number of elements in the array, etc. All other design requirements are as defined in the previous chapters.*

8.7.2. A Design

The search volume is

$$\Omega = \frac{10^\circ \times 45^\circ}{(57.296)^2} = 0.1371 \ \ steradian \tag{8.89}$$

For an X-band radar, choose $f_o = 9GHz$, then

$$\lambda = \frac{3 \times 10^8}{9 \times 10^9} = 0.0333m \tag{8.90}$$

Assume an aperture size $A_e = 2.25m^2$; thus

$$G = \frac{4\pi A_e}{\lambda^2} = \frac{4 \times \pi \times 2.25}{(0.0333)^2} = 25451.991 \Rightarrow G = 44dB \tag{8.91}$$

Assume square aperture. It follows that the aperture 3-dB beamwidth is calculated from

$$\left(G = \frac{4\pi}{\theta_{3db}^2}\right) \Rightarrow \theta_{3dB} = \sqrt{\frac{4 \times \pi \times 180^2}{25451.991 \times \pi^2}} = 1.3° \tag{8.92}$$

The number of beams required to fill the search volume is

$$n_b = k_p \frac{\Omega}{(1.3/57.296)^2}\bigg|_{k_p = 1.5} \Rightarrow n_b = 399.5 \Rightarrow choose \;\; n_b = 400 \tag{8.93}$$

Note that the packing factor k_p is used to allow for beam overlap in order to avoid gaps in the beam coverage. The search scan rate is 2 seconds. Thus, the minimum PRF should correspond to 200 beams per second (i.e., $f_r = 200Hz$). This PRF will allow the radar to visit each beam position only once during a complete scan.

It was determined in Chapter 2 that 4-pulse non-coherent integration along with a cumulative detection scheme are required to achieve the desired probability of detection. It was also determined that the single pulse energy for the missile and aircraft cases are respectively given by (see page 118)

$$E_m = 0.1147Joules \tag{8.94}$$

$$E_a = 0.1029Joules \tag{8.95}$$

However, these values were derived using $\lambda = 0.1m$ and $G = 2827.4$. The new wavelength is $\lambda = 0.0333m$ and the new gain is $G = 25451.99$. Thus, the missile and aircraft single pulse energy, assuming the same single pulse SNR as derived in Chapter 2 (i.e., $SNR = 4dB$) are

$$E_m = 0.1147 \times \frac{0.1^2 \times 2827.4^2}{0.0333^2 \times 25452^2} = 0.012765Joules \tag{8.96}$$

$$E_a = 0.1029 \times \frac{0.1^2 \times 2827.4^2}{0.0333^2 \times 25452^2} = 0.01145 Joules \quad \textbf{(8.97)}$$

The single pulse peak power that will satisfy detection for both target types is

$$P_t = \frac{0.012765}{20 \times 10^{-6}} = 638.25 W \quad \textbf{(8.98)}$$

where $\tau = 20 \mu s$ is used.

Note that since a 4-pulse non-coherent integration is adopted, the minimum PRF is increased to

$$f_r = 200 \times 4 = 800 Hz \quad \textbf{(8.99)}$$

and the total number of beams is $n_b = 1600$. Consequently the unambiguous range is

$$R_u \leq \frac{3 \times 10^8}{2 \times 800} = 187.5 Km \quad \textbf{(8.100)}$$

(1.101)

Since the effective aperture is $A_e = 2.25 m^2$, then by assuming an array efficiency $\rho = 0.8$ the actual array size is

$$A = \frac{2.25}{0.8} = 2.8125 m^2 \quad \textbf{(8.102)}$$

It follows that the physical array sides are $1.68 m \times 1.68 m$. Thus, by selecting the array element spacing $d = 0.6\lambda$ an array of size 84×84 elements satisfies the design requirements.

Since the field of view is less than $\pm 22.5°$, one can use element spacing as large as $d = 1.5\lambda$ without introducing any grating lobes into the array FOV. Using this option yields an array of size $34 \times 34 = 1156$ elements. Hence, the required power per element is less than $0.6 W$.

8.8. MATLAB Program and Function Listings

This section contains listings of all MATLAB programs and functions used in this chapter. Users are encouraged to rerun this code with different inputs in order to enhance their understanding of the theory.

Listing 8.1. MATLAB Program "fig8_5.m"

```
% Use this code to produce figure 8.5a and 8.5b based on equation 8.34
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)
xlabel('sine angle - dimensionless')
ylabel('Array pattern')
grid

figure(2)
plot(var,20*log10(pattern))
axis ([-1 1 -60 0])
xlabel('sine angle - dimensionless')
ylabel('Power pattern [dB]')
grid;

figure(3)
theta = theta +pi/2;
polar(theta,pattern)
title ('Array pattern')

figure(4)
```

```
polardb(theta,pattern)
title ('Power pattern [dB]')
```

Listing 8.2. MATLAB Program "fig8_7.m"

```
% Use this program to reproduce Fig. 8.7 of text
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, 512)))./32 +eps);
HAM =  20*log10(abs(fftshift(fft(ham, 512)))./32 +eps);
HAN =  20*log10(abs(fftshift(fft(han, 512)))./32+eps);
BLK = 20*log10(abs(fftshift(fft(blk, 512)))./32+eps);
K6 = 20*log10(abs(fftshift(fft(k6, 512)))./32+eps);
x = linspace(-1,1,512);
figure
subplot(2,1,1)
plot(x,RECT,'k--',x,HAM,'k',x,HAN,'k-.');
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-.')
xlabel('x')
ylabel('Window')
legend('Rectangular','Blackman','Kasier at \beta = 6')
grid
axis tight
```

Listing 8.3. MATLAB Function "linear_array.m"

```
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 compute 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 phase 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);
% determine and compute 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 quantization level at
each array element
   angleq = round(dolr .* n * sintheta0 * levels) .* qlevels; % vector of possi-
ble 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 that corresponds 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 radiationa 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
```

Listing 8.4. MATLAB Program "circular_array.m"

```
% 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
% Last modified: July 13, 2003
```

```
%
%%%% 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 = 20;          % 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)
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)
%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]')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function polardb(theta,rho,line_style)
%   POLARDB  Polar coordinate plot.
%   POLARDB(THETA, RHO) makes a plot using polar coordinates of
%   the angle THETA, in radians, versus the radius RHO in dB.
%   The maximum value of RHO should not exceed 1. It should not be
%   normalized, however (i.e., its max. value may be less than 1).
%   POLAR(THETA,RHO,S) uses the linestyle specified in string S.
%   See PLOT for a description of legal linestyles.
if nargin < 1
   error('Requires 2 or 3 input arguments.')
elseif nargin == 2
```

```
    if isstr(rho)
       line_style = rho;
       rho = theta;
       [mr,nr] = size(rho);
       if mr == 1
          theta = 1:nr;
       else
          th = (1:mr)';
          theta = th(:,ones(1,nr));
       end
    else
       line_style = 'auto';
    end
elseif nargin == 1
    line_style = 'auto';
    rho = theta;
    [mr,nr] = size(rho);
    if mr == 1
       theta = 1:nr;
    else
       th = (1:mr)';
       theta = th(:,ones(1,nr));
    end
end
if isstr(theta) | isstr(rho)
    error('Input arguments must be numeric.');
end
if ~isequal(size(theta),size(rho))
    error('THETA and RHO must be the same size.');
end
% get hold state
cax = newplot;
next = lower(get(cax,'NextPlot'));
hold_state = ishold;

% get x-axis text color so grid is in same color
tc = get(cax,'xcolor');
ls = get(cax,'gridlinestyle');
% Hold on to current Text defaults, reset them to the
% Axes' font attributes so tick marks use them.
fAngle  = get(cax, 'DefaultTextFontAngle');
fName   = get(cax, 'DefaultTextFontName');
fSize   = get(cax, 'DefaultTextFontSize');
fWeight = get(cax, 'DefaultTextFontWeight');
```

```
fUnits  = get(cax, 'DefaultTextUnits');
set(cax, 'DefaultTextFontAngle',  get(cax, 'FontAngle'), ...
   'DefaultTextFontName',   get(cax, 'FontName'), ...
   'DefaultTextFontSize',   get(cax, 'FontSize'), ...
   'DefaultTextFontWeight', get(cax, 'FontWeight'), ...
   'DefaultTextUnits','data')
% make a radial grid
   hold on;
   maxrho =1;
   hhh=plot([-maxrho -maxrho maxrho maxrho],[-maxrho maxrho maxrho -
maxrho]);
   set(gca,'dataaspectratio',[1 1 1],'plotboxaspectratiomode','auto')
   v = [get(cax,'xlim') get(cax,'ylim')];
   ticks = sum(get(cax,'ytick')>=0);
   delete(hhh);
% check radial limits and ticks
   rmin = 0; rmax = v(4); rticks = max(ticks-1,2);
   if rticks > 5   % see if we can reduce the number
      if rem(rticks,2) == 0
         rticks = rticks/2;
      elseif rem(rticks,3) == 0
         rticks = rticks/3;
      end
   end
% only do grids if hold is off
if ~hold_state
% define a circle
   th = 0:pi/50:2*pi;
   xunit = cos(th);
   yunit = sin(th);
% now really force points on x/y axes to lie on them exactly
   inds = 1:(length(th)-1)/4:length(th);
   xunit(inds(2:2:4)) = zeros(2,1);
   yunit(inds(1:2:5)) = zeros(3,1);
% plot background if necessary
   if ~isstr(get(cax,'color')),
     patch('xdata',xunit*rmax,'ydata',yunit*rmax, ...
         'edgecolor',tc,'facecolor',get(gca,'color'),...
         'handlevisibility','off');
   end
% draw radial circles with dB ticks
   c82 = cos(82*pi/180);
   s82 = sin(82*pi/180);
   rinc = (rmax-rmin)/rticks;
```

```
   tickdB=-10*(rticks-1);    % the innermost tick dB value
   for i=(rmin+rinc):rinc:rmax
      hhh = plot(xunit*i,yunit*i,ls,'color',tc,'linewidth',1,...
                 'handlevisibility','off');
      text((i+rinc/20)*c82*0,-(i+rinc/20)*s82, ...
         ['  ' num2str(tickdB) ' dB'],'verticalalignment','bottom',...
         'handlevisibility','off')
      tickdB=tickdB+10;
   end
   set(hhh,'linestyle','-') % Make outer circle solid
% plot spokes
   th = (1:6)*2*pi/12;
   cst = cos(th); snt = sin(th);
   cs = [-cst; cst];
   sn = [-snt; snt];
   plot(rmax*cs,rmax*sn,ls,'color',tc,'linewidth',1,...
      'handlevisibility','off')
% annotate spokes in degrees
   rt = 1.1*rmax;
   for i = 1:length(th)
      text(rt*cst(i),rt*snt(i),int2str(i*30),...
         'horizontalalignment','center',...
         'handlevisibility','off');
      if i == length(th)
         loc = int2str(0);
      else
         loc = int2str(180+i*30);
      end
      text(-rt*cst(i),-rt*snt(i),loc,'horizontalalignment','center',...
         'handlevisibility','off')
   end
% set view to 2-D
   view(2);
% set axis limits
   axis(rmax*[-1 1 -1.15 1.15]);
end
% Reset defaults.
set(cax, 'DefaultTextFontAngle', fAngle , ...
   'DefaultTextFontName',   fName , ...
   'DefaultTextFontSize',   fSize, ...
   'DefaultTextFontWeight', fWeight, ...
   'DefaultTextUnits', fUnits );
% Transfrom data to dB scale
rmin = 0; rmax=1;
```

```
rinc = (rmax-rmin)/rticks;
rhodb=zeros(1,length(rho));
for i=1:length(rho)
   if rho(i)==0
      rhodb(i)=0;
   else
      rhodb(i)=rmax+2*log10(rho(i))*rinc;
   end
   if rhodb(i)<=0
      rhodb(i)=0;
   end
end
% transform data to Cartesian coordinates.
xx = rhodb.*cos(theta);
yy = rhodb.*sin(theta);
% plot data on top of grid
if strcmp(line_style,'auto')
   q = plot(xx,yy);
else
   q = plot(xx,yy,line_style);
end
if nargout > 0
   hpol = q;
end
if ~hold_state
   set(gca,'dataaspectratio',[1 1 1]), axis off; set(cax,'NextPlot',next);
end
set(get(gca,'xlabel'),'visible','on')
set(get(gca,'ylabel'),'visible','on')
```

Listing 8.5. MATLAB Function "rect_array.m"

```
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-axis; Nyr ==> number of elements 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 posi-
tive ==> 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 pos-
sible 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 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)
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
```

Listing 8.6. MATLAB Function "circ_array.m"

```
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 The-
ory 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 posi-
tive ==> 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 indicate 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 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 pos-
sible 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:mid-
point+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 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)
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
```

Listing 8.7. MATLAB Function "rec_to_circ.m"

```
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 + .4);
amn (idx) = 1;
return
```

Listing 8.8. MATLAB Program "fig8_52.m"

```
%Use this program to reproduce Fig. 8.40. Based on Eq. (8.87)
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)
xlabel('scan angle in degrees')
ylabel('Scan loss in dB')
grid
title('Element spacing is d = 0.6 \lambda ')
```

Chapter 9 *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.

9.1. 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.

9.1.1. 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. 9.1.

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. 9.1a, the voltage difference is zero. However, when the target is off the tracking axis, as in Fig. 9.1b, a non-zero 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.

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.

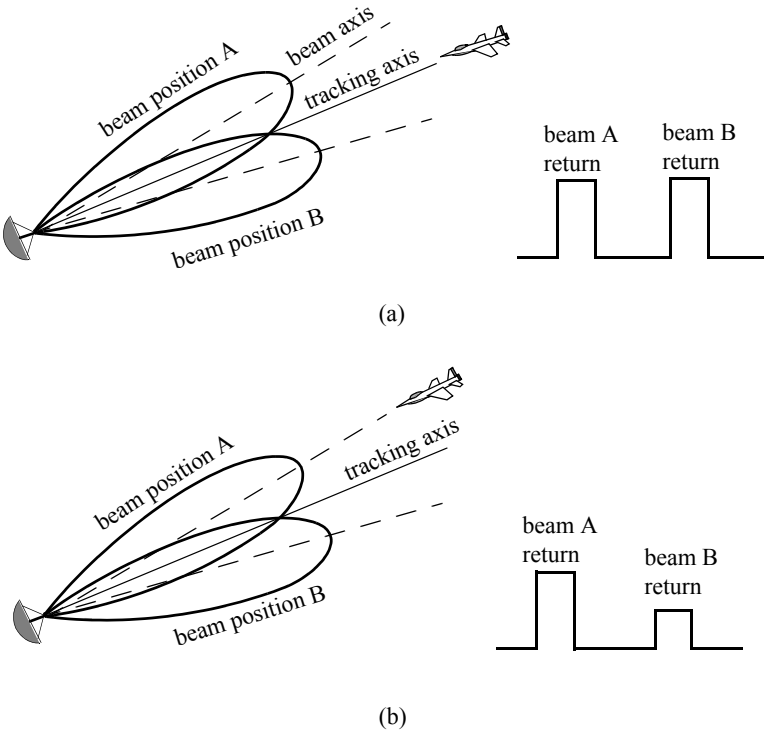

Figure 9.1. Sequential lobing. (a) Target is located on track axis. (b) Target is off track axis.

9.1.2. 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. Fig. 9.2 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.

Fig. 9.3 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. 9.4. 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.

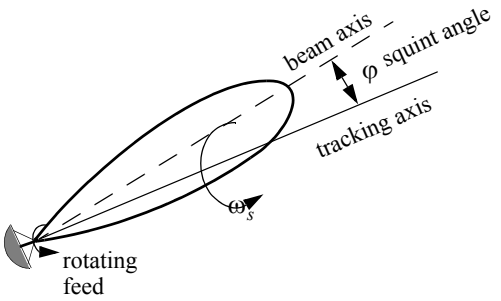

Figure 9.2. Conical scan beam.

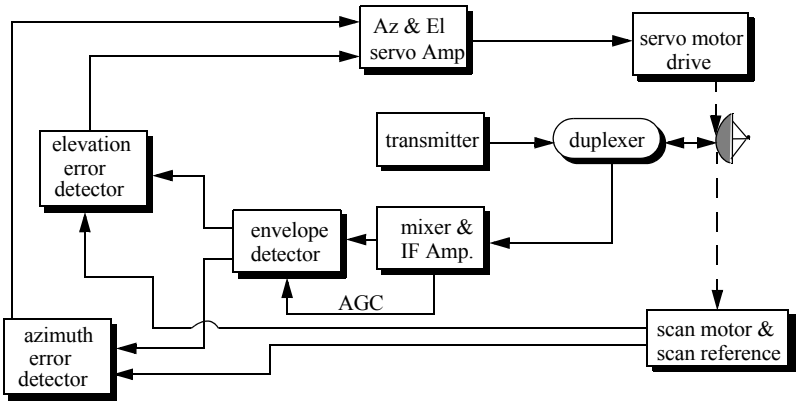

Figure 9.3. Simplified conical scan radar system.

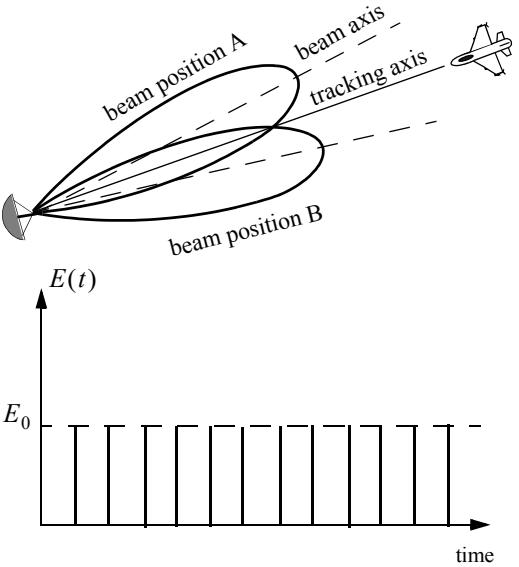

Figure 9.4. Error signal produced when the target is on the tracking axis for conical scan.

Next, consider the case depicted by Fig. 9.5. 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.

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. 9.6. 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

$$\varepsilon_a = \varepsilon \sin\varphi \tag{9.1}$$

$$\varepsilon_e = \varepsilon \cos\varphi \tag{9.2}$$

These are the error signals that the radar uses to align the tracking axis on the target.

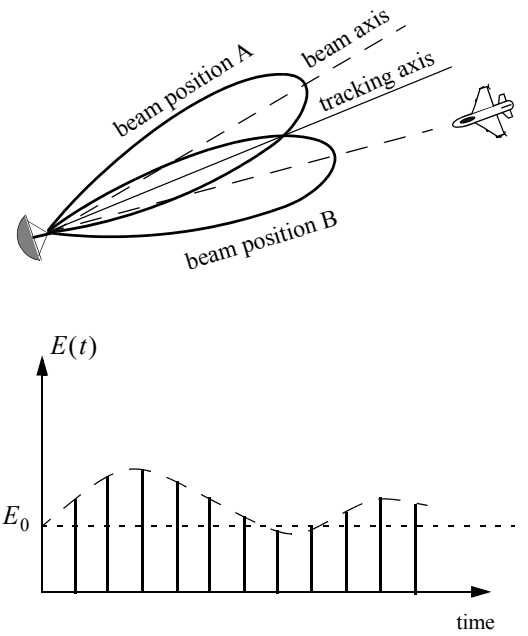

Figure 9.5. Error signal produced when the target is off the tracking axis for conical scan.

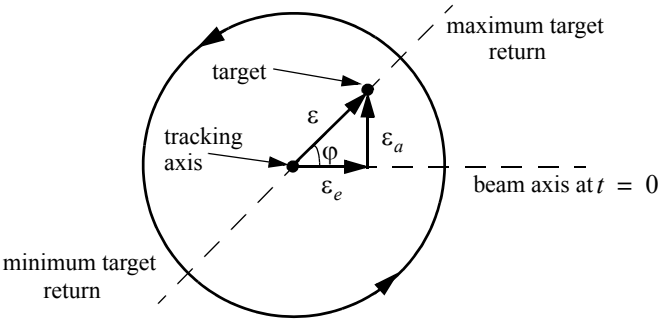

Figure 9.6. Top view of beam axis for a complete scan.

The AM signal $E(t)$ can then be written as

$$E(t) = E_0 \cos(\omega_s t - \varphi) = E_0 \varepsilon_e \cos\omega_s t + E_0 \varepsilon_a \sin\omega_s t \quad \textbf{(9.3)}$$

where E_0 is a constant called the error slope, ω_s is the scan frequency in radians per seconds, 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 low pass filtering. More precisely,

$$E_e(t) = E_0 \cos(\omega_s t - \varphi)\cos\omega_s t = -\frac{1}{2}E_0 \cos\varphi + \frac{1}{2}\cos(2\omega_s t - \varphi) \quad \textbf{(9.4)}$$

and after low pass filtering we get

$$E_e(t) = -\frac{1}{2}E_0 \cos\varphi \quad \textbf{(9.5)}$$

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 low pass filtering. It follows that

$$E_a(t) = \frac{1}{2}E_0 \sin\varphi \quad \textbf{(9.6)}$$

The antenna scan rate is limited by the scanning mechanism (mechanical or electronic), where electronic scanning is much faster and more accurate than mechanical scan. 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 seconds 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.

9.2. 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.

Fig. 9.7 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.

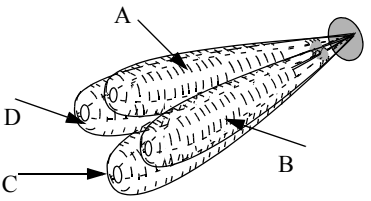

Figure 9.7. Monopulse antenna pattern.

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. 9.8a, 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. 9.8b-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. Fig. 9.9 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.

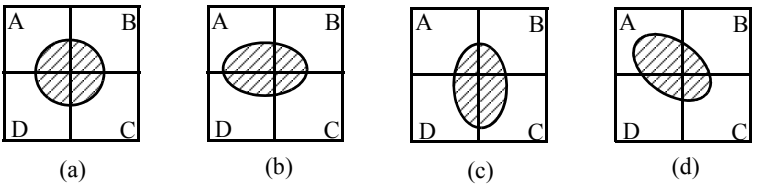

Figure 9.8. Illustration of monopulse concept. (a) Target is on the tracking axis. (b) - (d) Target is off the tracking axis.

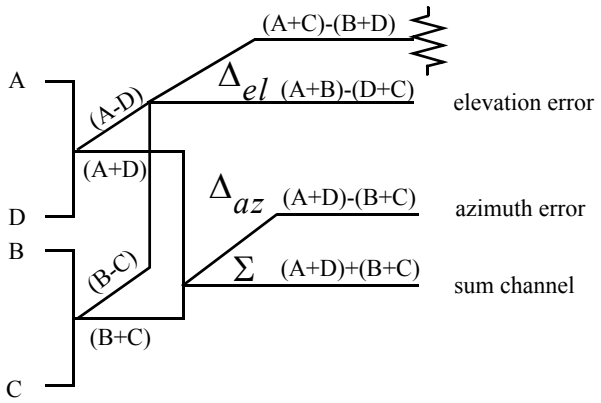

Figure 9.9. Monopulse comparator.

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.

A simplified monopulse radar block diagram is shown in Fig. 9.10. 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

$$\Sigma(\varphi) = \frac{\sin(\varphi - \varphi_0)}{(\varphi - \varphi_0)} + \frac{\sin(\varphi + \varphi_0)}{(\varphi + \varphi_0)} \qquad \textbf{(9.7)}$$

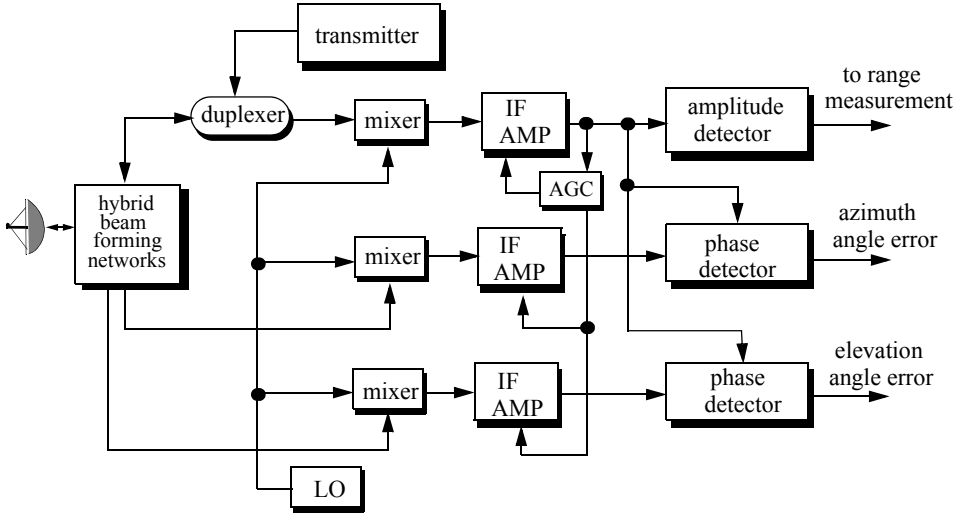

Figure 9.10. Simplified amplitude comparison monopulse radar block diagram.

and a difference signal in the same coordinate is

$$\Delta(\varphi) = \frac{\sin(\varphi - \varphi_0)}{(\varphi - \varphi_0)} - \frac{\sin(\varphi + \varphi_0)}{(\varphi + \varphi_0)} \tag{9.8}$$

MATLAB Function "mono_pulse.m"

The function "mono_pulse.m" implements Eqs. (9.7) and (9.8). Its output includes plots of the sum and difference antenna patterns as well as the difference-to-sum ratio. It is given in Listing 9.1 in Section 9.11. The syntax is as follows:

mono_pulse (phi0)

where *phi0* is the squint angle in radians.

Fig. 9.11 (a-c) shows the corresponding plots for the sum and difference patterns for $\varphi_0 = 0.15$ radians. Fig. 9.12 (a-c) is similar to Fig. 9.11, 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.

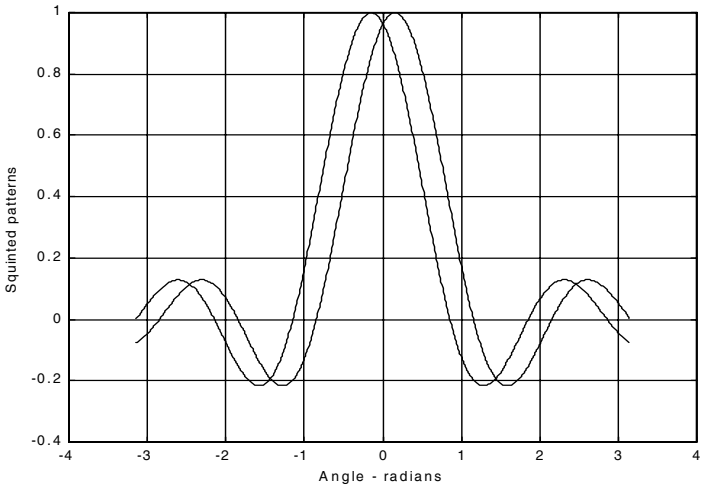

Figure 9.11a. Two squinted patterns. Squint angle is $\varphi_0 = 0.15$ radians.

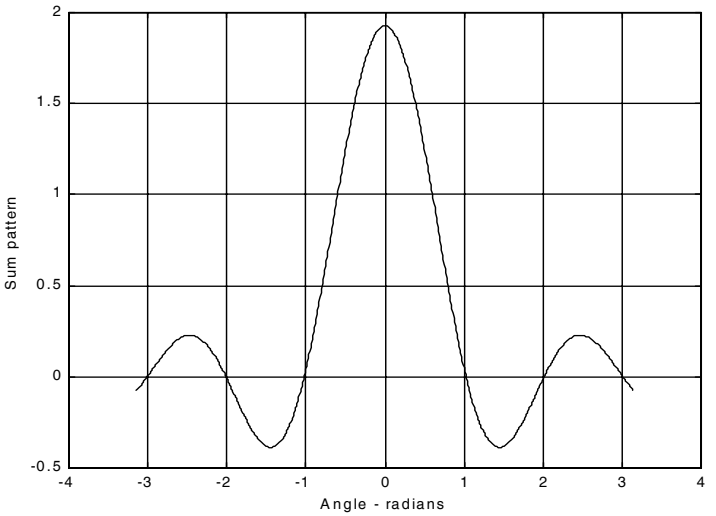

Figure 9.11b. Sum pattern corresponding to Fig. 9.11a.

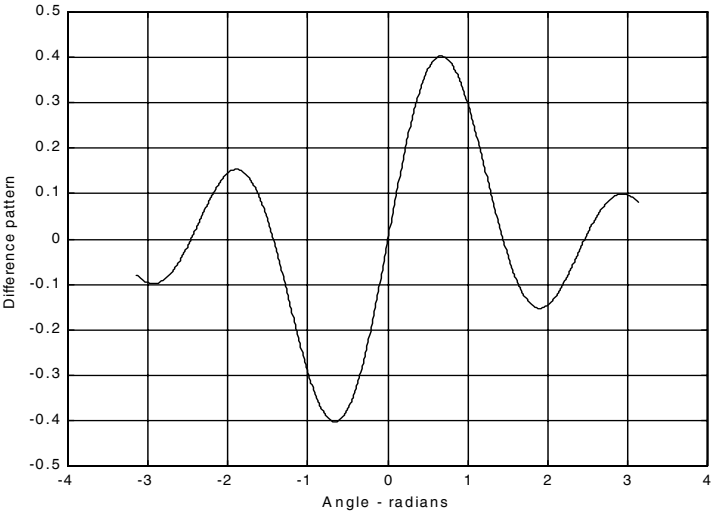

Figure 9.11c. Difference pattern corresponding to Fig. 9.11a.

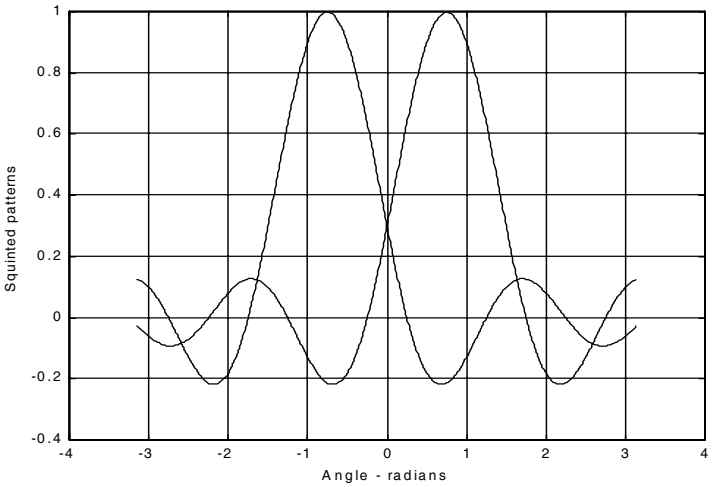

Figure 9.12a. Two squinted patterns. Squint angle is $\varphi_0 = 0.75$ radians.

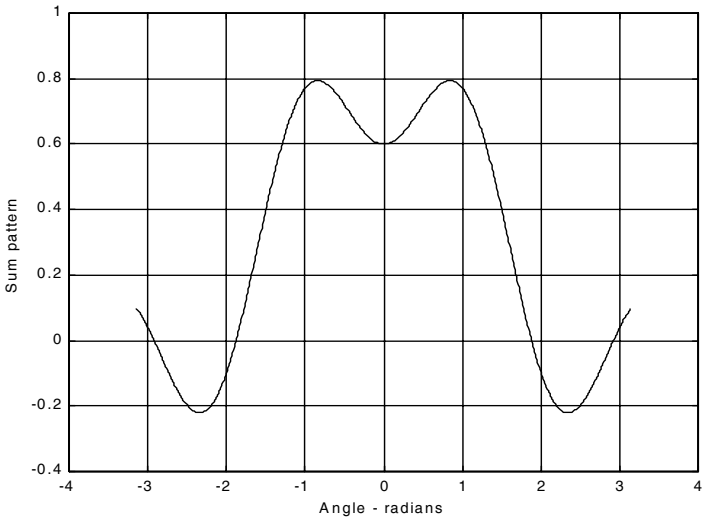

Figure 9.12b. Sum pattern corresponding to Fig. 9.12a.

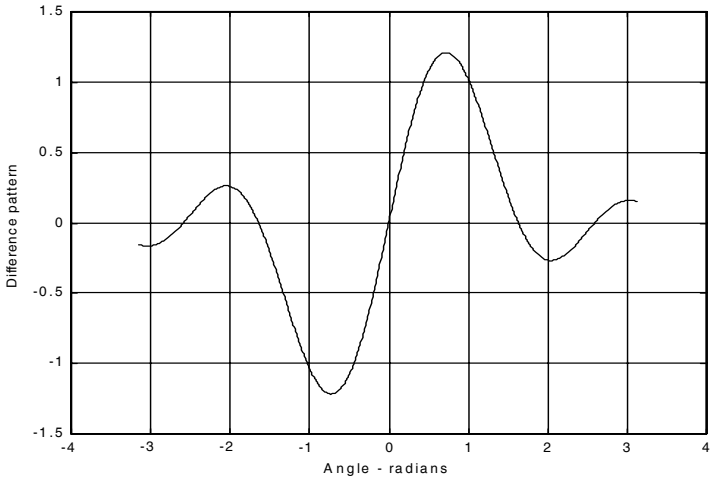

Figure 9.12c. Difference pattern corresponding to Fig. 9.12a.

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

$$S_1 = A + D \tag{9.9}$$

$$S_2 = B + C \tag{9.10}$$

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

$$\varepsilon_\varphi = \frac{|\Delta|}{|\Sigma|} \cos\xi \tag{9.11}$$

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 track-

ing axis; otherwise it is off the tracking axis. Fig. 9.13 (a,b) shows a plot for the ratio Δ/Σ for the monopulse radar whose sum and difference patterns are in Figs. 9.11 and 9.12.

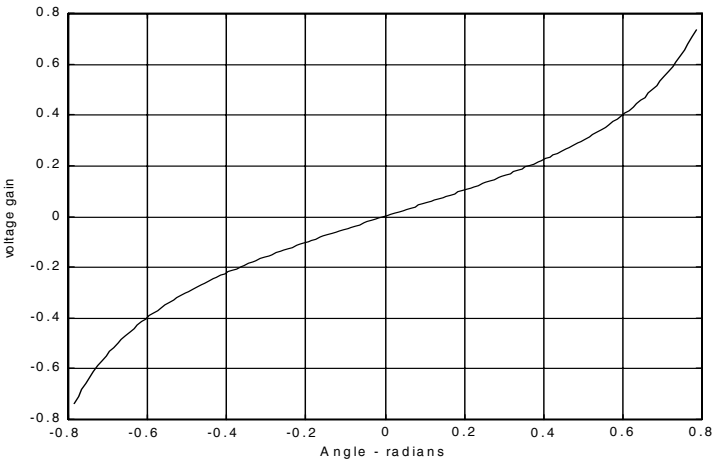

Figure 9.13a. Difference-to-sum ratio corresponding to Fig. 9.11a.

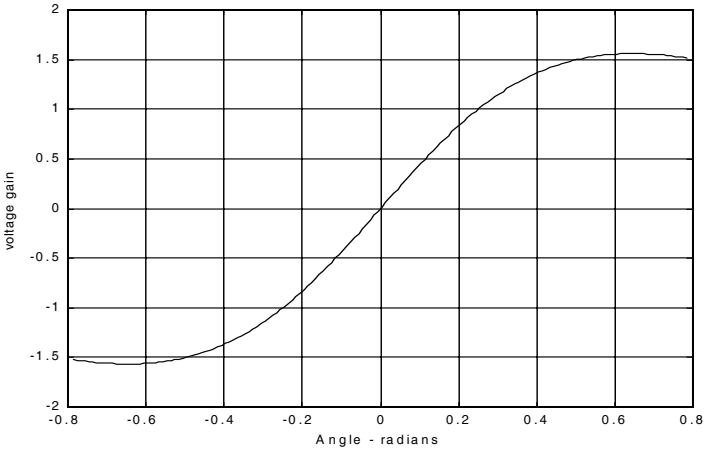

Figure 9.13b. Difference-to-sum ratio corresponding to Fig. 9.12a.

9.3. 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. 9.14. A phase error signal (for each coordinate) is computed from the phase difference between the signals generated in the antenna elements.

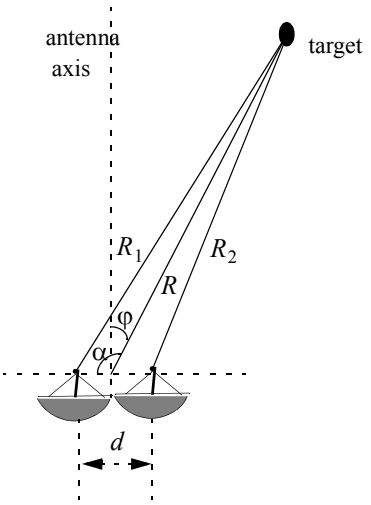

Figure 9.14. Single coordinate phase comparison monopulse antenna.

Consider Fig. 9.14; since the angle α is equal to $\varphi + \pi/2$, it follows that

$$R_1^2 = R^2 + \left(\frac{d}{2}\right)^2 - 2\frac{d}{2}R\cos\left(\varphi + \frac{\pi}{2}\right) \qquad \textbf{(9.12)}$$

$$= R^2 + \frac{d^2}{4} - dR\sin\varphi$$

and since $d \ll R$ we can use the binomial series expansion to get

$$R_1 \approx R\left(1 + \frac{d}{2R}\sin\varphi\right) \qquad \textbf{(9.13)}$$

Similarly,

$$R_2 \approx R\left(1-\frac{d}{2R}\sin\varphi\right) \tag{9.14}$$

The phase difference between the two elements is then given by

$$\phi = \frac{2\pi}{\lambda}(R_1 - R_2) = \frac{2\pi}{\lambda}d\sin\varphi \tag{9.15}$$

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. 9.15.

The (single coordinate) sum and difference signals are, respectively, given by

$$\Sigma(\varphi) = S_1 + S_2 \tag{9.16}$$

$$\Delta(\varphi) = S_1 - S_2 \tag{9.17}$$

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

$$S_1 = S_2 e^{-j\phi} \tag{9.18}$$

It follows that

$$\Delta(\varphi) = S_2(1 - e^{-j\phi}) \tag{9.19}$$

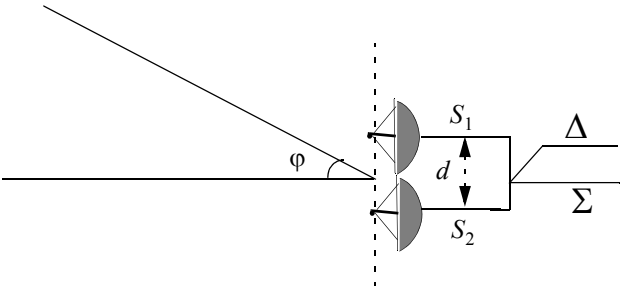

Figure 9.15. Single coordinate phase monopulse antenna, with sum and difference channels.

$$\Sigma(\varphi) = S_2(1 + e^{-j\phi}) \tag{9.20}$$

The phase error signal is computed from the ratio Δ/Σ. More precisely,

$$\frac{\Delta}{\Sigma} = \frac{1 - e^{-j\phi}}{1 + e^{-j\phi}} = j\tan\left(\frac{\phi}{2}\right) \tag{9.21}$$

which is purely imaginary. The modulus of the error signal is then given by

$$\frac{|\Delta|}{|\Sigma|} = \tan\left(\frac{\phi}{2}\right) \tag{9.22}$$

This kind of phase comparison monopulse tracker is often called the half-angle tracker.

9.4. 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. 9.16, 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.

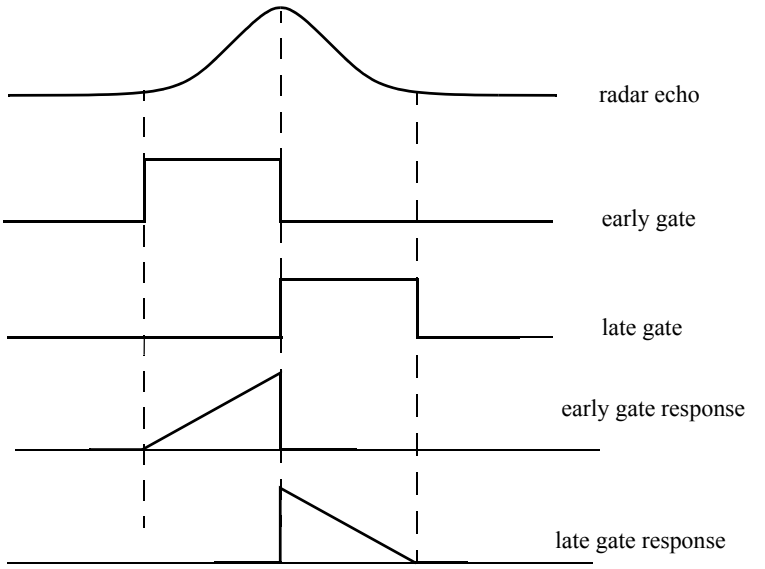

Figure 9.16. Illustration of split-range gate.

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 are introduced and analyzed. As a matter of notation, small case letters, with an underbar, are used.

9.5. 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 is exercised so

that the detection is assigned to the proper track. A simplified TWS data processing block diagram is shown in Fig. 9.17.

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.

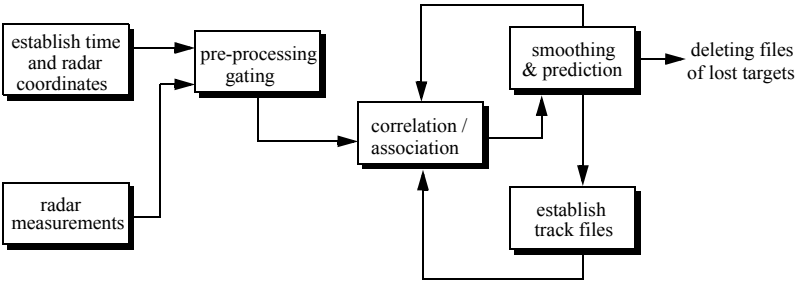

Figure. 9.17. Simplified block diagram of TWS data processing.

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 predetermined association rules can be utilized so that a single observation is assigned to a single track file.

9.6. 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

$$\underline{x} = \begin{bmatrix} R \\ \dot{R} \\ \ddot{R} \end{bmatrix} \tag{9.23}$$

where R, $\dot{R}$, and $\ddot{R}$ are, respectively, the range measurement, range rate (velocity), and acceleration. The state vector defined in Eq. (9.23) 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:

$$\underline{x} = \left[x_1 \; \dot{x}_1 \; \ldots \; x_2 \; \dot{x}_2 \; \ldots \; x_n \; \dot{x}_n \; \ldots \right]^t \tag{9.24}$$

where the superscript indicates the transpose operation.

The LTI system of interest can be represented using the following state equations:

$$\dot{\underline{x}}(t) = \underline{A} \; \underline{x}(t) + \underline{B}\underline{w}(t) \tag{9.25}$$

$$\underline{y}(t) = \underline{C} \; \underline{x}(t) + \underline{D}\underline{w}(t) \tag{9.26}$$

where: $\dot{\underline{x}}$ is the value of the $n \times 1$ state vector; $\underline{y}$ is the value of the $p \times 1$ output vector; $\underline{w}$ is the value of the $m \times 1$ input vector; $\underline{A}$ is an $n \times n$ matrix; $\underline{B}$ is an $n \times m$ matrix; $\underline{C}$ is $p \times n$ matrix; and $\underline{D}$ is an $p \times m$ matrix. The

homogeneous solution (i.e., $\underline{w} = \underline{0}$) to this linear system, assuming known initial condition $\underline{x}(0)$ at time t_0, has the form

$$\underline{x}(t) = \underline{\Phi}(t - t_0)\underline{x}(t - t_0) \tag{9.27}$$

The matrix $\underline{\Phi}$ is known as the state transition matrix, or fundamental matrix, and is equal to

$$\underline{\Phi}(t - t_0) = e^{\underline{A}(t - t_0)} \tag{9.28}$$

Eq. (9.28) can be expressed in series format as

$$\underline{\Phi}(t - t_0)\big|_{t_0 = 0} = e^{\underline{A}(t)} = \underline{I} + \underline{A}t + \underline{A}^2\frac{t^2}{2!} + \ldots = \sum_{k=0}^{\infty} \underline{A}^k \frac{t^k}{k!} \tag{9.29}$$

Example:

Compute the state transition matrix for an LTI system when

$$\underline{A} = \begin{bmatrix} 0 & 1 \\ -0.5 & -1 \end{bmatrix}$$

Solution:

The state transition matrix can be computed using Eq. (9.29). For this purpose, compute $\underline{A}^2$ *and* $\underline{A}^3 \ldots$. *It follows*

$$\underline{A}^2 = \begin{bmatrix} -\frac{1}{2} & -1 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad \underline{A}^3 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{4} & 0 \end{bmatrix} \qquad \ldots$$

Therefore,

$$\underline{\Phi} = \begin{bmatrix} 1 + 0t - \frac{\frac{1}{2}t^2}{2!} + \frac{\frac{1}{2}t^3}{3!} + \ldots & 0 + t - \frac{t^2}{2!} + \frac{\frac{1}{2}t^3}{3!} + \ldots \\ 0 - \frac{1}{2}t + \frac{\frac{1}{2}t^2}{2!} - \frac{\frac{1}{4}t^3}{3!} + \ldots & 1 - t + \frac{\frac{1}{2}t^2}{2!} + \frac{0t^3}{3!} + \ldots \end{bmatrix}$$

The state transition matrix has the following properties (the proof is left as an exercise):

1. *Derivative property*

$$\frac{\partial}{\partial t}\underline{\Phi}(t-t_0) = \underline{A}\underline{\Phi}(t-t_0) \tag{9.30}$$

2. *Identity property*

$$\underline{\Phi}(t_0-t_0) = \underline{\Phi}(0) = \underline{I} \tag{9.31}$$

3. *Initial value property*

$$\left.\frac{\partial}{\partial t}\underline{\Phi}(t-t_0)\right|_{t=t_0} = \underline{A} \tag{9.32}$$

4. *Transition property*

$$\underline{\Phi}(t_2-t_0) = \underline{\Phi}(t_2-t_1)\underline{\Phi}(t_1-t_0) \qquad ; \; t_0 \le t_1 \le t_2 \tag{9.33}$$

5. *Inverse property*

$$\underline{\Phi}(t_0-t_1) = \underline{\Phi}^{-1}(t_1-t_0) \tag{9.34}$$

6. *Separation property*

$$\underline{\Phi}(t_1-t_0) = \underline{\Phi}(t_1)\underline{\Phi}^{-1}(t_0) \tag{9.35}$$

The general solution to the system defined in Eq. (9.25) can be written as

$$\underline{x}(t) = \underline{\Phi}(t-t_0)\underline{x}(t_0) + \int_{t_0}^{t}\underline{\Phi}(t-\tau)\underline{B}\underline{w}(\tau)d\tau \tag{9.36}$$

The first term of the right-hand side of Eq. (9.36) 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. (9.26) and (9.36) an expression for the output is computed as

$$\underline{y}(t) = \underline{C}e^{\underline{A}(t-t_0)}\underline{x}(t_0) + \int_{t_0}^{t}[\underline{C}e^{\underline{A}(t-\tau)}\underline{B} - \underline{D}\delta(t-\tau)]\underline{w}(\tau)d\tau \tag{9.37}$$

Note that the system impulse response is equal to $\underline{C}e^{\underline{A}t}\underline{B} - \underline{D}\delta(t)$.

The difference equations describing a discrete time system, equivalent to Eqs. (9.25) and (9.26), are

$$\underline{x}(n+1) = \underline{A}\;\underline{x}(n) + \underline{B}\underline{w}(n) \tag{9.38}$$

$$\underline{y}(n) = \underline{C}\;\underline{x}(n) + \underline{D}\underline{w}(n) \tag{9.39}$$

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. (9.38), with initial condition $\underline{x}(n_0)$, is

$$\underline{x}(n) = \underline{A}^{n-n_0}\underline{x}(n_0) \tag{9.40}$$

In this case the state transition matrix is an $n \times n$ matrix given by

$$\underline{\Phi}(n, n_0) = \underline{\Phi}(n-n_0) = \underline{A}^{n-n_0} \tag{9.41}$$

The following is the list of properties associated with the discrete transition matrix

$$\underline{\Phi}(n+1-n_0) = \underline{A}\underline{\Phi}(n-n_0) \tag{9.42}$$

$$\underline{\Phi}(n_0-n_0) = \underline{\Phi}(0) = \underline{I} \tag{9.43}$$

$$\underline{\Phi}(n_0+1-n_0) = \underline{\Phi}(1) = \underline{A} \tag{9.44}$$

$$\underline{\Phi}(n_2-n_0) = \underline{\Phi}(n_2-n_1)\underline{\Phi}(n_1-n_0) \tag{9.45}$$

$$\underline{\Phi}(n_0-n_1) = \underline{\Phi}^{-1}(n_1-n_0) \tag{9.46}$$

$$\underline{\Phi}(n_1-n_0) = \underline{\Phi}(n_1)\underline{\Phi}^{-1}(n_0) \tag{9.47}$$

The solution to the general case (i.e., non-homogeneous system) is given by

$$\underline{x}(n) = \underline{\Phi}(n-n_0)\underline{x}(n_0) + \sum_{m=n_0}^{n-1} \underline{\Phi}(n-m-1)\underline{B}\underline{w}(m) \tag{9.48}$$

It follows that the output is given by

$$\underline{y}(n) = \underline{C}\underline{\Phi}(n-n_0)\underline{x}(n_0) + \sum_{m=n_0}^{n-1} \underline{C}\ \underline{\Phi}(n-m-1)\underline{B}\underline{w}(m) + \underline{D}\underline{w}(n) \tag{9.49}$$

where the system impulse response is given by

$$\underline{h}(n) = \sum_{m=n_0}^{n-1} \underline{C}\ \underline{\Phi}(n-m-1)\underline{B}\underline{\delta}(m) + \underline{D}\underline{\delta}(n) \tag{9.50}$$

Taking the Z-transform for Eqs. (9.38) and (9.39) yields

$$z\underline{x}(z) = \underline{A}\underline{x}(z) + \underline{B}\underline{w}(z) + z\underline{x}(0) \tag{9.51}$$

$$\underline{y}(z) = \underline{C}\underline{x}(z) + \underline{D}\underline{w}(z) \tag{9.52}$$

Manipulating Eqs. (9.51) and (9.52) yields

$$\underline{x}(z) = [z\underline{I} - \underline{A}]^{-1}\underline{B}\underline{w}(z) + [z\underline{I} - \underline{A}]^{-1}z\underline{x}(0) \tag{9.53}$$

$$\underline{y}(z) = \{\underline{C}[z\underline{I} - \underline{A}]^{-1}\underline{B} + \underline{D}\}\underline{w}(z) + \underline{C}[z\underline{I} - \underline{A}]^{-1}z\underline{x}(0) \tag{9.54}$$

It follows that the state transition matrix is

$$\underline{\Phi}(z) = z[z\underline{I} - \underline{A}]^{-1} = [\underline{I} - z^{-1}\underline{A}]^{-1} \tag{9.55}$$

and the system impulse response in the z-domain is

$$\underline{h}(z) = \underline{C}\underline{\Phi}(z)z^{-1}\underline{B} + \underline{D} \tag{9.56}$$

9.7. 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. 9.18. 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

$$\dot{\underline{x}}(t) = \underline{A}\ \underline{x}(t) + \underline{u}(t) \tag{9.57}$$

$$\underline{y}(t) = \underline{G}\ \underline{x}(t) + \underline{v}(t) \tag{9.58}$$

where $\underline{y}$ is the observable part of the system (i.e., output), $\underline{u}$ is a driving force, and $\underline{v}$ is the measurement noise. The matrices $\underline{A}$ and $\underline{G}$ vary depending on the system. The noise observation $\underline{v}$ is assumed to be uncorrelated. If the initial condition vector is $\underline{x}(t_0)$, then from Eq. (9.36) we get

$$\underline{x}(t) = \underline{\Phi}(t - t_0)\underline{x}(t_0) + \int_{t_0}^{t} \underline{\Phi}(t - \tau)\underline{u}(\tau)d\tau \tag{9.59}$$

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. (9.57) and (9.58) are

$$\underline{x}(n) = \underline{A}\ \underline{x}(n - 1) + \underline{u}(n) \tag{9.60}$$

$$\underline{y}(n) = \underline{G}\ \underline{x}(n) + \underline{v}(n) \tag{9.61}$$

The homogeneous solution to this system is given in Eq. (9.27) for continuous time, and in Eq. (9.40) for discrete time.

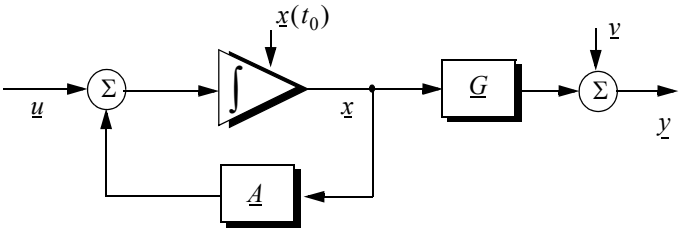

Figure 9.18. An LTI system.

The state transition matrix corresponding to this system can be obtained using Taylor series expansion of the vector $\underline{x}$. More precisely,

$$\begin{aligned} x &= x + T\dot{x} + \frac{T^2}{2!}\ddot{x} + \ldots \\ \dot{x} &= \dot{x} + T\ddot{x} + \ldots \\ \ddot{x} &= \ddot{x} + \ldots \end{aligned} \tag{9.62}$$

It follows that the elements of the state transition matrix are defined by

$$\underline{\Phi}[ij] = \begin{Bmatrix} T^{j-i} \div (j-i)! & 1 \le i, j \le n \\ 0 & j < i \end{Bmatrix} \tag{9.63}$$

Using matrix notation, the state transition matrix is then given by

$$\underline{\Phi} = \begin{bmatrix} 1 & T & \frac{T^2}{2!} & \ldots \\ 0 & 1 & T & \ldots \\ 0 & 0 & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{bmatrix} \tag{9.64}$$

The matrix given in Eq. (9.64) is often called the Newtonian matrix.

9.8. 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

$$\underline{x}(n|n) = \underline{\Phi}\underline{x}(n-1|n-1) + \underline{K}[y_n - \underline{G}\underline{\Phi}\underline{x}(n-1|n-1)] \tag{9.65}$$

Since the transition matrix assists in predicting the next state,

$$\underline{x}(n+1|n) = \underline{\Phi}\underline{x}(n|n) \tag{9.66}$$

Substituting Eq. (9.66) into Eq. (9.65) yields

$$\underline{x}(n|n) = \underline{x}(n|n-1) + \underline{K}[y_n - \underline{G}\underline{x}(n|n-1)] \tag{9.67}$$

The term enclosed within the brackets on the right hand side of Eq. (9.67) is often called the residual (error) which is the difference between the measured input and predicted output. Eq. (9.67) means that the estimate of $\underline{x}(n)$ is the sum of the prediction and the weighted residual. The term $\underline{G}\underline{x}(n|n-1)$ represents the prediction state. In the case of the $\alpha\beta\gamma$ estimator, $\underline{G}$ is the row vector given by

$$\underline{G} = \begin{bmatrix} 1 & 0 & 0 & \ldots \end{bmatrix} \tag{9.68}$$

and the gain matrix $\underline{K}$ is given by

$$\underline{K} = \begin{bmatrix} \alpha \\ \beta/T \\ \gamma/T^2 \end{bmatrix} \tag{9.69}$$

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

$$\underline{C}(n|n) = E\{(\underline{x}(n|n)\ )\underline{x}^t(n|n)\} \qquad ;\ y_n = v_n \tag{9.70}$$

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 also assumed to be uncorrelated,

$$E\{v_n v_m\} = \begin{cases} \delta\sigma_v^2 & n = m \\ 0 & n \neq m \end{cases} \tag{9.71}$$

Eq. (9.65) can be written as

$$\underline{x}(n|n) = \underline{A}\underline{x}(n-1|n-1) + \underline{K}y_n \tag{9.72}$$

where

$$\underline{A} = (\underline{I} - \underline{K}\underline{G})\underline{\Phi} \tag{9.73}$$

Substituting Eqs. (9.72) and (9.73) into Eq. (9.70) yields

$$\underline{C}(n|n) = E\{(\underline{A}\underline{x}(n-1|n-1) + \underline{K}y_n)(\underline{A}\underline{x}(n-1|n-1) + \underline{K}y_n)^t\} \tag{9.74}$$

Expanding the right hand side of Eq. (9.74) and using Eq. (9.71) give

$$\underline{C}(n|n) = \underline{A}\underline{C}(n-1|n-1)\underline{A}^t + \underline{K}\sigma_v^2\underline{K}^t \tag{9.75}$$

Under the steady state condition, Eq. (9.75) collapses to

$$\underline{C}(n|n) = \underline{A}\underline{C}\underline{A}^t + \underline{K}\sigma_v^2\underline{K}^t \tag{9.76}$$

where $\underline{C}$ is the steady state noise covariance matrix. In the steady state,

$$\underline{C}(n|n) = \underline{C}(n-1|n-1) = \underline{C} \qquad for\ any\ n \tag{9.77}$$

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. (9.72),

$$\underline{x}(z) = \underline{A}z^{-1}\underline{x}(z) + \underline{K}y_n(z) \tag{9.78}$$

Rearranging Eq. (9.78) yields the following system transfer functions:

$$\underline{h}(z) = \frac{\underline{x}(z)}{y_n(z)} = (\underline{I} - \underline{A}z^{-1})^{-1}\underline{K} \tag{9.79}$$

where $(\underline{I} - \underline{A}z^{-1})$ 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,

$$\left|(\underline{I} - \underline{A}z^{-1})\right| = 0 \tag{9.80}$$

The filter's steady state errors can be determined with the help of Fig. 9.19. The error transfer function is

$$\underline{e}(z) = \frac{\underline{y}(z)}{1 + \underline{h}(z)} \tag{9.81}$$

and by using Abel's theorem, the steady state error is

$$\underline{e}_\infty = \lim_{t \to \infty} \underline{e}(t) = \lim_{z \to 1}\left(\frac{z-1}{z}\right) \underline{e}(z) \tag{9.82}$$

Substituting Eq. (9.82) into (9.81) yields

$$\underline{e}_\infty = \lim_{z \to 1} \frac{z-1}{z} \frac{\underline{y}(z)}{1 + \underline{h}(z)} \tag{9.83}$$

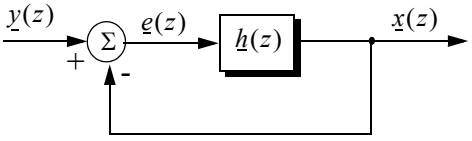

Figure 9.19. Steady state error computation.

9.8.1. The $\alpha\beta$ Filter

The $\alpha\beta$ tracker produces, on the nth observation, smoothed estimates for position and velocity, and a predicted position for the $(n+1)th$ observation. Fig. 9.20 shows an implementation of this filter. Note that the subscripts "p" and "s" are used to indicate, respectively, the predicated and smoothed values. The $\alpha\beta$ 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:

$$x_s(n) = x(n|n) = x_p(n) + \alpha(x_0(n) - x_p(n)) \tag{9.84}$$

$$\dot{x}_s(n) = x'(n|n) = \dot{x}_s(n-1) + \frac{\beta}{T}(x_0(n) - x_p(n)) \tag{9.85}$$

x_0 is the position input samples. The predicted position is given by

$$x_p(n) = x_s(n|n-1) = x_s(n-1) + T\dot{x}_s(n-1) \tag{9.86}$$

The initialization process is defined by

$$x_s(1) = x_p(2) = x_0(1)$$

$$\dot{x}_s(1) = 0$$

$$\dot{x}_s(2) = \frac{x_0(2) - x_0(1)}{T}$$

A general form for the covariance matrix was developed in the previous section, and is given in Eq. (9.75). In general, a second order one-dimensional covariance matrix (in the context of the $\alpha\beta$ filter) can be written as

$$\underline{C}(n|n) = \begin{bmatrix} C_{xx} & C_{x\dot{x}} \\ C_{\dot{x}x} & C_{\dot{x}\dot{x}} \end{bmatrix} \tag{9.87}$$

where, in general, C_{xy} is

$$C_{xy} = E\{xy^t\} \tag{9.88}$$

By inspection, the $\alpha\beta$ filter has

$$\underline{A} = \begin{bmatrix} 1-\alpha & (1-\alpha)T \\ -\beta/T & (1-\beta) \end{bmatrix} \tag{9.89}$$

$$\underline{K} = \begin{bmatrix} \alpha \\ \beta/T \end{bmatrix} \tag{9.90}$$

$$\underline{G} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{9.91}$$

$$\underline{\Phi} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{9.92}$$

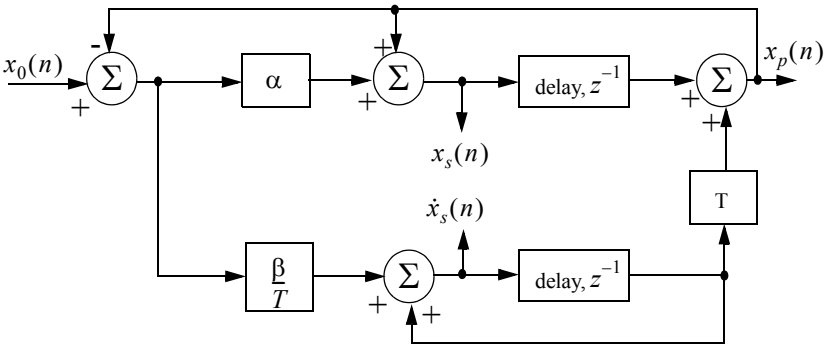

Figure 9.20. An implementation of an $\alpha\beta$ tracker.

Finally, using Eqs. (9.89) through (9.92) in Eq. (9.72) yields the steady state noise covariance matrix,

$$\underline{C} = \frac{\sigma_v^2}{\alpha(4-2\alpha-\beta)} \begin{bmatrix} 2\alpha^2 - 3\alpha\beta + 2\beta & \dfrac{\beta(2\alpha-\beta)}{T} \\ \dfrac{\beta(2\alpha-\beta)}{T} & \dfrac{2\beta^2}{T^2} \end{bmatrix} \tag{9.93}$$

It follows that the position and velocity VRR ratios are, respectively, given by

$$(VRR)_x = C_{xx}/\sigma_v^2 = \frac{2\alpha^2 - 3\alpha\beta + 2\beta}{\alpha(4-2\alpha-\beta)} \tag{9.94}$$

$$(VRR)_{\dot{x}} = C_{\dot{x}\dot{x}}/\sigma_v^2 = \frac{1}{T^2}\ \frac{2\beta^2}{\alpha(4-2\alpha-\beta)} \tag{9.95}$$

The stability of the $\alpha\beta$ filter is determined from its system transfer functions. For this purpose, compute the roots for Eq. (9.80) with $\underline{A}$ from Eq. (9.89),

$$\left|\underline{I} - \underline{A}z^{-1}\right| = 1 - (2 - \alpha - \beta)z^{-1} + (1 - \alpha)z^{-2} = 0 \tag{9.96}$$

Solving Eq. (9.96) for z yields

$$z_{1,2} = 1 - \frac{\alpha + \beta}{2} \pm \frac{1}{2}\sqrt{(\alpha - \beta)^2 - 4\beta} \tag{9.97}$$

and in order to guarantee stability

$$\left|z_{1,2}\right| < 1 \tag{9.98}$$

Two cases are analyzed. First, $z_{1,2}$ are real. In this case (the details are left as an exercise),

$$\beta > 0 \qquad ; \ \alpha > -\beta \tag{9.99}$$

The second case is when the roots are complex; in this case we find

$$\alpha > 0 \tag{9.100}$$

The system transfer functions can be derived by using Eqs. (9.79), (9.89), and (9.90),

$$\begin{bmatrix} h_x(z) \\ h_{\dot{x}}(z) \end{bmatrix} = \frac{1}{z^2 - z(2 - \alpha - \beta) + (1 - \alpha)} \begin{bmatrix} \alpha z\left(z - \dfrac{(\alpha - \beta)}{\alpha}\right) \\ \dfrac{\beta z(z - 1)}{T} \end{bmatrix} \tag{9.101}$$

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 advan-

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.

tage 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

$$\beta = \frac{\alpha^2}{2-\alpha} \tag{9.102}$$

In this case, the position and velocity VRR ratios are, respectively, given by

$$(VRR)_x = \frac{\alpha(6-5\alpha)}{\alpha^2 - 8\alpha + 8} \tag{9.103}$$

$$(VRR)_{\dot{x}} = \frac{2}{T^2}\,\frac{\alpha^3/(2-\alpha)}{\alpha^2 - 8\alpha + 8} \tag{9.104}$$

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

$$\alpha = 1 - \xi^2 \tag{9.105}$$

$$\beta = (1-\xi)^2 \tag{9.106}$$

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

$$C_{xx} = \frac{1-\xi}{(1+\xi)^3}\,(1 + 4\xi + 5\xi^2)\,\sigma_v^2 \tag{9.107}$$

$$C_{x\dot{x}} = C_{\dot{x}x} = \frac{1}{T}\,\frac{1-\xi}{(1+\xi)^3}\,(1 + 2\xi + 3\xi^2)\,\sigma_v^2 \tag{9.108}$$

$$C_{\dot{x}\dot{x}} = \frac{2}{T^2}\,\frac{1-\xi}{(1+\xi)^3}\,(1-\xi)^2\,\sigma_v^2 \tag{9.109}$$

9.8.2. 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. 9.21.

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:

$$x_s(n) = x_p(n) + \alpha(x_0(n) - x_p(n)) \tag{9.110}$$

$$\dot{x}_s(n) = \dot{x}_s(n-1) + T\ddot{x}_s(n-1) + \frac{\beta}{T}\,(x_0(n) - x_p(n)) \tag{9.111}$$

$$\ddot{x}_s(n) = \ddot{x}_s(n-1) + \frac{2\gamma}{T^2}\,(x_0(n) - x_p(n)) \tag{9.112}$$

$$x_p(n+1) = x_s(n) + T\,\dot{x}_s(n) + \frac{T^2}{2}\,\ddot{x}_s(n) \tag{9.113}$$

and the initialization process is

$$x_s(1) = x_p(2) = x_0(1)$$

$$\dot{x}_s(1) = \ddot{x}_s(1) = \ddot{x}_s(2) = 0$$

$$\dot{x}_s(2) = \frac{x_0(2) - x_0(1)}{T}$$

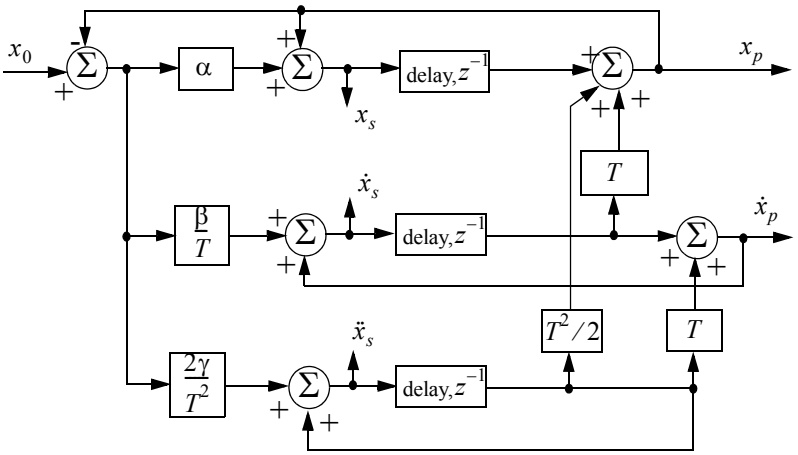

Figure 9.21. An implementation for an $\alpha\beta\gamma$ tracker.

$$\ddot{x}_s(3) = \frac{x_0(3) + x_0(1) - 2x_0(2)}{T^2}$$

Using Eq. (9.63) the state transition matrix for the $\alpha\beta\gamma$ filter is

$$\underline{\Phi} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \tag{9.114}$$

The covariance matrix (which is symmetric) can be computed from Eq. (9.76). For this purpose, note that

$$\underline{K} = \begin{bmatrix} \alpha \\ \beta/T \\ \gamma/T^2 \end{bmatrix} \tag{9.115}$$

$$\underline{G} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{9.116}$$

and

$$\underline{A} = (\underline{I} - \underline{K}\underline{G})\underline{\Phi} = \begin{bmatrix} 1-\alpha & (1-\alpha)T & (1-\alpha)T^2/2 \\ -\beta/T & -\beta+1 & (1-\beta/2)T \\ -2\gamma/T^2 & -2\gamma/T & (1-\gamma) \end{bmatrix} \tag{9.117}$$

Substituting Eq. (9.117) into (9.76) and collecting terms the VRR ratios are computed as

$$(VRR)_x = \frac{2\beta(2\alpha^2 + 2\beta - 3\alpha\beta) - \alpha\gamma(4 - 2\alpha - \beta)}{(4 - 2\alpha - \beta)(2\alpha\beta + \alpha\gamma - 2\gamma)} \tag{9.118}$$

$$(VRR)_{\dot{x}} = \frac{4\beta^3 - 4\beta^2\gamma + 2\gamma^2(2-\alpha)}{T^2(4 - 2\alpha - \beta)(2\alpha\beta + \alpha\gamma - 2\gamma)} \tag{9.119}$$

$$(VRR)_{\ddot{x}} = \frac{4\beta\gamma^2}{T^4(4 - 2\alpha - \beta)(2\alpha\beta + \alpha\gamma - 2\gamma)} \tag{9.120}$$

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

$$\left|\underline{I} - \underline{A}z^{-1}\right| = 0 \tag{9.121}$$

Substituting Eq. (9.117) into (9.121) and collecting terms yield the following characteristic function:

$$f(z) = z^3 + (-3\alpha + \beta + \gamma)z^2 + (3 - \beta - 2\alpha + \gamma)z - (1 - \alpha) \tag{9.122}$$

The $\alpha\beta\gamma$ becomes a Benedict-Bordner filter when

$$2\beta - \alpha\left(\alpha + \beta + \frac{\gamma}{2}\right) = 0 \tag{9.123}$$

Note that for $\gamma = 0$ Eq. (9.123) reduces to Eq. (9.102). For a critically damped filter the gain coefficients are

$$\alpha = 1 - \xi^3 \tag{9.124}$$

$$\beta = 1.5(1 - \xi^2)(1 - \xi) = 1.5(1 - \xi)^2(1 + \xi) \tag{9.125}$$

$$\gamma = (1 - \xi)^3 \tag{9.126}$$

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. It is given in Listing 9.2 in Section 9.11. 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*
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)

which is also part of Listing 9.2. In this case, noise measurements are either to be considered 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. 9.22 and 9.23. Fig. 9.22 assumes an input with lazy maneuvering, while Fig. 9.23 assumes an aggressive maneuvering case. For this purpose, the program called *"fig9_21.m"* was written. It is given in Listing 9.3 in Section 9.11.

Figs. 9.24 and 9.25 show the residual error and predicted position corresponding (generated using the program *"fig9_21.m"*) to Fig. 9.22 for two cases: heavy smoothing and little smoothing with and without noise. The noise is white Gaussian with zero mean and variance of $\sigma_v^2 = 0.05$. Figs. 9. 26 and 9.27 show the residual error and predicted position corresponding (generated using the program *"fig9_20.m")* to Fig. 9.23 with and without noise.

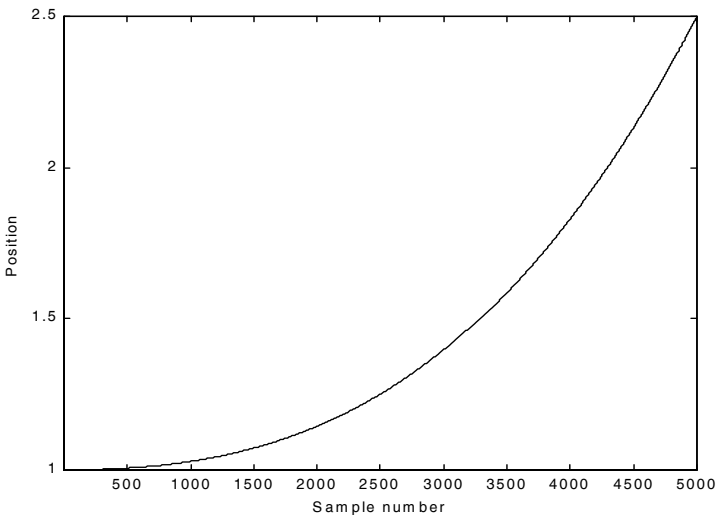

Figure 9.22. Position (truth-data); lazy maneuvering.

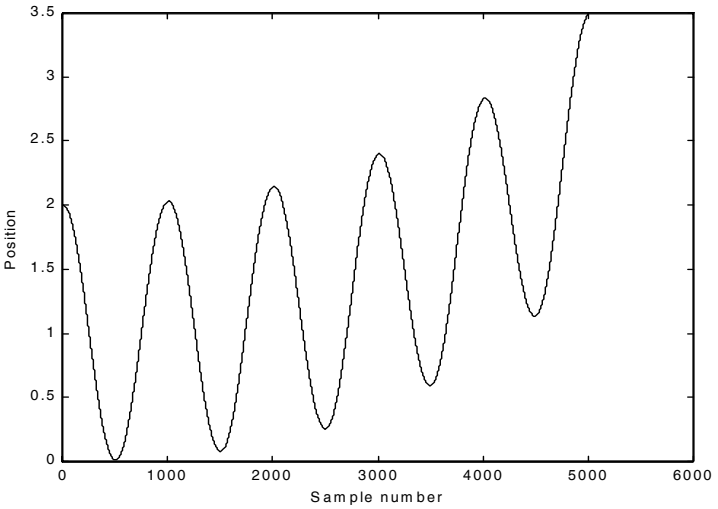

Figure 9.23. Position (truth-data); aggressive maneuvering.

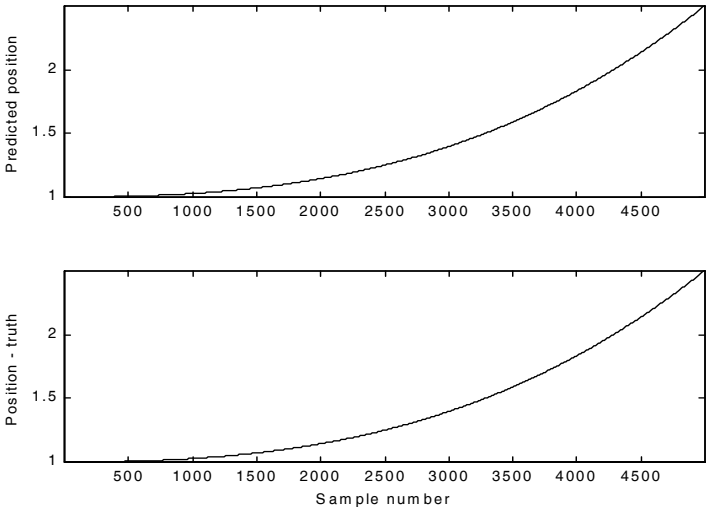

Figure 9.24a-1. Predicted and true position. $\xi = 0.1$ (i.e., large gain coefficients). No noise present.

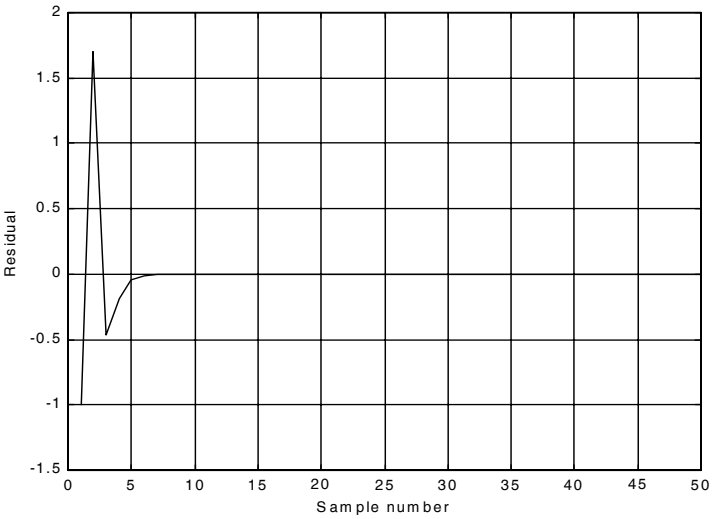

Figure 9.24a-2. Position residual (error). Large gain coefficients. No noise. The error settles to zero fairly quickly.

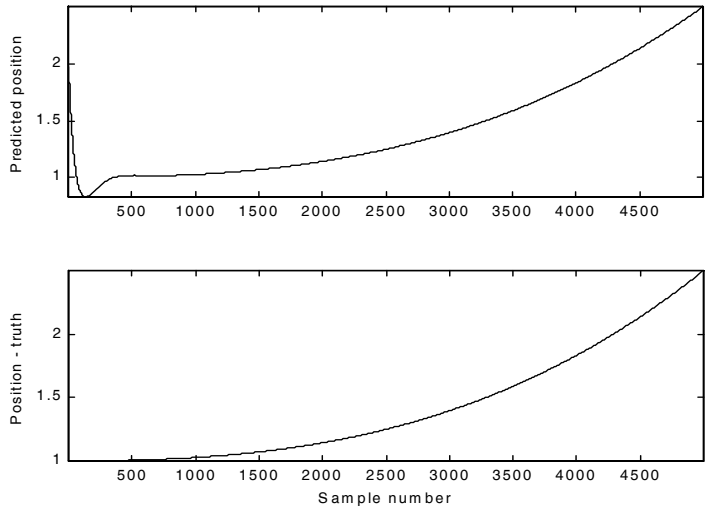

Figure 9.24b-1. Predicted and true position. $\xi = 0.9$ (i.e., small gain coefficients). No noise present.

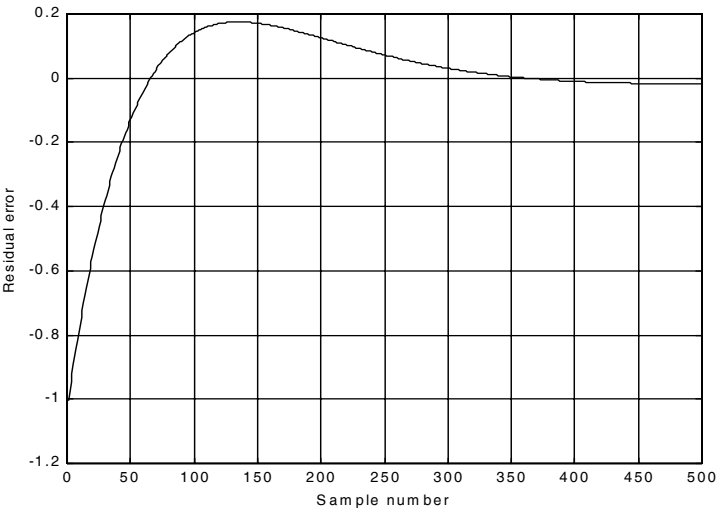

Figure 9.24b-2. Position residual (error). Small gain coefficients. No noise. It takes the filter longer time for the error to settle down.

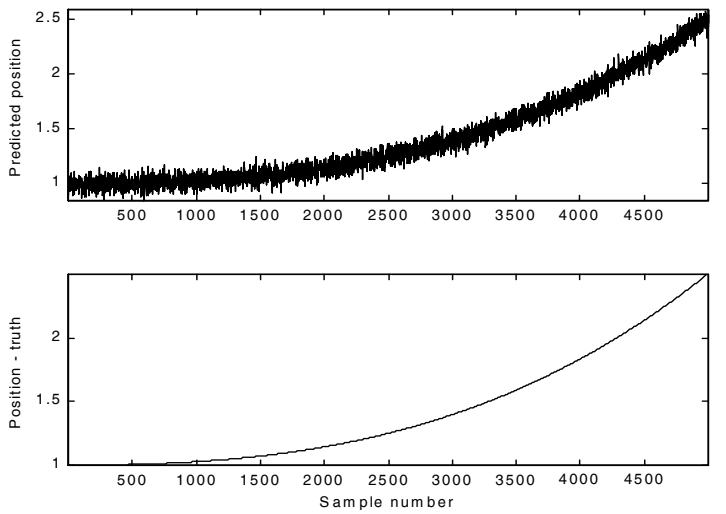

Figure 9.25a-1. Predicted and true position. $\xi = 0.1$ (i.e., large gain coefficients). Noise is present.

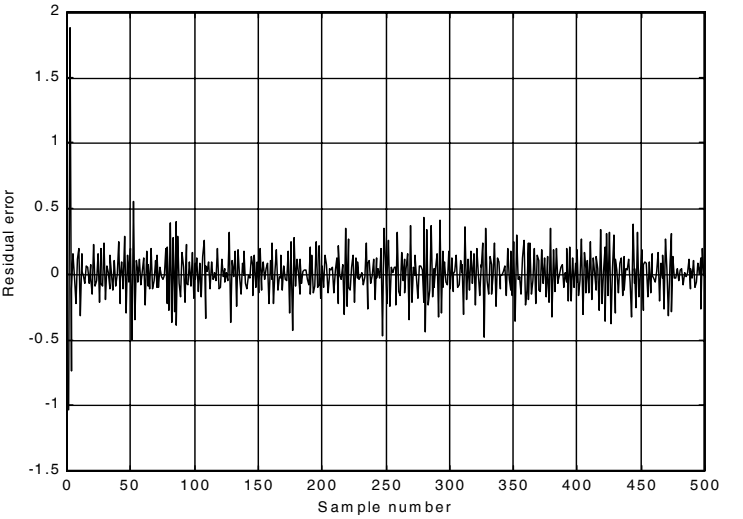

Figure 9.25a-2. Position residual (error). Large gain coefficients. Noise present. The error settles down quickly. The variation is due to noise.

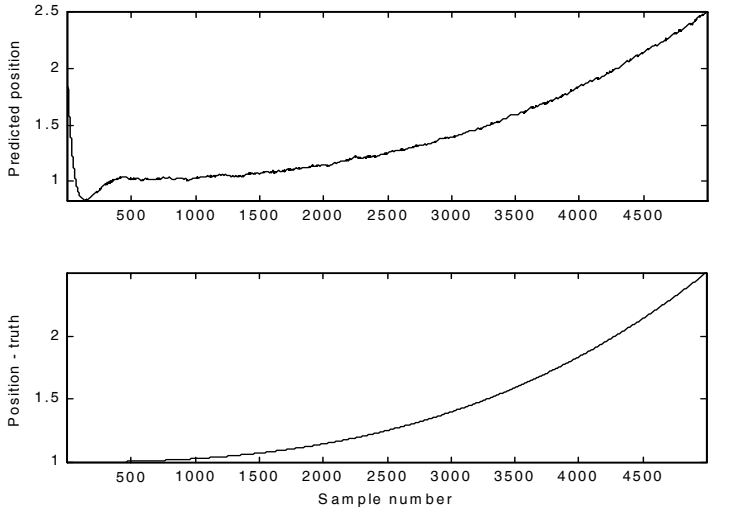

Figure 9.25b-1. Predicted and true position. $\xi = 0.9$ (i.e., small gain coefficients). Noise is present.

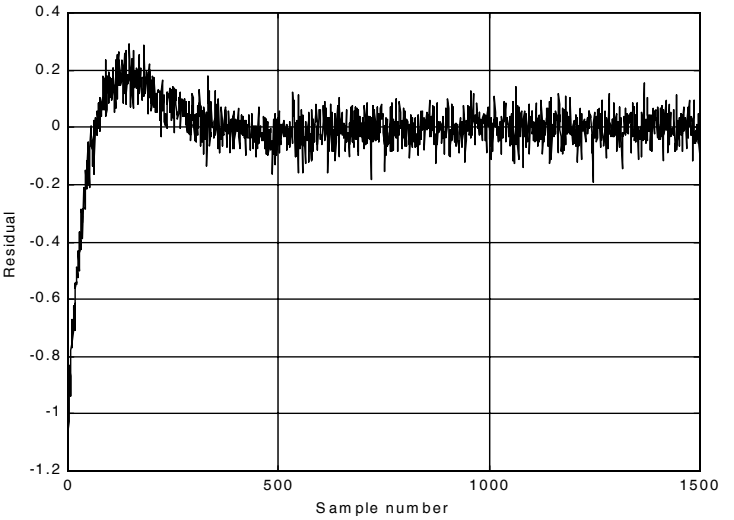

Figure 9.25b-2. Position residual (error). Small gain coefficients. Noise present. The error requires more time before settling down. The variation is due to noise.

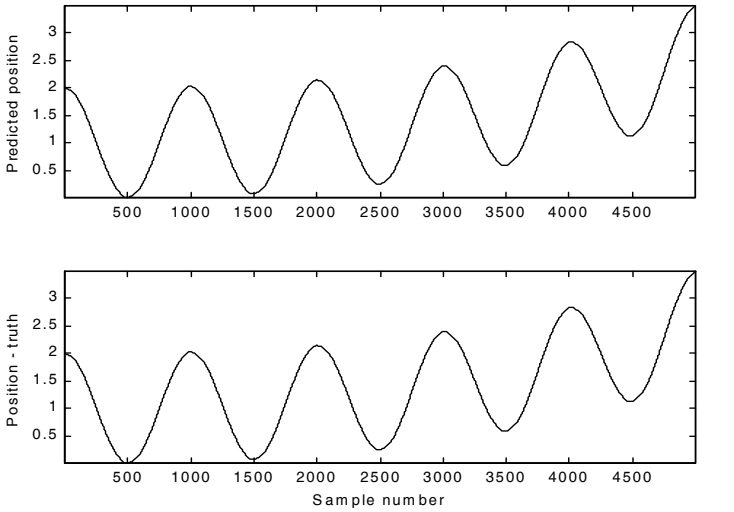

Figure 9.26a. Predicted and true position. $\xi = 0.1$ (i.e., large gain coefficients). Noise is present.

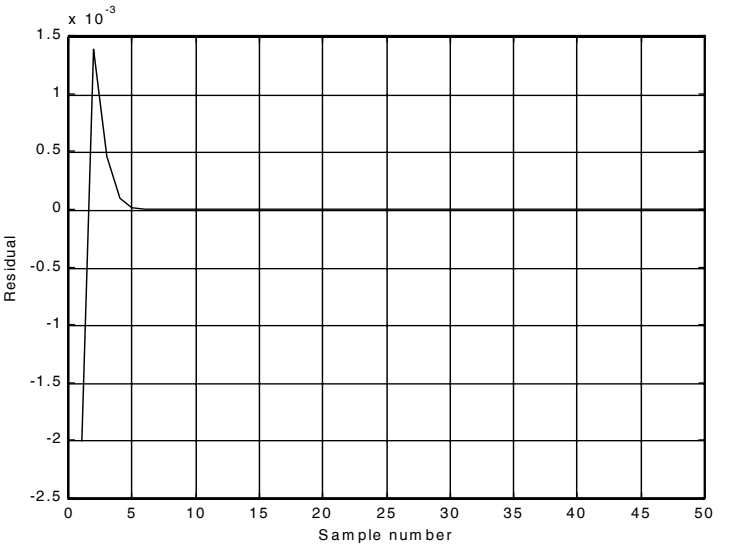

Figure 9.26b. Position residual (error). Large gain coefficients. No noise. The error settles down quickly.

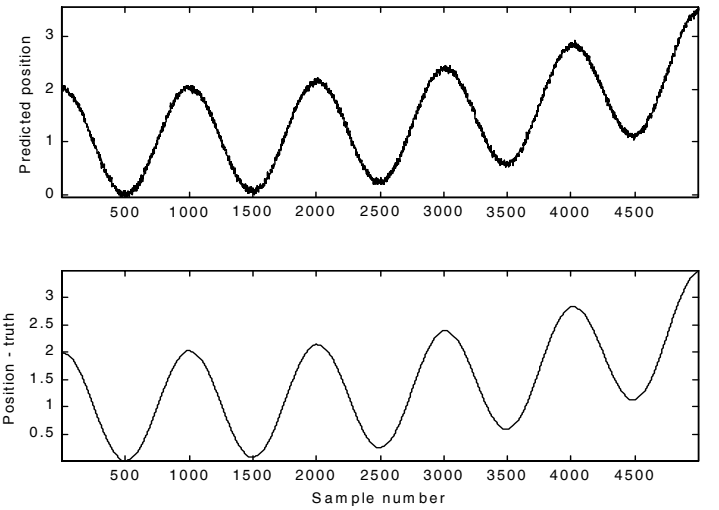

Figure 9.27a. Predicted and true position. $\xi = 0.8$ (i.e., small gain coefficients). Noise is present.

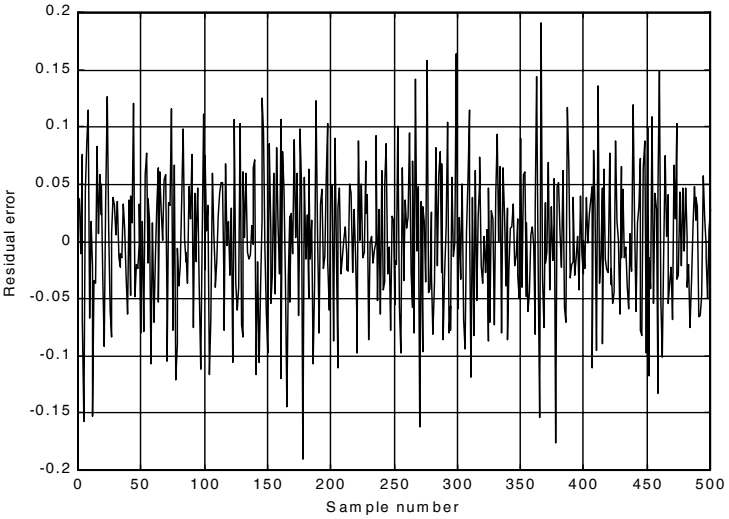

Figure 9.27b. Position residual (error). Small gain coefficients. Noise present. The error stays fairly large; however, its average is around zero. The variation is due to noise.

9.9. 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. Fig. 9.28 shows a block diagram for the Kalman filter.

The Kalman filter equations can be deduced from Fig. 9.28. The filtering equation is

$$\underline{x}(n|n) = \underline{x}_s(n) = \underline{x}(n|n-1) + K(n)[\underline{y}(n) - \underline{G}\underline{x}(n|n-1)] \quad \textbf{(9.127)}$$

The measurement vector is

$$\underline{y}(n) = \underline{G}\underline{x}(n) + \underline{v}(n) \quad \textbf{(9.128)}$$

where $\underline{v}(n)$ is zero mean, white Gaussian noise with covariance $\Re_c$,

$$\Re_c = E\{\underline{y}(n)\ \underline{y}^t(n)\} \quad \textbf{(9.129)}$$

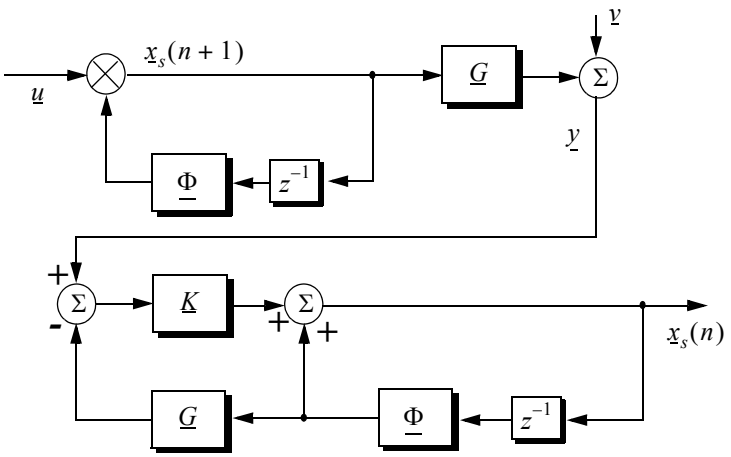

Figure 9.28. Structure of the Kalman filter.

The gain (weight) vector is dynamically computed as

$$\underline{K}(n) = \underline{P}(n|n-1)\underline{G}^t[\underline{G}\underline{P}(n|n-1)\underline{G}^t + \Re_c]^{-1} \quad \textbf{(9.130)}$$

where the measurement noise matrix $\underline{P}$ represents the predictor covariance matrix, and is equal to

$$\underline{P}(n+1|n) = E\{\underline{x}_s(n+1)\underline{x}^*_s(n)\} = \underline{\Phi}\underline{P}(n|n)\underline{\Phi}^t + \underline{Q} \quad \textbf{(9.131)}$$

where $\underline{Q}$ is the covariance matrix for the input $\underline{u}$,

$$\underline{Q} = E\{\underline{u}(n)\ \underline{u}^t(n)\} \quad \textbf{(9.132)}$$

The corrector equation (covariance of the smoothed estimate) is

$$\underline{P}(n|n) = [\underline{I} - \underline{K}(n)\underline{G}]\underline{P}(n|n-1) \tag{9.133}$$

Finally, the predictor equation is

$$\underline{x}(n+1|n) = \underline{\Phi}\underline{x}(n|n) \tag{9.134}$$

9.9.1. 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

$$E\{\ddot{x}(t)\ \ddot{x}(t+t_1)\} = \sigma_a^2\ e^{-\frac{|t_1|}{\tau_m}} \tag{9.135}$$

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

$$\ddot{x}(n+1) = \rho_m\ \ddot{x}(n) + \sqrt{1-\rho_m^2}\ \sigma_m\ w(n) \tag{9.136}$$

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

$$\rho_m = e^{-\frac{T}{\tau_m}} \tag{9.137}$$

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

$$\frac{d}{dt}v(t) = -\beta_m v(t) + w(t) \tag{9.138}$$

---

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.

where β_m is equal to $1/\tau_m$. The maneuvering variance using Singer's model is given by

$$\sigma_m^2 = \frac{A_{max}^2}{3}[1 + 4P_{max} - P_0] \tag{9.139}$$

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

$$\underline{\Phi} = \begin{bmatrix} 1 & T & \frac{1}{\beta_m^2}(-1 + \beta_m T + \rho_m) \\ 0 & 1 & \frac{1}{\beta_m}(1 - \rho_m) \\ 0 & 0 & \rho_m \end{bmatrix} \tag{9.140}$$

Note that when $T\beta_m = T/\tau_m$ is small (the target has constant acceleration), then Eq. (9.140) reduces to Eq. (9.114). Typically, the sampling interval T is much less than the maneuvering time constant τ_m; hence, Eq. (9.140) can be accurately replaced by its second order approximation. More precisely,

$$\underline{\Phi} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T(1 - T/2\tau_m) \\ 0 & 0 & \rho_m \end{bmatrix} \tag{9.141}$$

The covariance matrix was derived by Singer, and it is equal to

$$\underline{C} = \frac{2\sigma_m^2}{\tau_m} \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{9.142}$$

where

$$C_{11} = \sigma_x^2 = \frac{1}{2\beta_m^5}\left[1 - e^{-2\beta_m T} + 2\beta_m T + \frac{2\beta_m^3 T^3}{3} - 2\beta_m^2 T^2 - 4\beta_m T e^{-\beta_m T}\right] \tag{9.143}$$

$$C_{12} = C_{21} = \frac{1}{2\beta_m^4}[e^{-2\beta_m T} + 1 - 2e^{-\beta_m T} + 2\beta_m T e^{-\beta_m T} - 2\beta_m T + \beta_m^2 T^2] \tag{9.144}$$

$$C_{13} = C_{31} = \frac{1}{2\beta_m^3}[1 - e^{-2\beta_m T} - 2\beta_m T e^{-\beta_m T}] \tag{9.145}$$

$$C_{22} = \frac{1}{2\beta_m^3}[4e^{-\beta_m T} - 3 - e^{-2\beta_m T} + 2\beta_m T] \tag{9.146}$$

$$C_{23} = C_{32} = \frac{1}{2\beta_m^2}[e^{-2\beta_m T} + 1 - 2e^{-\beta_m T}] \tag{9.147}$$

$$C_{33} = \frac{1}{2\beta_m}[1 - e^{-2\beta_m T}] \tag{9.148}$$

Two limiting cases are of interest:

1. *The short sampling interval case ($T \ll \tau_m$),*

$$\lim_{\beta_m T \to 0} \underline{C} = \frac{2\sigma_m^2}{\tau_m} \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix} \tag{9.149}$$

and the state transition matrix is computed from Eq. (9.141) as

$$\lim_{\beta_m T \to 0} \underline{\Phi} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \tag{9.150}$$

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*

$$\lim_{\beta_m T \to \infty} \underline{C} = \sigma_m^2 \begin{bmatrix} \frac{2T^3\tau_m}{3} & T^2\tau_m & \tau_m^2 \\ T^2\tau_m & 2T\tau_m & \tau_m \\ \tau_m^2 & \tau_m & 1 \end{bmatrix} \tag{9.151}$$

$$\lim_{\beta_m T \to \infty} \underline{\Phi} = \begin{bmatrix} 1 & T & T\tau_m \\ 0 & 1 & \tau_m \\ 0 & 0 & 0 \end{bmatrix} \tag{9.152}$$

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,

$$\underline{C} = 2\sigma_m^2 \tau_m \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \tag{9.153}$$

$$\underline{\Phi} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{9.154}$$

9.9.2. 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. (9.127). Thus,

$$\begin{bmatrix} x(n|n) \\ \dot{x}(n|n) \\ \ddot{x}(n|n) \end{bmatrix} = \begin{bmatrix} x(n|n-1) \\ \dot{x}(n|n-1) \\ \ddot{x}(n|n-1) \end{bmatrix} + \begin{bmatrix} k_1(n) \\ k_2(n) \\ k_3(n) \end{bmatrix} [x_0(n) - x(n|n-1)] \tag{9.155}$$

with (see Fig. 9.21)

$$x(n|n-1) = x_s(n-1) + T\ \dot{x}_s(n-1) + \frac{T^2}{2}\ \ddot{x}_s(n-1) \tag{9.156}$$

$$\dot{x}(n|n-1) = \dot{x}_s(n-1) + T\ \ddot{x}_s(n-1) \tag{9.157}$$

$$\ddot{x}(n|n-1) = \ddot{x}_s(n-1) \tag{9.158}$$

Comparing the previous three equations with the $\alpha\beta\gamma$ filter equations yields

$$\begin{bmatrix} \alpha \\ \underline{\beta} \\ T \\ \underline{\gamma} \\ T^2 \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} \tag{9.159}$$

Additionally, the covariance matrix elements are related to the gain coefficients by

$$\begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = \frac{1}{C_{11} + \sigma_v^2} \begin{bmatrix} C_{11} \\ C_{12} \\ C_{13} \end{bmatrix} \tag{9.160}$$

Eq. (9.160) 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. It is given in Listing 9.4 in Section 9.11. 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. (9-129)*	*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. 9.22 and 9.23. Figs. 9.29 and 9.30 show the residual error and predicted position corresponding to Figs. 9.22 and 9.23. These plots can be reproduced using the program *"fig9_28.m"* given in Listing 9.5 in Section 9.11.

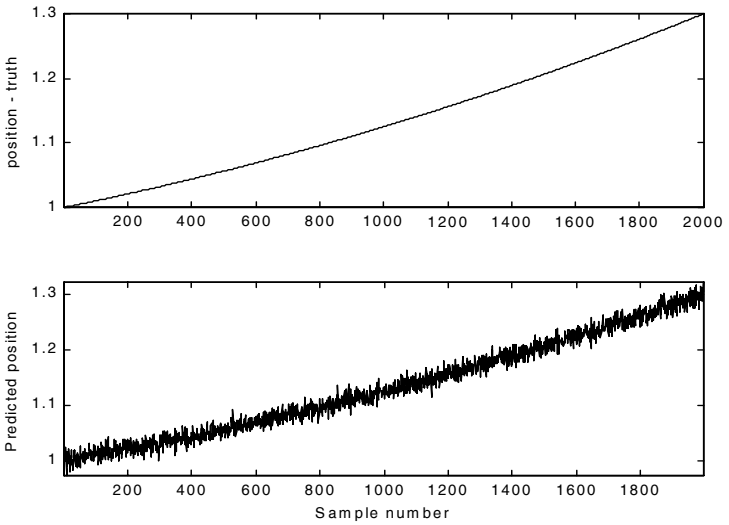

Figure 9.29a. True and predicted positions. Lazy maneuvering. Plot produced using the function *"kalman_filter.m"*.

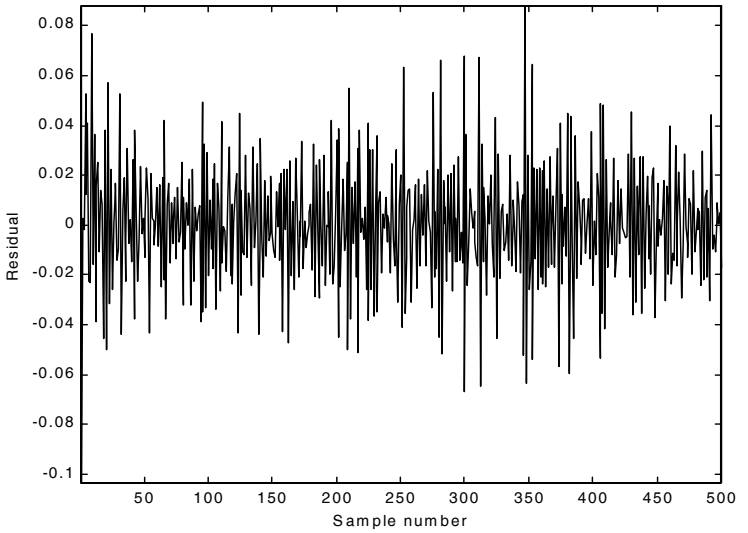

Figure 9.29b. Residual corresponding to Fig. 9.29a.

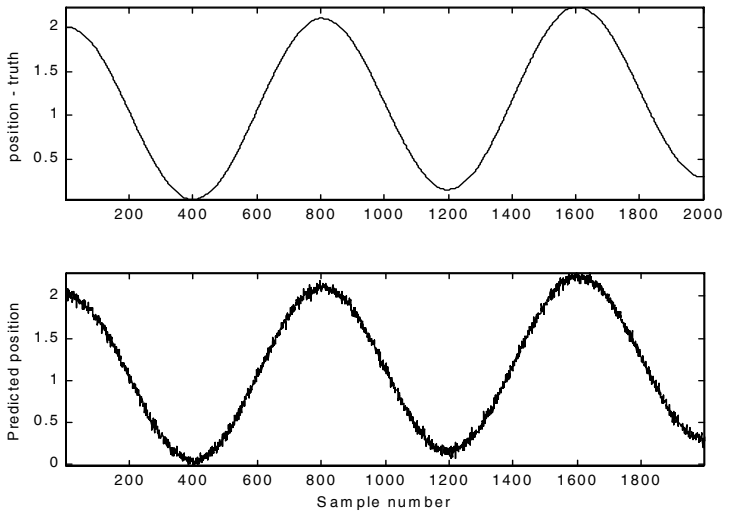

Figure 9.30a. True and predicted positions. Aggressive maneuvering. Plot produced using the function *"kalman_filter.m"*.

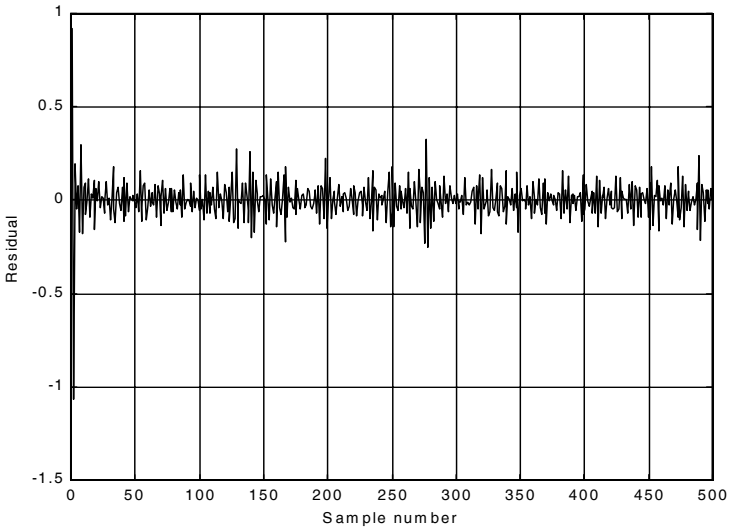

Figure 9.30b. Residual corresponding to Fig. 9.30a.

9.10. "MyRadar" Design Case Study - Visit 9

9.10.1.Problem Statement

Implement a Kalman filter tracker into the "MyRadar" design case study.

9.10.2. A Design[1]

For this purpose, the MATLAB GUI workspace entitled *"kalman_gui.m"* was developed. It is shown in Fig. 9.31. In this design, the inputs can be initialized to correspond to either target type (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 trajectory for the selected target type. This is accomplished using the function *"maketraj.m"*. It is given in Listing 9.6 in Section 9.11. The user can either use this program, or import their own specific trajectory. The function *"maketraj.m"* assumes constant altitude, and generates a manuevering trajectory in the *x*-*y* plane, as shown in Fig. 9.32. This trajectory can be changed using the different fields in the *"trajectory Parameter"* fields.

Next the program corrupts the trajectory by adding white Guassian noise to it. This is accomplished by the function *"addnoise.m"* which is given in Listing 9.7 in Section 9.11. A six-state Kalman filter named *"kalfilt.m"* is then utilized to perform the tracking task. This function is given in Listing 9.8.

The azimuth, elevation, and range errors are input to the program using their corresponding fields on the GUI. In this example, these entries are assumed constant throughout the simulation. In practice, this is not true and these values will change. They are caluclated 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

$$\sigma_R = \frac{\Delta R}{\sqrt{2 \times SNR}} = \frac{c}{2B\sqrt{2 \times SNR}} \qquad \textbf{(9.161)}$$

1. The MATLAB code in this section was developed by Mr. David Hall, Consultant to Decibel Research, Inc., Huntsville, Alabama.

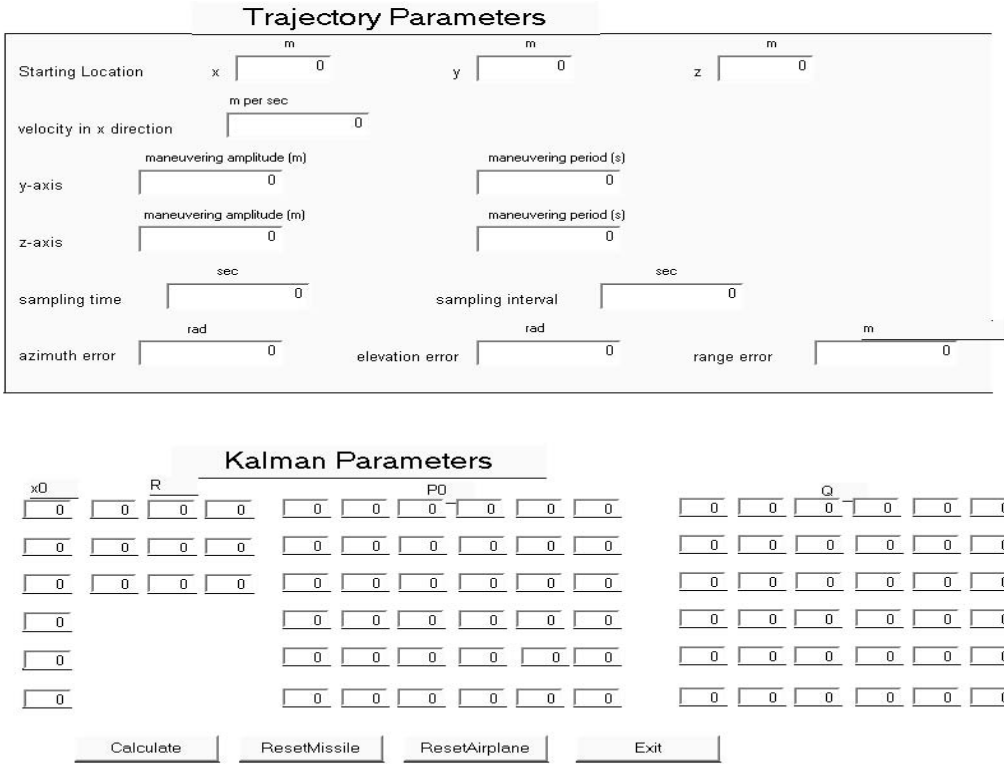

Figure 9.31. MATLAB GUI workspace associated with the *"MyRadar"* design case study- visit 9.

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

$$\sigma_v = \frac{\lambda}{2\tau\sqrt{2 \times SNR}} \tag{9.162}$$

where λ is the wavelength and τ is the uncompressed pulsewidth. The standard deviation of the error in the angle measurement is

$$\sigma_a = \frac{\Theta}{1.6\sqrt{2 \times SNR}} \tag{9.163}$$

where Θ is the antenna beamwidth of the angular coordinate of the measurement (azimuth and elevation).

In this example, the radar is located at $(x, y, z) = (0, 0, 0)$. This simulation calculates and plots the following outputs:

TABLE 9.1. Output list generated by the *"kalman_gui.m"* simulation

Figure #	Description
9.32	*uncorrupted input trajectory*
9.33	*corrupted input trajectory*
9.34	*corrupted and uncorrupted x-position*
9.35	*corrupted and uncorrupted y-position*
9.36	*corrupted and uncorrupted z-position*
9.37	*corrupted and filtered x-, y- and z-positions*
9.38	*predicted x-, y-, and z- velocities*
9.39	*position residuals*
9.40	*velocity residuals*
9.41	*covariance matrix components versus time*
9.42	*Kalman filter gains versus time*

Fig. 9.32 through Fig. 9.42 shows typical outputs produced using this simulation for the missile.

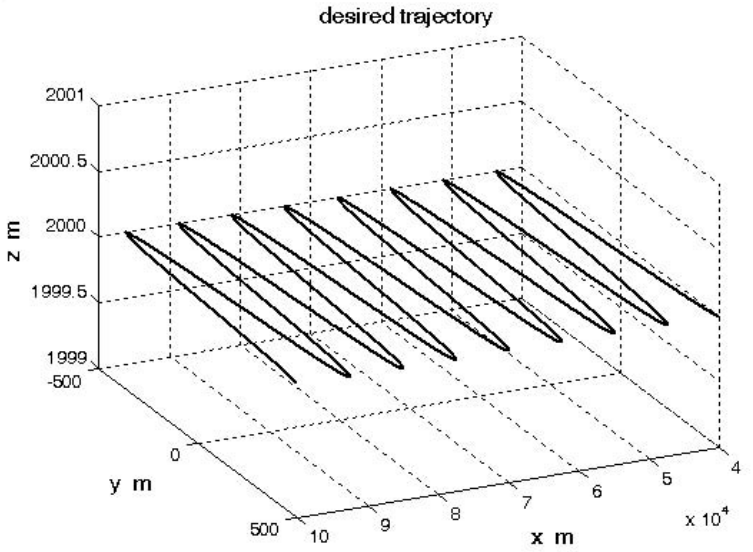

Figure 9.32. Missile uncorrupted trajectory.

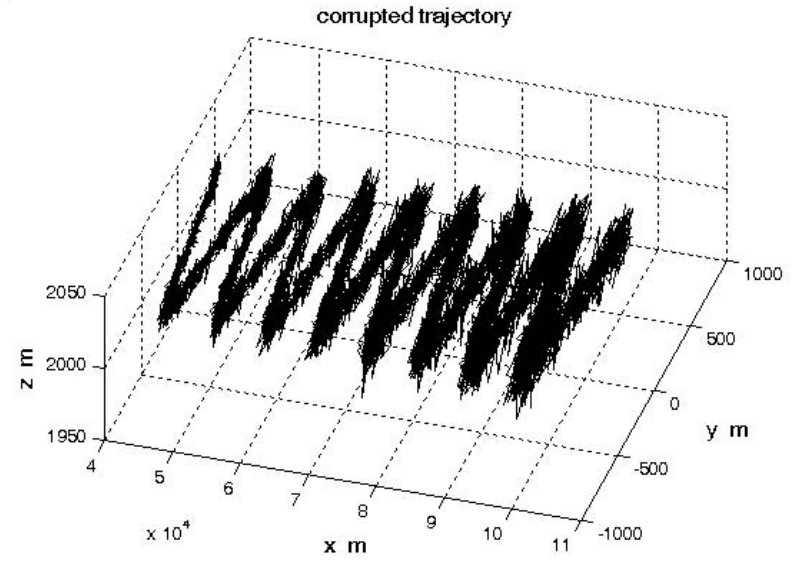

Figure 9.33. Missile corrupted trajectory.

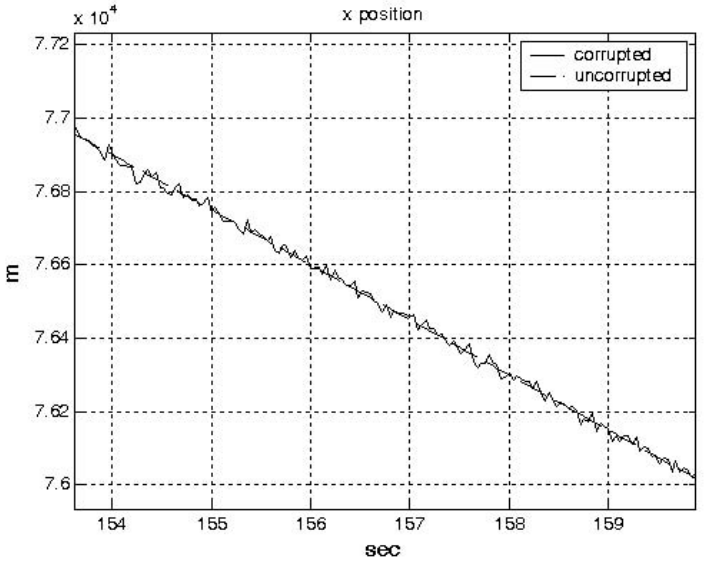

Figure 9.34. Missile x-position from 153 to 160 seconds.

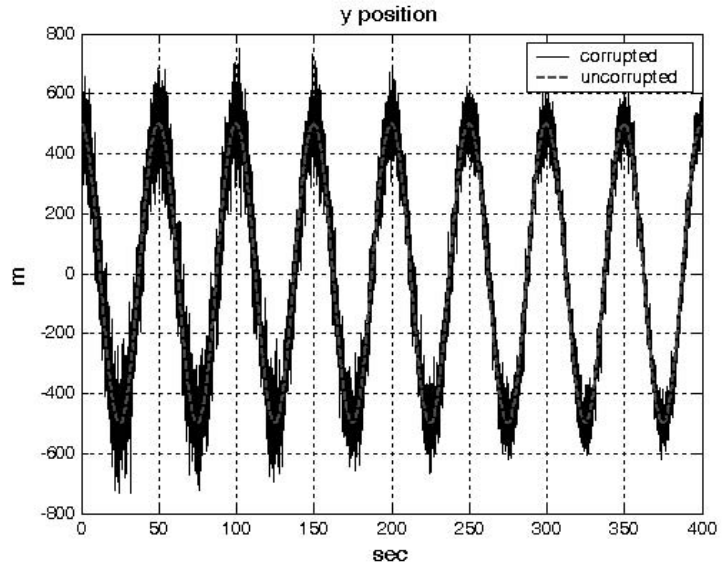

Figure 9.35. Missile y-position.

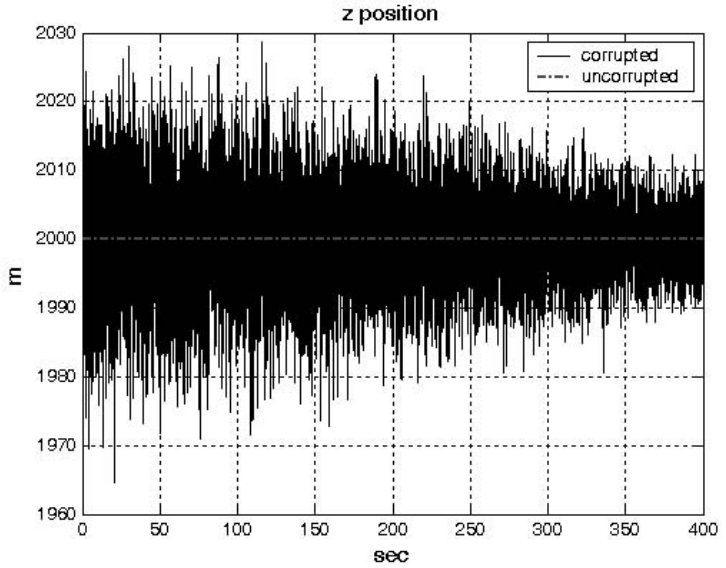

Figure 9.36. Missile z-position.

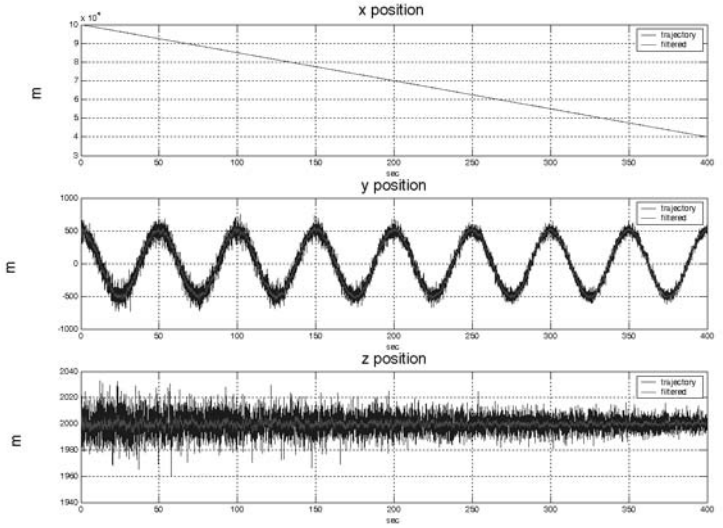

Figure 9.37. Missile trajectory and filtered trajectory.

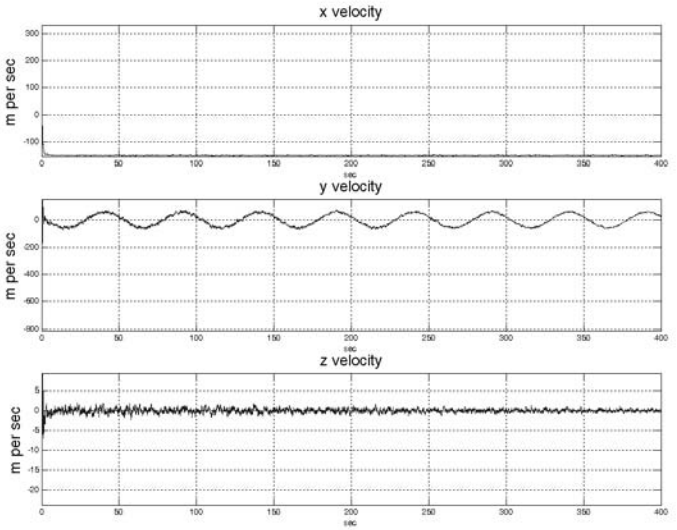

Figure 9.38. Missile velocity filtered.

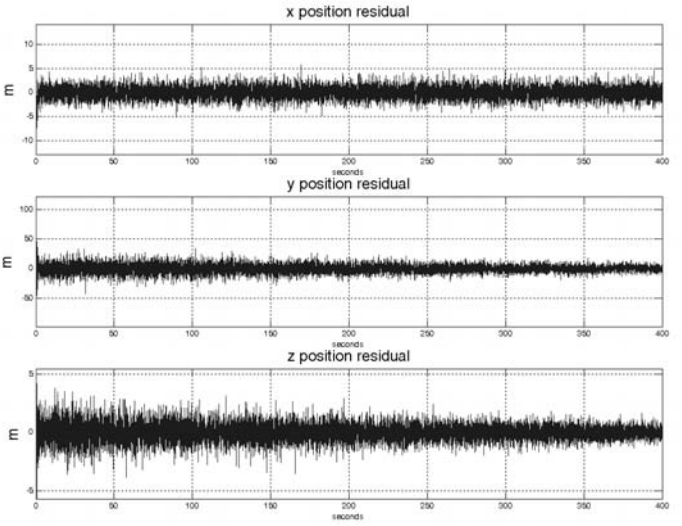

Figure 9.39. Missile position residuals.

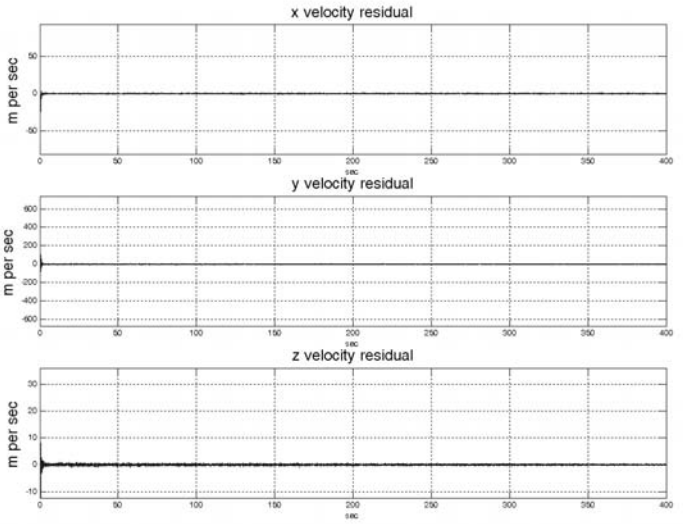

Figure 9.40. Missile velocity residuals.

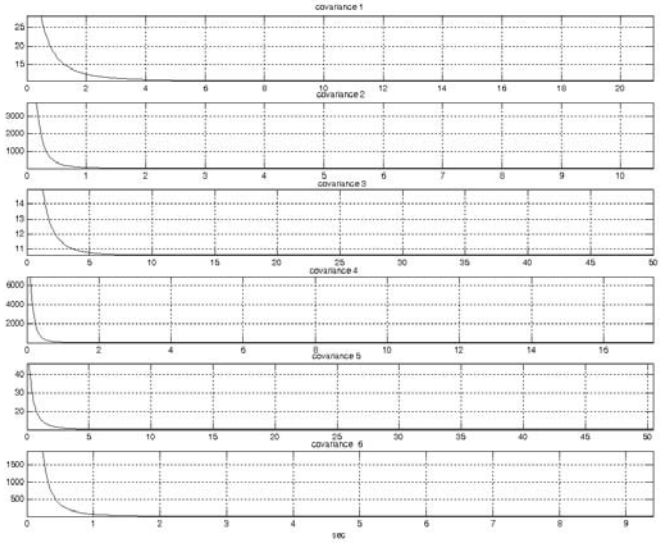

Figure 9.41. Missile covariance matrix components versus time.

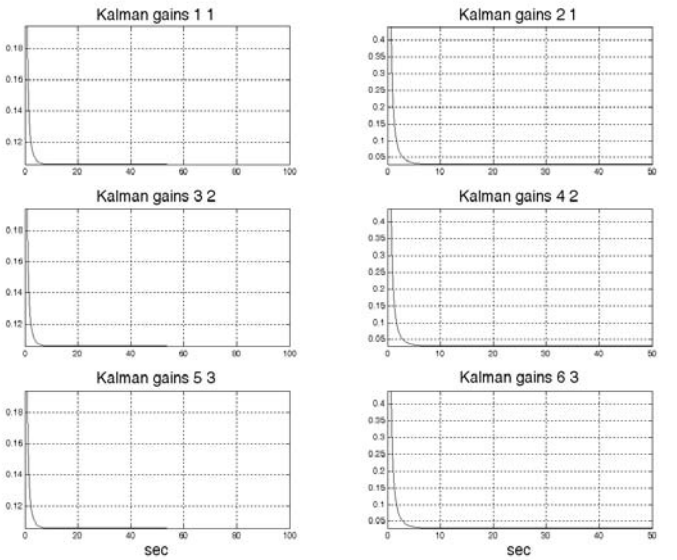

Figure 9.42. Kalman filter gains versus time.

9.11. MATLAB Program and Function Listings

This section contains listings of all MATLAB programs and functions used in this chapter. Users are encouraged to rerun this code with different inputs in order to enhance their understanding of the theory.

Listing 9.1. MATLAB Function "mono_pulse.m"

```
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')
```

Listing 9.2. MATLAB Function "ghk_tracker.m"

```
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"

```
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
```

Listing 9.3. MATLAB Program "fig9_21.m"

```
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')
```

Listing 9.4. MATLAB Function "kalman_filter.m"

```
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
```

Listing 9.5. MATLAB Program "fig9_28.m"

```
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 measurement 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)
xlabel ('Sample number')
ylabel ('Residual')
figure(2)
subplot(2,1,1)
plot(inp)
axis tight
ylabel ('position - truth')
subplot(2,1,2)
plot(estimate)
axis tight
xlabel ('Sample number')
ylabel ('Predicted position')
```

Listing 9.6. MATLAB Function "maketraj.m"

```
function [times , trajectory] = maketraj(start_loc, xvelocity, yamp, yperiod,
zamp, zperiod, samplingtime, deltat)
% maketraj.m
% by David J. Hall
% for Bassem Mahafza
% 17 June 2003
% 17:01
% USAGE:  [times , trajectory] = maketraj(start_loc, xvelocity, yamp, yperiod,
zamp, zperiod, samplingtime, 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] ;
```

Listing 9.7. MATLAB Function "addnoise.m"

```
function [noisytraj ] = addnoise(trajectory, sigmaaz, sigmael, sigmarange )
% addnoise.m
% by David J. Hall
% for Bassem Mahafza
% 10 June 2003
% 11:46
% 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
```

Listing 9.8. MATLAB Function "kalfilt.m"

```
function [filtered, residuals , covariances, kalmgains] = kalfilt(trajectory, x0,
P0, phi, R, Q )
% kalfilt.m
% by David J. Hall
% for Bassem Mahafza
% 10 June 2003
% 11:46
% 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 10 *Electronic Countermeasures (ECM)*

This chapter is coauthored with J. Michael Madewell[1]

10.1. Introduction

Any deliberate electronic effort intended to disturb normal radar operation is usually referred to as an Electronic Countermeasure (ECM). This may also include chaff, radar decoys, radar RCS alterations (e.g., radio frequency absorbing materials), and, of course, radar jamming.

In general, ECM is used by the offense to accomplish one, several, or possibly all of the following objectives: (1) deny proper target detection; (2) generate operator confusion and / or deception; (3) force delays in detection and tracking initiation; (4) generate false tracks of non-real targets; (5) overload the radar computer with an excessive number of targets; (6) deny accurate measurements of the target range and range rate; (7) force dropped tracks; and (8) introduce errors in target position and range rate. Alternatively, the defense may utilize Electronic counter-countermeasures (ECCM) to overcome and mitigate the effects of ECM on the radar. When deployed properly, ECCM techniques and / or hardware can have the following effects: (1) prevent receiver saturation; (2) maintain a reasonable CFAR rate; (3) enhance the signal to jammer ratio; (4) properly identify and discriminate directional interference; (5) reject invalid targets; and (6) maintain true target tracks.

ECM techniques can be exploited by a radar system in many different ways and can be categorized into two classes:

1. Mr. J. Michael Madewell is with the US Army Space and Missile Defense Command in Huntsville, Alabama.

1. Denial ECM techniques: Denial ECM techniques can be either active or passive. Active denial ECM techniques include: CW, short pulse, long pulse, spot noise, barrage noise, and sidelobe repeaters. Passive ECM techniques include chaff and Radar Absorbing Material (RAM).
2. Deception ECM techniques: Deception ECM techniques are also broken down into active and passive techniques. Active deception ECM techniques include repeater jammers and false target generators. Passive deception ECM include chaff and RAM.

10.2. Jammers

Jammers can be categorized into two general types: (1) barrage jammers and (2) deceptive jammers (repeaters). When strong jamming is present, detection capability is determined by receiver signal-to-noise plus interference ratio rather than SNR. In fact, in most cases, detection is established based on the signal-to-interference ratio alone.

Barrage jammers attempt to increase the noise level across the entire radar operating bandwidth. Consequently, this lowers the receiver SNR, and, in turn, makes it difficult to detect the desired targets. This is the reason why barrage jammers are often called maskers (since they mask the target returns). Barrage jammers can be deployed in the main beam or in the sidelobes of the radar antenna. If a barrage jammer is located in the radar main beam, it can take advantage of the antenna maximum gain to amplify the broadcasted noise signal. Alternatively, sidelobe barrage jammers must either use more power, or operate at a much shorter range than main beam jammers. Main beam barrage jammers can be deployed either on-board the attacking vehicle, or act as an escort to the target. Sidelobe jammers are often deployed to interfere with a specific radar, and since they do not stay close to the target, they have a wide variety of stand-off deployment options.

Repeater jammers carry receiving devices on board in order to analyze the radar's transmission, and then send back false target-like signals in order to confuse the radar. There are two common types of repeater jammers: spot noise repeaters and deceptive repeaters. The spot noise repeater measures the transmitted radar signal bandwidth and then jams only a specific range of frequencies. The deceptive repeater sends back altered signals that make the target appear in some false position (ghosts). These ghosts may appear at different ranges or angles than the actual target. Furthermore, there may be several ghosts created by a single jammer. By not having to jam the entire radar bandwidth, repeater jammers are able to make more efficient use of their jamming power. Radar frequency agility may be the only way possible to defeat spot noise repeaters.

In general a jammer can be identified by its effective operating bandwidth B_J and by its Effective Radiated Power (ERP), which is proportional to the jammer transmitter power P_J. More precisely,

$$ERP = \frac{P_J G_J}{L_J} \tag{10.1}$$

where G_J is the jammer antenna gain and L_J is the total jammer losses. The effect of a jammer on a radar is measured by the Signal-to-Jammer ratio (S/J).

10.2.1. Self-Screening Jammers (SSJ)

Self-screening jammers, also known as self-protecting jammers and as main beam jammers, are a class of ECM systems carried on the vehicle they are protecting. Escort jammers (carried on vehicles that accompany the attacking vehicles) can also be treated as SSJs if they appear at the same range as that of the target(s).

Assume a radar with an antenna gain G, wavelength λ, aperture A_r, bandwidth B_r, receiver losses L, and peak power P_t. The single pulse power received by the radar from a target of RCS σ, at range R, is

$$S = \frac{P_t G^2 \lambda^2 \sigma \tau}{(4\pi)^3 R^4 L} \tag{10.2}$$

τ is the radar pulsewidth. The power received by the radar from an SSJ jammer at the same range is

$$J = \frac{P_J G_J}{4\pi R^2} \frac{A_r}{B_J L_J} \tag{10.3}$$

where P_J, G_J, B_J, L_J are, respectively, the jammer's peak power, antenna gain, operating bandwidth, and losses. Using the relation

$$A_r = \frac{\lambda^2 G}{4\pi} \tag{10.4}$$

then Eq. (10.3) can be written as

$$J = \frac{P_J G_J}{4\pi R^2} \frac{\lambda^2 G}{4\pi} \frac{1}{B_J L_J} \tag{10.5}$$

Note that $B_J > B_r$. This is needed in order to compensate for the fact that the jammer bandwidth is usually larger than the operating bandwidth of the radar. Jammers are normally designed to operate against a wide variety of radar systems with different bandwidths.

Substituting Eq. (10.1) into Eq. (10.5) yields,

$$J = ERP \frac{\lambda^2 G}{(4\pi)^2 R^2} \frac{1}{B_J} \tag{10.6}$$

Thus, S/J ratio for a SSJ case is obtained from Eqs. (10.6) and (10.2),

$$\frac{S}{J} = \frac{P_t \tau G \sigma B_J}{(ERP)(4\pi) R^2 L} \tag{10.7}$$

and when pulse compression is used, with time-bandwidth-product G_{PC}, then Eq. (10.7) can be written as

$$\frac{S}{J} = \frac{P_t G \sigma B_J G_{PC}}{(ERP)(4\pi) R^2 B_r L} \tag{10.8}$$

Note that to obtain Eq. (10.8), one must multiply Eq. (10.7) by the factor B_r / B_r and use the fact that $G_{PC} = B_r \tau$.

The jamming power reaches the radar on a one-way transmission basis, whereas the target echoes involve two-way transmission. Thus, the jamming power is generally greater than the target signal power. In other words, the ratio S/J is less than unity. However, as the target becomes closer to the radar, there will be a certain range such that the ratio S/J is equal to unity. This range is known as the cross-over range. The range window where the ratio S/J is sufficiently larger than unity is denoted as the detection range. In order to compute the crossover range R_{co}, set S/J to unity in Eq. (10.8) and solve for range. It follows that

$$(R_{CO})_{SSJ} = \left(\frac{P_t G \sigma B_J}{4\pi B_r L (ERP)} \right)^{1/2} \tag{10.9}$$

MATLAB Program "ssj_req.m"

The program *"ssj_req.m"* implements Eqs. (10.9); it is given in Listing 10.1 in Section 10.5. This program calculates the cross-over range and generates plots of relative S and J versus range normalized to the cross-over range, as illustrated in Fig. 10.1a.

In this example, the following parameters were utilized: radar peak power $P_t = 50KW$, jammer peak power $P_J = 200W$, radar operating bandwidth $B_r = 667KHz$, jammer bandwidth $B_J = 50MHz$, radar and jammer losses $L = L_J = 0.10dB$, target cross section $\sigma = 10.m^2$, radar antenna gain $G = 35dB$, jammer antenna gain $G_J = 10dB$, the radar operating frequency is $f = 5.6GHz$. The syntax is as follows:

[BR_range] = ssj_req (pt, g, freq, sigma, br, loss, pj, bj, gj, lossj)

where

Symbol	Description	Units	Status
pt	*radar peak power*	*W*	*input*
g	*radar antenna gain*	*dB*	*input*
freq	*radar operating frequency*	*Hz*	*input*
sigma	*target cross section*	m^2	*input*
br	*radar operating bandwidth*	*Hz*	*input*
loss	*radar losses*	*dB*	*input*
pj	*jammer peak power*	*W*	*input*
bj	*jammer bandwidth*	*Hz*	*input*
gj	*jammer antenna gain*	*dB*	*input*
lossj	*jammer losses*	*dB*	*input*
BR_range	*cross-over range*	*Km*	*output*

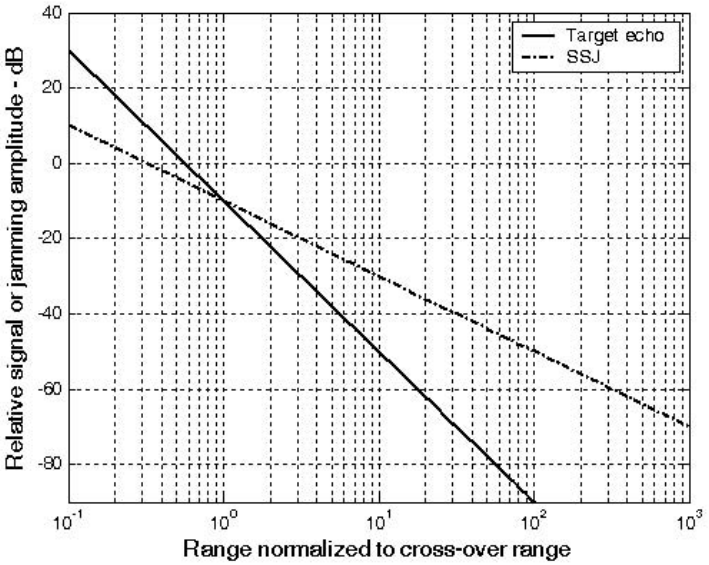

Figure 10.1a. Target and jammer echo signals. Plots were generated using the program *"ssj_req.m"* and using the input parameters defined on the previous page.

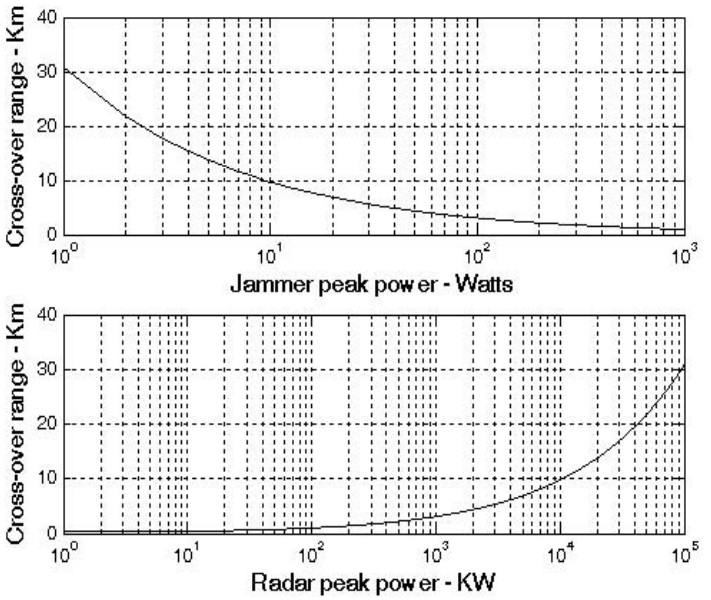

Figure 10.1b. Burn-through range versus jammer and radar peak powers corresponding to example used in generating Fig. 10.1a.

Burn-through Range

If jamming is employed in the form of Gaussian noise, then the radar receiver has to deal with the jamming signal the same way it deals with noise power in the radar. Thus, detection, tracking, and other functions of the radar signal and data processors are no longer dependent on the SNR. In this case, the S/(J+N) ratio must be calculated. More precisely,

$$\frac{S}{J+N} = \frac{\left(\frac{P_t G \sigma A_r \tau}{(4\pi)^2 R^4 L}\right)}{\left(\frac{(ERP)A_r}{4\pi R^2 B_J} + kT_0\right)} \qquad \textbf{(10.10)}$$

where k is Boltzman's constant and T_0 is the effective noise temperature.

The S/(J+N) ratio should be used in place of the SNR when calculating the the radar equation and when computing the probability of detection. Furthermore, S/(J+N) must also be used in place of the SNR when using coherent or non-coherent pulse integration.

The range at which the radar can detect and perform proper measurements for a given S/(J+N) value is defined as the burn-through range. It is given by

$$R_{BT} = \left\{ \sqrt{\left(\frac{(ERP)A_r}{8\pi B_j k T_0}\right)^2 + \frac{P_t G \sigma A_r \tau}{(4\pi)^2 L \frac{S}{(J+N)} k T_0}} - \frac{(ERP)A_r}{8\pi B_j k T_0} \right\}^{\frac{1}{2}} \quad \textbf{(10.11)}$$

MATLAB Function "sir.m"

The MATLAB function *"sir.m"* implements Eq. (10.10). It generates plots of the S/(J+N) versus detection range and plots of the burn-through range versus the jammer ERP. It is given in Listing 10.2 in Section 10.5. The syntax is as follows:

[SIR] = sir (pt, g, sigma, freq, tau,T0, loss, R, pj, bj, gj, lossj)

where

Symbol	Description	Units	Status
pt	*radar peak power*	*W*	*input*
g	*radar antenna gain*	*dB*	*input*
sigma	*target cross section*	m^2	*input*
freq	*radar operating frequency*	*Hz*	*input*
tau	*radar pulsewidth*	*seconds*	*input*
T0	*effective noise temperature*	*Kelvin*	*input*
loss	*radar losses*	*dB*	*input*
R	*range. can be single value or a vector*	*Km*	*input*
pj	*jammer peak power*	*W*	*input*
bj	*jammer bandwidth*	*Hz*	*input*
gj	*jammer antenna gain*	*dB*	*input*
lossj	*jammer losses*	*dB*	*input*
SIR	*S/(J+N)*	*dB*	*output*

Fig. 10.2 shows some typical outputs generated by this function when the inputs are as follows:

Input Parameter	Value
pt	*50KW*
g	*35 dB*
sigma	*10 square meters*
freq	*5.6 GHz*

Input Parameter	Value
tau	*50 micro-seconds*
T0	*290*
loss	*5 dB*
R	*linspace(10,400,5000) Km*
pj	*200 Watts*
bj	*50 MHz*
gj	*10 dB*
lossj	*0.3 dB*

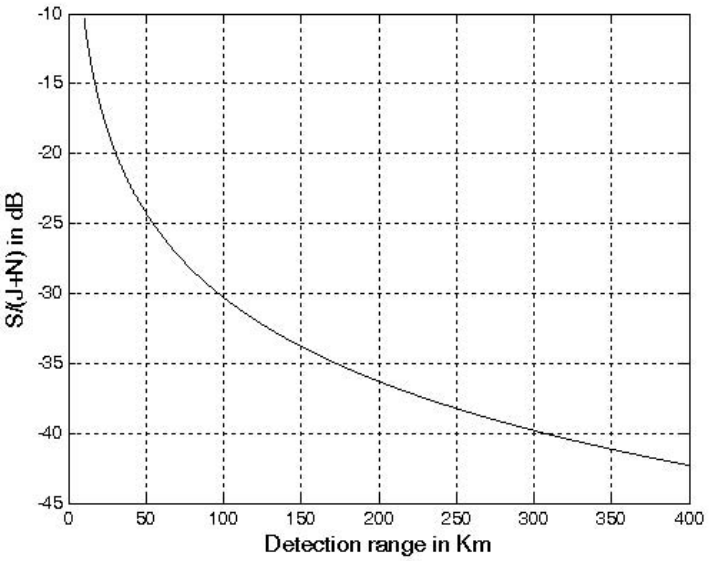

Figure 10.2. S/(J+N) versus detection range.

MATLAB Function "burn_thru.m"

The MATLAB function *"burn_thru.m"* implements Eq. (10.10) and (10.11). It generates plots of the S/(J+N) versus detection range and plots of the burn-through range versus the jammer ERP. It is given in Listing 10.3 in Section 10.5. The syntax is as follows:

[Range] = burn_thru (pt, g, sigma, freq, tau, T0, loss, pj, bj, gj, lossj, sir0, ERP)

where

Symbol	Description	Units	Status
pt	*radar peak power*	*W*	*input*
g	*radar antenna gain*	*dB*	*input*
sigma	*target cross section*	m^2	*input*
freq	*radar operating frequency*	*Hz*	*input*
tau	*radar pulsewidth*	*seconds*	*input*
T0	*effective noise temperature*	*Kelvin*	*input*
loss	*radar losses*	*dB*	*input*
pj	*jammer peak power*	*W*	*input*
bj	*jammer bandwidth*	*Hz*	*input*
gj	*jammer antenna gain*	*dB*	*input*
lossj	*jammer losses*	*dB*	*input*
sir0	*desired SIR*	*dB*	*input*
ERP	*desired ERP. can be a vector*	*Watts*	*input*
Range	*burn-through range*	*Km*	*output*

Fig. 10.3 shows some typical outputs generated by this function when the inputs are as follows:

Input Parameter	Value
pt	*50KW*
g	*35 dB*
sigma	*10 square meters*
freq	*5.6 GHz*
tau	*0.5 milli-seconds*
T0	*290*
loss	*5 dB*
pj	*200 Watts*
bj	*500 MHz*
gj	*10 dB*
lossj	*0.3 dB*
sir0	*15dB*
ERP	*linspace(1, 1000, 1000) W*

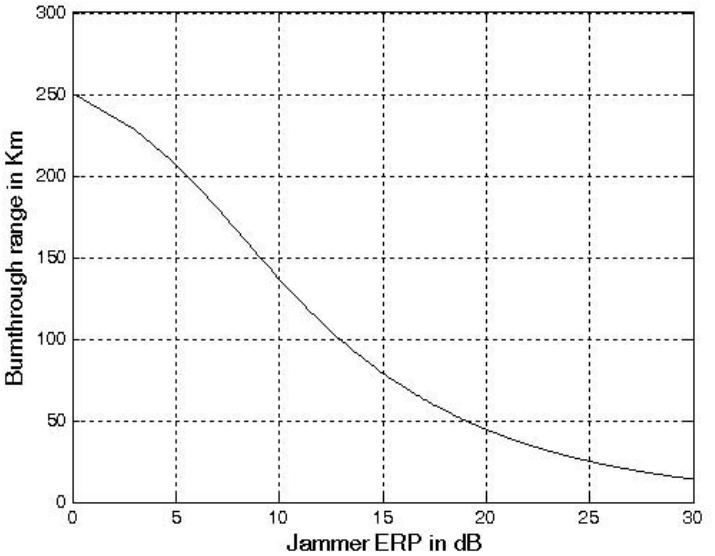

Figure 10.3. Burn-through range versus ERP. (S/(J+N) = 15 dB.

10.2.2. Stand-Off Jammers (SOJ)

Stand-off jammers (SOJ) emit ECM signals from long ranges which are beyond the defense's lethal capability. The power received by the radar from an SOJ jammer at range R_J is

$$J = \frac{P_J G_J}{4\pi R_J^2} \frac{\lambda^2 G'}{4\pi} \frac{1}{B_J L_J} = \frac{ERP}{4\pi R_J^2} \frac{\lambda^2 G'}{4\pi} \frac{1}{B_J} \tag{10.12}$$

where all terms in Eq. (10.12) are the same as those for the SSJ case except for G'. The gain term G' represents the radar antenna gain in the direction of the jammer and is normally considered to be the sidelobe gain.

The SOJ radar equation is then computed as

$$\frac{S}{J} = \frac{P_t \tau G^2 R_J^2 \sigma B_J}{4\pi (ERP) G' R^4 L} \tag{10.13}$$

and when pulse compression is used, with time-bandwidth-product G_{PC} then Eq. (10.13) can be written as

$$\frac{S}{J} = \frac{P_t G^2 R_J^2 \sigma B_J P_{PC}}{4\pi (ERP) G' R^4 B_r L} \tag{10.14}$$

Again, the cross-over range is that corresponding to $S = J$; it is given by

$$(R_{CO})_{SOJ} = \left(\frac{P_t G^2 R_J^2 \sigma B_J P_{PC}}{4\pi (ERP) G' B_r L} \right)^{1/4} \tag{10.15}$$

MATLAB Program "soj_req.m"

The program *"soj_req.m"* implements Eqs. (10.15); it is given in Listing 10.4 in Section 10.5. The inputs to the program *"soj_req.m"* are the same as in the SSJ case, with two additional inputs: the radar antenna gain on the jammer G' and radar-to-jammer range R_J. This program generates the same types of plots as in the case of the SSJ. Typical output is in Fig. 10.4 utilizing the same parameters as those in the SSJ case, with jammer peak power $P_J = 5000W$, jammer antenna gain $G_J = 30dB$, radar antenna gain on the jammer $G' = 10dB$, and radar to jammer range $R_J = 22.2Km$.

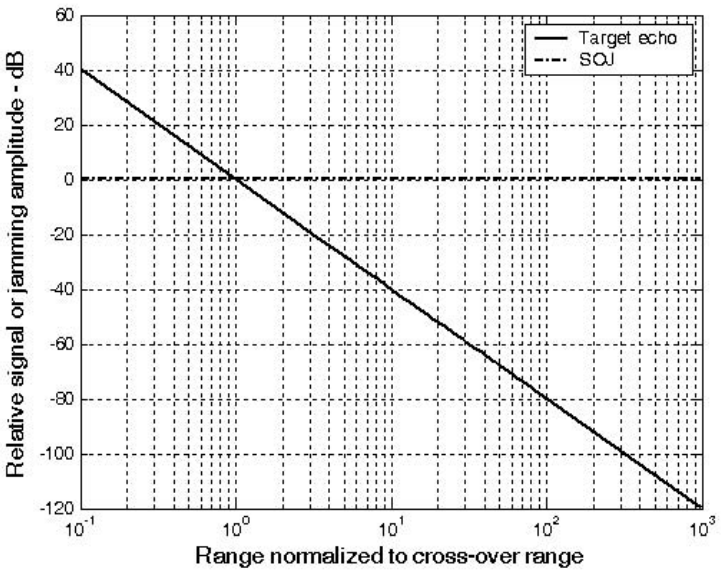

Figure 10.4. Target and jammer echo signals. Plots were generated using the program *"soj_req.m"*.

Again if the jamming is employed in the form of Gaussian noise, then the radar receiver has to deal with the jamming signal the same way it deals with noise power in the radar. In this case, the S/(J+N) is

$$\frac{S}{J+N} = \frac{\left(\frac{P_t G \sigma A_r \tau}{(4\pi)^2 R^4 L}\right)}{\left(\frac{(ERP)A_r G'}{4\pi R_J^2 B_J} + kT_0\right)} \tag{10.16}$$

10.3. Range Reduction Factor

Consider a radar system whose detection range R in the absence of jamming is governed by

$$(SNR)_o = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 k T_e B_r F L R^4} \tag{10.17}$$

The term Range Reduction Factor (RRF) refers to the reduction in the radar detection range due to jamming. More precisely, in the presence of jamming the effective radar detection range is

$$R_{dj} = R \times RRF \tag{10.18}$$

In order to compute RRF, consider a radar characterized by Eq. (10.17), and a barrage jammer whose output power spectral density is J_o (i.e., Gaussian-like). Then the amount of jammer power in the radar receiver is

$$J = kT_J B_r \tag{10.19}$$

where T_J is the jammer effective temperature. It follows that the total jammer plus noise power in the radar receiver is given by

$$N_i + J = kT_e B_r + kT_J B_r \tag{10.20}$$

In this case, the radar detection range is now limited by the receiver signal-to-noise plus interference ratio rather than SNR. More precisely,

$$\left(\frac{S}{J+N}\right) = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 k (T_e + T_J) B_r F L R^4} \tag{10.21}$$

The amount of reduction in the signal-to-noise plus interference ratio because of the jammer effect can be computed from the difference between Eqs. (10.17) and (10.21). It is expressed (in dB) by

$$\Upsilon = 10.0 \times \log\left(1 + \frac{T_J}{T_e}\right) \tag{10.22}$$

Consequently, the RRF is

$$RRF = 10^{\frac{-\Upsilon}{40}} \tag{10.23}$$

MATLAB Function "range_red_factor.m"

The function *"range_red_factor.m"* implements Eqs. (10.22) and (10.23); it is given in Listing 10.5 in Section 10.5. This function generates plots of RRF versus: (1) the radar operating frequency; (2) radar to jammer range; and (3) jammer power. Its syntax is as follows:

[RRF] = range_red_factor (te, pj, gj, g, freq, bj, rangej, lossj)

where

Symbol	Description	Units	Status
te	*radar effective temperature*	*K*	*input*
pj	*jammer peak power*	*W*	*input*
gj	*jammer antenna gain*	*dB*	*input*
g	*radar antenna gain on jammer*	*dB*	*input*
freq	*radar operating frequency*	*Hz*	*input*
bj	*jammer bandwidth*	*Hz*	*input*
rangej	*radar to jammer range*	*Km*	*input*
lossj	*jammer losses*	*dB*	*input*

The following values were used to produce Figs. 10.5 through 10.7.

Symbol	Value
te	*500 kelvin*
pj	*500 KW*
gj	*3 dB*
g	*45 dB*
freq	*10 GHz*
bj	*10 MHZ*
rangej	*750 Km*
lossj	*1 dB*

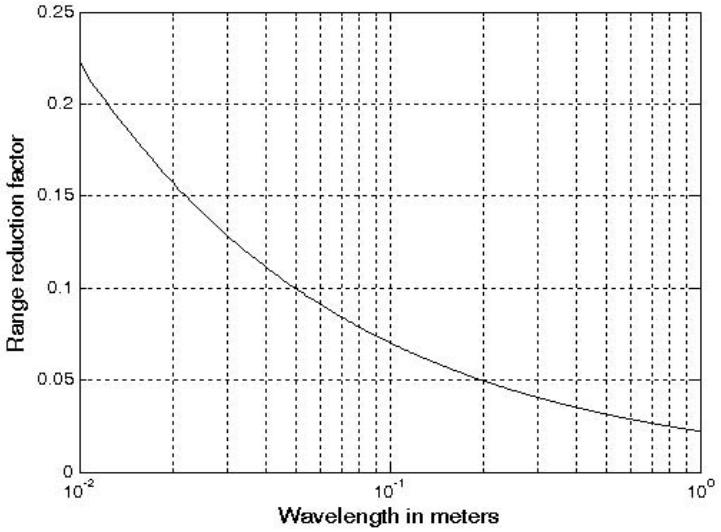

Figure 10.5. Range reduction factor versus radar operating wavelength. This plot was generated using the function *"range_red_factor.m".*

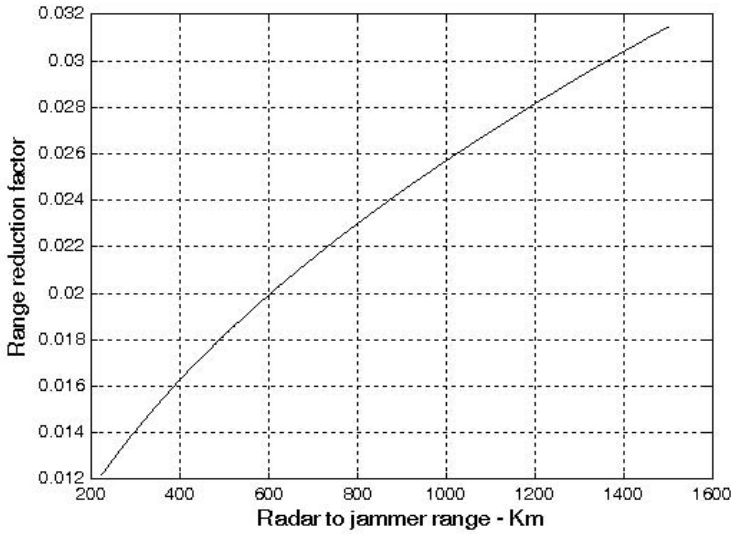

Figure 10.6. Range reduction factor versus radar to jammer range. This plot was generated using the function *"range_red_factor.m".*

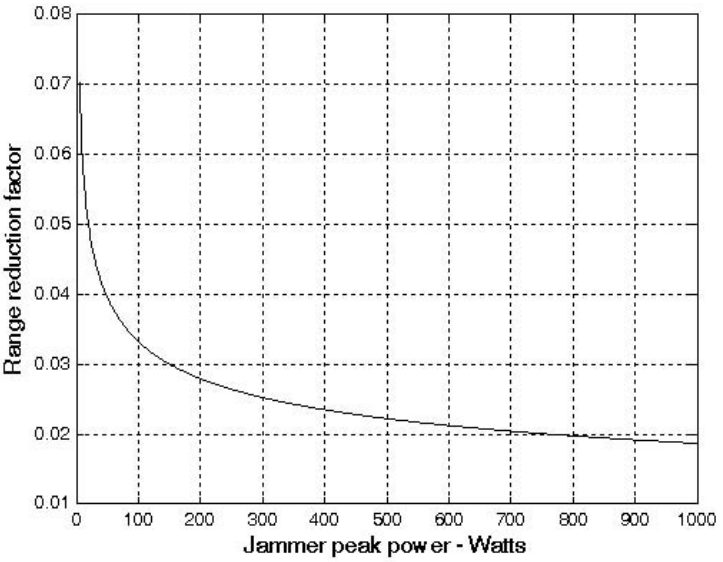

Figure 10.7. Range reduction factor versus jammer peak power. This plot was generated using the function *"range_red_factor.m"*.

10.4. Chaff

In principle, chaff is composed of a large number of small RF reflectors that have large RCS values. Chaff is usually deployed around the target as means of ECM. Historically, chaff was made of aluminum foil; however, in recent years most chaff is made of the more rigid fiber glass with conductive coating.

Chaff can be categorized into two types: (1) denial chaff and (2) deceptive chaff. In the first case, the chaff is deployed in order to screen targets that reside within or near the deployed chaff cloud. In the second case, the chaff cloud is dispersed to complicate and/or overwhelm the tracking and processing functions of the radar by luring the tracker away from the target and/or creating multiple false targets.

The maximum chaff RCS occurs when the individual chaff-dipole length L is one half the radar wavelength. The average RCS for a single dipole when viewed broadside is

$$\sigma_{chaff1} \approx 0.88\lambda^2 \qquad \textbf{(10.24)}$$

and for an average aspect angle, it drops to

$$\sigma_{chaff1} \approx 0.18\lambda^2 \tag{10.25}$$

where the subscript $chaff1$ is used to indicate a single dipole, and λ is the radar wavelength. The total chaff RCS within a radar resolution volume is

$$\sigma_c \approx \frac{0.18\lambda^2 N_D V_{CS}}{L_{beam} V_R} \tag{10.26}$$

where N_D is the total number of dipoles, V_R is the radar resolution cell volume, V_{CS} is the chaff scattering volume, and L_{beam} is the radar antenna beam shape loss for the chaff cloud.

Echoes from a chaff cloud are typically random and have thermal noise-like characteristics because the individual clutter components (scatterers) have random phases and amplitudes. Due to these characteristics, chaff is often statistically described by a probability distribution function. The type of distribution depends on the nature of the chaff cloud itself, radar operating parameters, and the viewing angle of the radar. Thus, the signal-to-chaff ratio is given by

$$\frac{S}{C_{chaff}} = \frac{\sigma}{\sigma_c} CCR \tag{10.27}$$

where σ is the target RCS and CCR is the chaff-cancellation-ratio. The value of CCR depends on the type of chaff mitigation techniques adopted by the radar signal and data processors. Since chaff is a form of volumetric clutter, signal processing and MTI techniques developed for rain and other forms of volumetric clutter can be applied to mitigate many of the effects of chaff. The next section provides an example of one such chaff mitigation technique.

10.4.1. Multiple MTI Chaff Mitigation Technique[1]

In this section, an algorithmic (schema) approach for detecting and tracking targets in highly cluttered environments is presented. The approach is to accurately track the centroid of the chaff cloud using a combination of medium band (MB) and wide-band (WB) range resolution radar waveforms.

At moderate Pulse Repetition Frequencies (PRFs), differential target velocities (about the centroid of the chaff cloud) are detected and tracked via Doppler banks of transversal filters that are tuned to detect the target velocity differ-

1. This section is extracted from the paper: J. Michael Madewell, *Mitigating the Effects of Chaff in Ballistic Missile Defense,* 2003 IEEE Radar Conference, Huntsville, AL, May 2003.

ences. Through sensitivity analysis models, the theoretical lower bound on detectable differential target velocity as a function of the chaff cloud composition (e.g., clutter cross section, clutter spectral width, number of dipoles, and clutter velocity standard deviation) and radar related parameters (e.g., waveform frequency, bandwidth, integration times, PRFs, and signal-to-clutter ratio) are analyzed.

Overview

A five-step approach for detecting and tracking targets in highly cluttered environments has been developed. The five steps are:

1. Utilize a 1 to 5 percent MB bandwidth, high PRF radar waveform, to measure the chaff cloud range extent, centroid, and velocity growth rate.
2. Establish track on the centroid of the chaff cloud with the MB waveform.
3. Based on course track information obtained in steps 1) and 2), implement WB track (10% or greater bandwidth waveform) on the cloud centroid.
4. Design a doppler bank of Moving Target Indicator (MTI) transversal filters to provide adequate gain at specific velocity increments about the WB centroid track.
5. Process the Multiple MTI (M^2) doppler filters in parallel to detect differences in target Doppler (with respect to the cloud centroid track velocity). Targets are detected when integration at the correct Doppler difference occurs.

Operational concerns that have been identified for implementation of this approach include: (1) the ability of a radar to adequately track the centroid of the chaff cloud (i.e., track precision); (2) the ability of a radar to detect small differences in target Doppler relative to the chaff cloud centroid (i.e., Doppler precision); and (3) the ability of a filter (in this case, a bank of MTI's) to achieve the necessary processing gain to detect the target

Theoretical tracking accuracy of a chaff cloud

The single pulse thermal-noise error σ_f in a velocity tracking measurement for optimum processing can be described by

$$\sigma_f = \frac{1}{1.81\tau\sqrt{2 \times SNR}} \tag{10.28}$$

where τ is the pulsewidth and SNR is that for the target in track. To detect targets in clutter, substitute the difference-channel chaff-to-signal ratio for SNR. More precisely,

$$\sigma_f = \frac{1}{1.81\tau\sqrt{2 \times C_{chaff}/S}} \tag{10.29}$$

Fig. 10.8 shows a graph for σ_f versus C_{chaff}/S and τ. This figure can be reproduced using MATLAB program *"fig10_8.m"* given in Listing 10.6 in Section 10.5. This graph will be utilized in the analysis and of the expected M^2 signal processing performance.

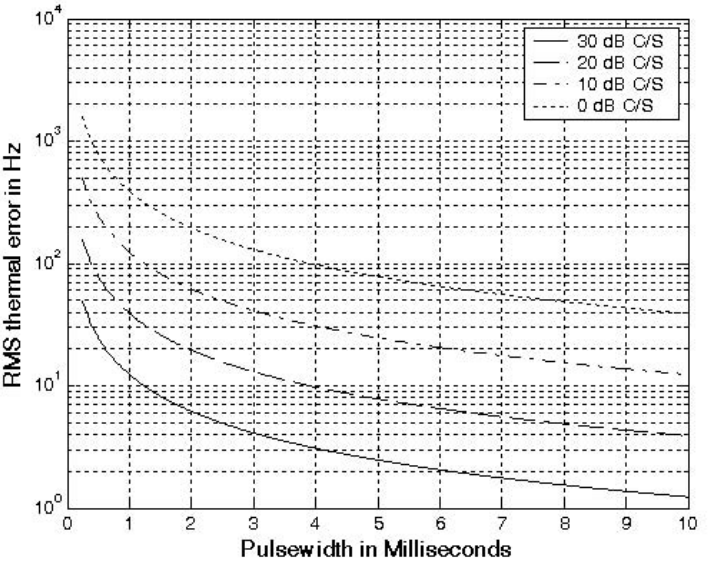

Figure 10.8. Single pulse thermal noise error versus C_{chaff}/S and τ.

Multiple MTI (M^2) Doppler Filter Design

The M^2 Doppler filter design is derived from the theoretical N-tap delay line MTI canceller. The general formula for the improvement factor was derived in Chapter 7 (Section 7.7.2). A bank of N MTI Doppler filters that cover the frequency range from 0 to the PRF will achieve performance beyond that of a conventional MTI. The weights are given by:

$$W_{i,k} = e^{\frac{j2\pi(i-1)}{k/N}} \tag{10.30}$$

where the index k is between 0 to N-1 and corresponds to the N MTI Doppler filter bank. In this design, a 5-tap delay line MTI filter is considered. The transfer function for the overall Doppler bank is

$$H^k(f) = \sum_{i=1}^{N} e^{(-j2\pi)(i-1)(fT-k/N)} \tag{10.31}$$

where

$$T = 1/PRF \tag{10.32}$$

It follows that the magnitude of the frequency response is

$$\left|H^k(f)\right| = \left|\frac{\sin(\pi N(fT - k/N))}{\sin(\pi(fT - k/N))}\right| \tag{10.33}$$

The impulse response for a kth 5-tap MTI filter is

$$y^k(t) = v_k(t) - 5v_k(t-T) + 10v_k(t-2T) - 10v_k(t-3T) + 5v_k(t-4T) - v_k(t-5T) \tag{10.34}$$

v_k is the input signal. The corresponding transfer function is

$$Y^k(f) = 25(\sin(\pi fT))^5 \tag{10.35}$$

Fig. 10.9 shows a block diagram for the M^2 filter. Since each filter occupies approximately $(1/N)th$ the clutter and signal bandwidth, the combined performance of the M^2 Doppler filter performance is greater than that of a single delay-line canceller that does not utilize Doppler information. The clutter mitigation performance of the M^2 Doppler filter, however, will likely be determined by the coherence times of the target and/or the clutter.

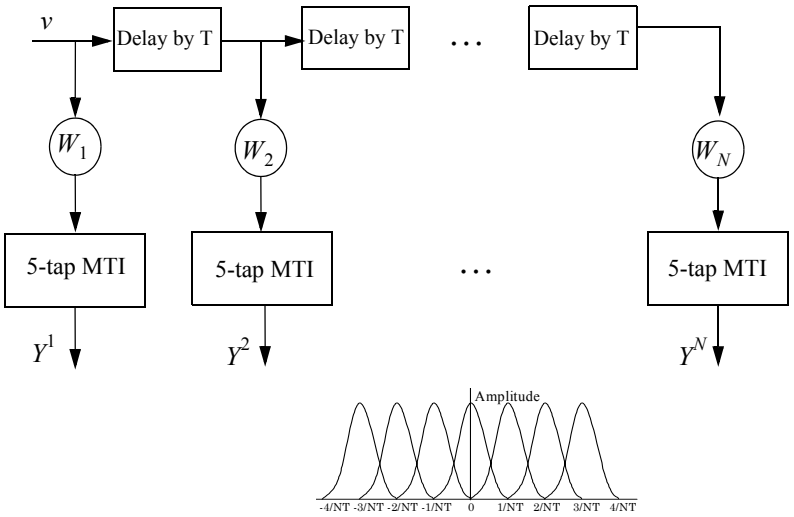

Figure 10.9. Block diagram for the M^2 algorithm, and corresponding frequency response of the MTI filters (N=8).

Processor Implementation And Simulated Results

The M^2 filter approach outlined in this section requires a very accurate track of the centroid of the chaff cloud being probed. As described earlier, initiation of track on the chaff cloud centroid is achieved with a MB range resolution waveform (step 1). As an example, assume that an X-band radar (10 GHz) is engaging one or more ballistic targets enveloped in a chaff cloud that contains 1 million dipoles occupying a 1-kilometer range extent. Assuming that the chaff cloud velocity distribution can be accurately modeled by Gaussian statistics, approximately 67% of these dipoles will reside in 333 meters of range extent. With these assumptions, the combined average RCS of the dipoles (RCS_D) contained within a radar range resolution cell of this length (333 m) can be approximated by

$$RCS_D = 0.18 N_D \lambda^2 = 0.18 \times 670,000 \times 0.03^2 = 108.54 \Rightarrow 20.4 dBsm \quad \textbf{(10.36)}$$

The RCS of a typical ballistic Reentry Vehicle (RV) at forward aspect viewing angles can be $-20 dBsm$ or smaller. Therefore, the MB C_{chaff}/S for a typical RV enveloped by the chaff cloud assumed above can approach $40 dB$ or greater. Using an 8-msec pulsewidth and $30 dB$ C_{chaff}/S, the theoretical, single pulse, minimum rms track error is approximately $f_e = 1 Hz$. At X-band frequencies, this translates to a single pulse velocity error of

$$v_e = \frac{f_e \lambda}{2} = 0.015 m/s \quad \textbf{(10.37)}$$

Note that for a train of pulses, this velocity error can be reduced by a factor of 10 or more. Thus, for a typical X-band radar, theory suggests that the track precision of the chaff cloud centroid can approach 0.0015 m/s or better. This track precision is much less than the WB range resolution capability of the radar and therefore can be utilized to bootstrap the WB tracker (steps 2 and 3).

Assume a Gaussian chaff clutter velocity distribution and denote it v_g. If $v_g = 1.8 m/s$ ($\pm 0.9 m/s$ relative to the cloud centroid velocity), the minimum PRF required to meet the Nyquist sampling criterion is

$$PRF = f_r \geq 2 \times \frac{2 v_g}{\lambda} = 240 Hz \quad \textbf{(10.38)}$$

Also, assume that a bank of Doppler MTI's (step 4) can be formed to cover this frequency range. Note that 256 is the closest 2^N multiple for implementation with the Fast Fourier Transform (FFT). Using a 256 point FFT design, each filter will contain approximately 1/256 of the total clutter velocities (about 0.03 m/s of velocity clutter per MTI Doppler filter). In addition, by utilizing the WB track waveform, a very precise range-Doppler image can be formed (with each range-Doppler resolution cell containing approximately 15 cm by 0.03 m/s of

clutter). This design effectively reduces the amount of clutter that competes with an individual target scatter by a factor of more than 40 dB, thus reducing the C_{chaff}/S by this same amount.

For extreme chaff cases where the initial WB range-Doppler image S/C is negative, an N-pulse coherent sliding window routine can be applied to the data prior to implementing the M^2 algorithm. For example, a 16 pulse coherent sliding window can provide up to 12 dB of S/C_{chaff} improvement. One should ensure that the number of pulses integrated is less than the coherency time of the target and clutter. Other constraints in implementing this approach are to ensure that the target phase does not deviate very much during the integration period (to ensure optimum coherent processing gain) and the target position does not migrate to another range and/or Doppler cell (often referred to as range-Doppler walk). The zero Doppler filter (and/or near zero Doppler filters) can be used to perform statistics on the clutter and to adaptively adjust the optimal threshold setting to obtain low false alarms and high probabilities of detection over time.

A model for the M^2 signal processor has been developed using MATLAB. Fig. 10.10 shows a plot of the amplitude versus range and Doppler (256x256 range-Doppler image) of three constant -20 dBsm target scatterers that are embedded in approximately -15 dBsm Gaussian white noise. In this figure, the noise completely envelops the signal. These modeling results are comparable to the output of a typical range Doppler imaging radar. Fig. 10.11 shows the results obtained by executing the first two blocks of the M^2 signal processor. As expected, the three scatterers rise from above the noise and now have an S/C_{chaff} ratio of approximately 7 dB.

Finally, Fig. 10.12 shows the results obtained by implementing the entire top portion of the M^2 signal processing chain. No attempt was made to optimize the threshold level. Instead, the threshold was manually set to -43 dB to allow for some of the higher false alarms to be seen in the figure. The largest amplitude false alarms are approximately -34 dB. Meanwhile, the amplitudes of the target returns have been reduced (less than 1 dB) from that of Fig. 10.11. Therefore, the S/C_{chaff} improvement in Fig. 10.12 over that shown in Fig. 10.11 is approximately 8 to 9 dB. Hence, the processing gain attributed to the M^2 signal processor is more than 20 dB above that of traditional range Doppler processing.

In summary, one concludes that the M^2 signal processing algorithm for detecting and tracking ballistic missile targets in highly cluttered environments can provide better than 20 dB S/C_{chaff} improvement over that of traditional range Doppler processing techniques alone.

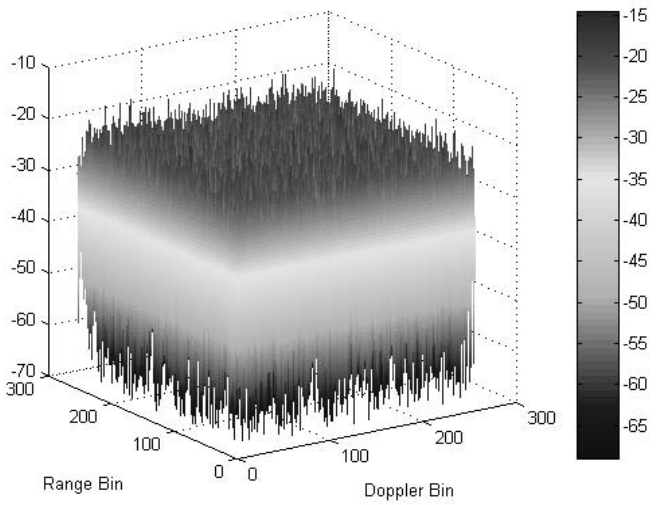

Figure 10.10. Range -Doppler image for three targets embedded in chaff.

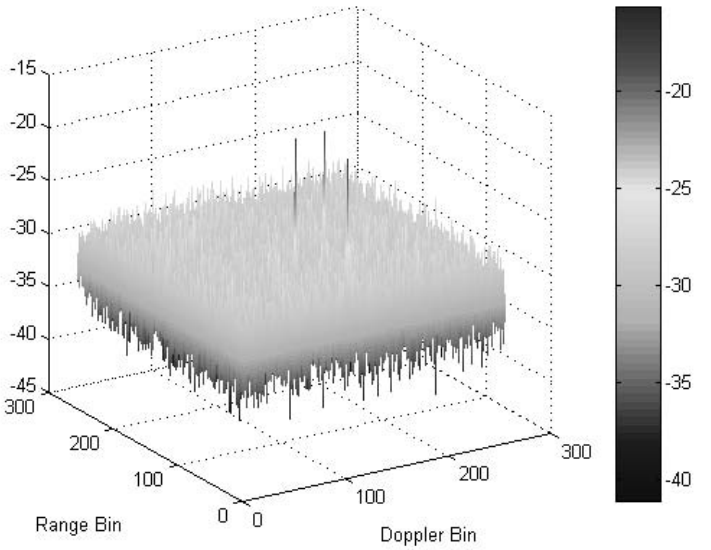

Figure 10.11. Image from Fig. 10.10 after a 16-point sliding window coherent integration process.

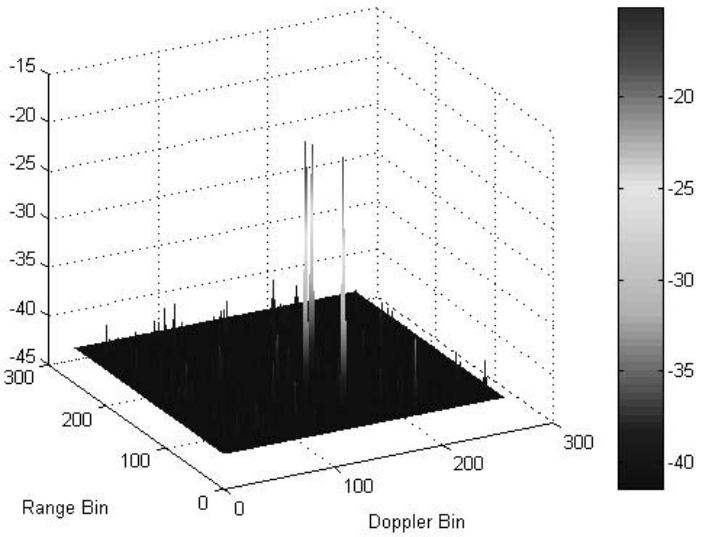

Figure 10.12. Image from Fig. 10.11 after applying the M^2 algorithm.

10.5. MATLAB Program and Function Listings

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

Listing 10.1. MATLAB Function "ssj_req.m"

```
function [BR_range] = ssj_req (pt, g, freq, sigma, b, loss, ...
  pj, bj, gj, lossj)
% This function implements Eq. (10.9)
c = 3.0e+8;
lambda = c / freq;
lambda_db = 10*log10(lambda^2);
if (loss == 0.0)
  loss = 0.000001;
end
if (lossj == 0.0)
  lossj =0.000001;
end
sigmadb =10*log10(sigma);
```

```
pt_db = 10*log10(pt);
b_db = 10*log10(b);
bj_db = 10*log10(bj);
pj_db = 10*log10(pj);
factor = 10*log10(4.0 *pi);
BR_range = sqrt((pt * (10^(g/10)) * sigma * bj * (10^(lossj/10))) / ...
   (4.0 * pi * pj * (10^(gj/10)) * b * ...
   (10^(loss/10)))) / 1000.0
s_at_br = pt_db + 2.0 * g + lambda_db + sigmadb - ...
      3.0 * factor - 4.* 10*log10(BR_range) - loss
index =0;
for ran_var = .1:10:10000
   index = index + 1;
   ran_db = 10*log10(ran_var * 1000.0);
   ssj(index) = pj_db + gj + lambda_db + g + b_db - 2.0 * factor - ...
      2.0 * ran_db - bj_db - lossj + s_at_br ;
   s(index) = pt_db + 2.0 * g + lambda_db + sigmadb - ...
      3.0 * factor - 4.* ran_db - loss + s_at_br ;
end
ranvar = .1:10:10000;
ranvar = ranvar ./ BR_range;
semilogx (ranvar,s,'k',ranvar,ssj,'k-.');
axis([.1 1000 -90 40])
xlabel ('Range normalized to cross-over range');
legend('Target echo','SSJ')
ylabel ('Relative signal or jamming amplitude - dB');
grid
pj_var = 1:1:1000;
BR_pj = sqrt((pt * (10^(g/10)) * sigma * bj * (10^(lossj/10))) ...
   ./ (4.0 * pi .* pj_var * (10^(gj/10)) * b * (10^(loss/10)))) ./ 1000;
pt_var = 1000:100:10e6;
BR_pt = sqrt((pt_var * (10^(g/10)) * sigma * bj * (10^(lossj/10))) ...
   ./ (4.0 * pi .* pj * (10^(gj/10)) * b * (10^(loss/10)))) ./ 1000;
figure (2)
subplot (2,1,1)
semilogx (BR_pj,'k')
xlabel ('Jammer peak power - Watts');
ylabel ('Burn-through range - Km')
grid
subplot (2,1,2)
semilogx (BR_pt,'k')
xlabel ('Radar peak power - KW')
ylabel ('Burn-through range - Km')
grid
```

Listing 10.2. MATLAB Function "sir.m"

```
function [SIR] = sir (pt, g, freq, sigma, tau,T0,  loss, R, pj, bj, gj, lossj);
c = 3.0e+8;
k = 1.38e-23;
%R = linspace(rmin, rmax, 1000);
range = R .* 1000;
lambda = c / freq;
gj = 10^(gj/10);
G = 10^(g/10);
ERP1 = pj * gj / lossj;
ERP_db = 10*log10(ERP1);
% Calculate Eq. (10.10)
Ar = lambda *lambda * G / 4 /pi;
num1 = pt * tau * G * sigma * Ar;
demo1 = 4^2 * pi^2 * loss .* range.^4;
demo2 = 4 * pi * bj .* range.^2;
num2 = ERP1 * Ar;
val11 = num1 ./ demo1;
val21 = num2 ./demo2;
sir = val11 ./ (val21 + k * T0);
SIR = 10*log10(sir);
figure (1)
plot (R, SIR,'k')
xlabel ('Detection range in Km');
ylabel ('S/(J+N) in dB')
grid
```

Listing 10.3. MATLAB Function "burn_thru.m"

```
function [Range] = burn_thru (pt, g, freq, sigma, tau, T0, loss, pj, bj, gj,
lossj,sir0,ERP);
c = 3.0e+8;
k = 1.38e-23;
%R = linspace(rmin, rmax, 1000);
sir0 = 10^(sir0/10);
lambda = c / freq;
gj = 10^(gj/10);
G = 10^(g/10);
 Ar = lambda *lambda * G / 4 /pi;
%ERP = linspace(1,1000,5001);
num32 = ERP .* Ar;
demo3 = 8 *pi * bj * k * T0;
demo4 = 4^2 * pi^2 * k * T0 * sir0;
```

```
val1 = (num32 ./ demo3).^2;
val2 = (pt * tau * G * sigma * Ar)/(4^2 * pi^2 * loss * sir0 * k *T0);
val3 = sqrt(val1 + val2);
val4 = (ERP .* Ar) ./ demo3;
Range = sqrt(val3 - val4) ./ 1000;
figure (1)
plot (10*log10(ERP), Range,'k')
xlabel (' Jammer ERP in dB')
ylabel ('Burnthrough range in Km')
grid
```

Listing 10.4. MATLAB Function "soj_req.m"

```
function [BR_range] = soj_req (pt, g, sigma, b, freq, loss, range, ...
  pj, bj,gj, lossj, gprime, rangej)
% This function implements equations for SOJs
c = 3.0e+8;
lambda = c / freq;
lambda_db = 10*log10(lambda^2)
if (loss == 0.0)
  loss = 0.000001;
end
if (lossj == 0.0)
  lossj =0.000001;
end
sigmadb = 10*log10(sigma);
range_db = 10*log10(range * 1000.);
rangej_db = 10*log10(rangej * 1000.)
pt_db = 10*log10(pt);
b_db = 10*log10(b);
bj_db = 10*log10(bj);
pj_db = 10*log10(pj);
factor = 10*log10(4.0 *pi);
BR_range = ((pt * 10^(2.0*g/10) * sigma * bj * 10^(lossj/10) * ...
  (rangej)^2) / (4.0 * pi * pj * 10^(gj/10) * 10^(gprime/10) * ...
  b * 10^(loss/10)))^.25 / 1000.
s_at_br = pt_db + 2.0 * g + lambda_db + sigmadb - ...
  3.0 * factor - 4.0 * 10*log10(BR_range) - loss
index =0;
for ran_var = .1:1:1000;
  index = index + 1;
  ran_db = 10*log10(ran_var * 1000.0);
  s(index) = pt_db + 2.0 * g + lambda_db + sigmadb - ...
    3.0 * factor - 4.0 * ran_db - loss + s_at_br;
```

```
  soj(index) = s_at_br - s_at_br;
end
ranvar = .1:1:1000;
%ranvar = ranvar ./BR_range;
semilogx (ranvar,s,'k',ranvar,soj,'k-.');
xlabel ('Range normalized to cross-over range');
legend('Target echo','SOJ')
ylabel ('Relative signal or jamming amplitude - dB');
```

Listing 10.5. MATLAB Function "range_red_factor.m"

```
function RRF = range_red_factor (te, pj, gj, g, freq, bj, rangej, lossj)
% This function computes the range reduction factor and produces
% plots of RRF versus wavelength, radar to jammer range, and jammer power
c = 3.0e+8;
k = 1.38e-23;
lambda = c / freq;
gj_10 = 10^( gj/10);
g_10 = 10^( g/10);
lossj_10 = 10^(lossj/10);
index = 0;
for wavelength = .01:.001:1
  index = index +1;
  jamer_temp = (pj * gj_10 * g_10 *wavelength^2) / ...
    (4.0^2 * pi^2 * k * bj * lossj_10 * (rangej * 1000.0)^2);
  delta = 10.0 * log10(1.0 + (jamer_temp / te));
  rrf(index) = 10^(-delta /40.0);
end
w = 0.01:.001:1;
figure (1)
semilogx(w,rrf,'k')
grid
xlabel ('Wavelength in meters')
ylabel ('Range reduction factor')
index = 0;
for ran =rangej*.3:1:rangej*2
  index = index + 1;
  jamer_temp = (pj * gj_10 * g_10 *wavelength^2) / ...
    (4.0^2 * pi^2 * k * bj * lossj_10 * (ran * 1000.0)^2);
  delta = 10.0 * log10(1.0 + (jamer_temp / te));
  rrf1(index) = 10^(-delta /40.0);
end
figure(2)
ranvar = rangej*.3:1:rangej*2 ;
```

```
plot(ranvar,rrf1,'k')
grid
xlabel ('Radar to jammer range - Km')
ylabel ('Range reduction factor')
index = 0;
for pjvar = pj*.01:1:pj*2
  index = index + 1;
  jamer_temp = (pjvar * gj_10 * g_10 *wavelength^2) / ...
    (4.0^2 * pi^2 * k * bj * lossj_10 * (rangej * 1000.0)^2);
  delta = 10.0 * log10(1.0 + (jamer_temp / te));
  rrf2(index) = 10^(-delta /40.0);
end
figure(3)
pjvar = pj*.01:1:pj*2;
plot(pjvar,rrf2,'k')
grid
xlabel ('Jammer peak power - Watts')
ylabel ('Range reduction factor')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Use this input file to reproduce Figs. 10.5 through 10.7
clear all
te = 500.0;    % radar effective temp. in Kelvin
pj= 500; % jammer peak power in W
gj = 3.0;      % jammer antenna gain in dB
g = 45.0;      % radar antenna gain
freq = 10.0e+9;% radar operating frequency in Hz
bj= 10.0e+6;   % radar operating bandwidth in Hz
rangej = 750.0;% radar to jammer range in Km
lossj = 1.0;   % jammer losses in dB
```

Listing 10.6. MATLAB Program "fig10_8.m"

```
% Use this program to reproduce Fig. 10.8 in the text
clear all
close all
tau = linspace(.25,10,500);
taum = tau .* 1e-3;
C_S = [-20 -10 0 10];
c_s = 10.^(C_S./10);
for n = 1:size(C_S,2)
   val1 = 1 / (1.81*sqrt(2*c_s(n)));
   sigma(n,:) = val1 ./ taum;
end
figure (1)
```

```
semilogy(tau,sigma(1,:),'k',tau,sigma(2,:),'k-- ',tau,sigma(3,:),'k-.', ...
   tau,sigma(4,:),'k:');
xlabel('Pulsewidth in Milliseconds')
ylabel('RMS thermal error in Hz')
legend('-20 dB C/S','-10 dB C/S','0 dB C/S','10 dB C/S')
grid
```

Chapter 11

Radar Cross Section (RCS)

In this chapter, the phenomenon of target scattering and methods of RCS calculation are examined. Target RCS fluctuations due to aspect angle, frequency, and polarization are presented. Radar cross section characteristics of some simple and complex targets are also introduced.

11.1. 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 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. 11.1. The amount of reflected power from the target is

$$P_r = \sigma P_{Di} \tag{11.1}$$

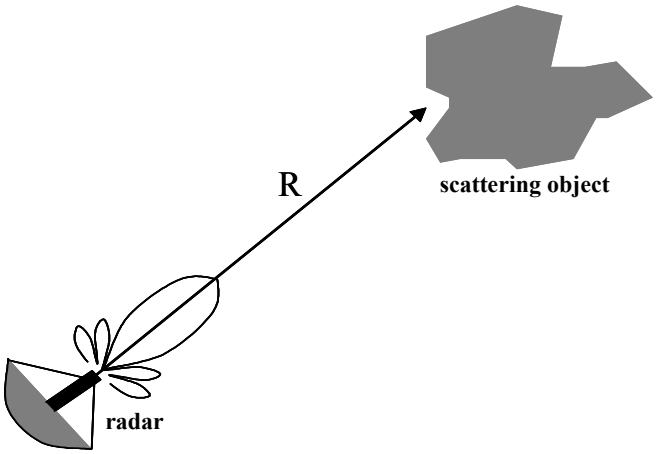

Figure 11.1. Scattering object located at range R.

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

$$P_{Dr} = P_r / (4\pi R^2) \tag{11.2}$$

Equating Eqs. (11.1) and (11.2) yields

$$\sigma = 4\pi R^2 \left(\frac{P_{Dr}}{P_{Di}} \right) \tag{11.3}$$

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. (11.3) is modified

$$\sigma = 4\pi R^2 \lim_{R \to \infty} \left(\frac{P_{Dr}}{P_{Di}} \right) \tag{11.4}$$

The RCS defined by Eq. (11.4) is often referred to as either the monostatic RCS, the backscattered RCS, or simply 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

$$\sigma_t = \frac{1}{4\pi} \int_{\varphi_s = 0}^{2\pi} \int_{\theta_s = 0}^{\pi} \sigma(\theta_s, \varphi_s) \sin\theta_s \, d\theta \, d\varphi_s \qquad (11.5)$$

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. RCS measurements in the frequency region, where the target extent and the wavelength are comparable, are 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. Wide Band (WB) RCS measurements will be briefly addressed in a later section. Wide band 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.

11.2. 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 scattered waves from an object 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 few dBs of the truth. In general, such a variation is quite acceptable by radar engineers and designers. Approximate methods are usually the main source for predicting 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 (see bibliography) for more details on these and other approximate methods.

11.3. 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. 11.2. 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

$$elec\text{–}spacing = \frac{2 \times (1.0 \times \cos(10°))}{\lambda} \tag{11.6}$$

λ is the radar operating wavelength.

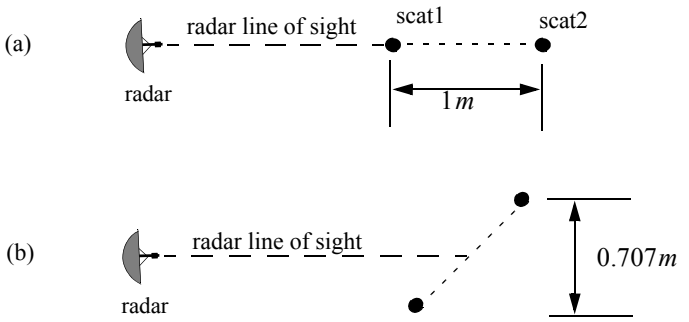

Figure 11.2. RCS dependency on aspect angle. (a) Zero aspect angle, zero electrical spacing. (b) $45°$ **aspect angle,** 1.414λ **electrical spacing.**

Fig. 11.3 shows the composite RCS corresponding to this experiment. This plot can be reproduced using MATLAB function *"rcs_aspect.m"* given in Listing 11.1 in Section 11.9. As clearly indicated by Fig. 11.3, 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 to be 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.

MATLAB Function "rcs_aspect.m"

The function *"rcs_aspect.m"* computes and plots the RCS dependency on 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*

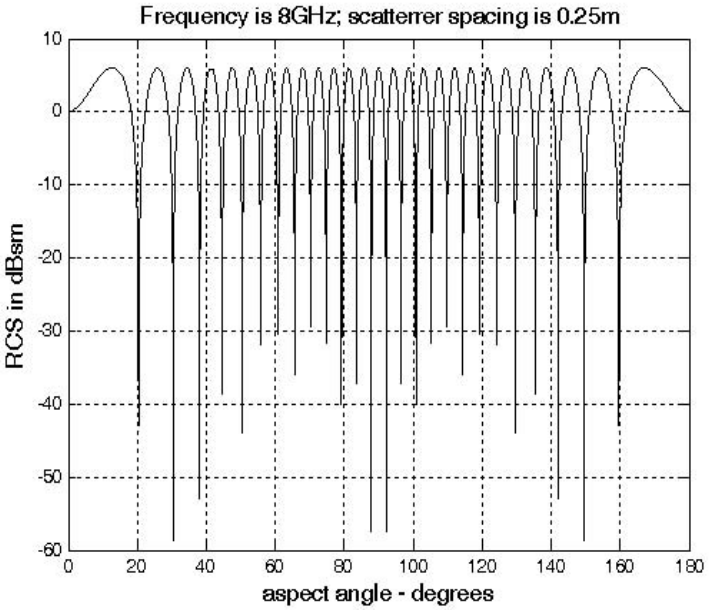

Figure 11.3. Illustration of RCS dependency on aspect angle.

Next, to demonstrate RCS dependency on frequency, consider the experiment shown in Fig. 11.4. 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). Figs. 11.5 and 11.6 show the composite RCS versus frequency for scatterer spacing of 0.25 and 0.75 meters.

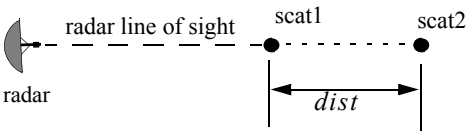

Figure 11.4. Experiment setup which demonstrates RCS dependency on frequency; dist = 0.1, or 0.7 m.

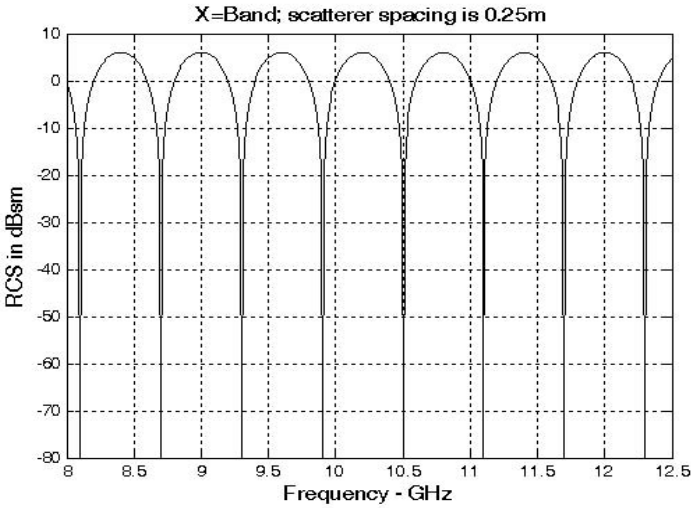

Figure 11.5. Illustration of RCS dependency on frequency.

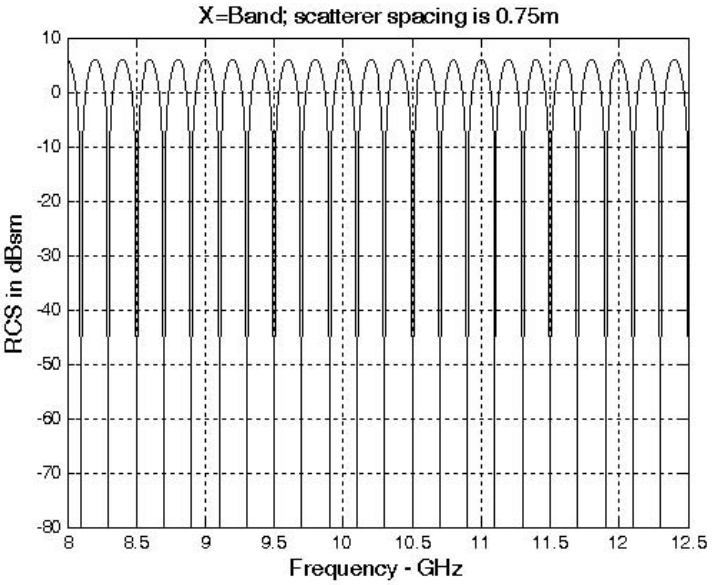

Figure 11.6. Illustration of RCS dependency on frequency.

The plots shown in Figs. 11.5 and 11.6 can be reproduced using MATLAB function *"rcs_frequency.m"* given in Listing 11.2 in Section 11.9. From those two figures, RCS fluctuation as a function of frequency is evident. Little 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.

MATLAB Function "rcs_frequency.m"

The function *"rcs_frequency.m"* computes and plots 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	*start of frequency band*	*Hz*	*input*
frequ	*end of frequency band*	*Hz*	*input*
rcs	*array of RCS versus aspect angle*	*dBsm*	*output*

Referring to Fig. 11.2, assume that the two scatterers complete a full revolution about the radar line of sight in $T_{rev} = 3\,\text{sec}$. Furthermore, assume that an X-band radar ($f_0 = 9GHz$) is used to detect (observe) those two scatterers using a PRF $f_r = 300Hz$ for a period of 3 seconds. Finally, assume a NB bandwidth $B_{NB} = 1MHz$ and a WB bandwidth $B_{WB} = 2GHz$. It follows that the radar's NB and WB range resolutions are respectively equal to $\Delta R_{NB} = 150m$ and $\Delta R_{WB} = 7.5cm$.

Fig. 11.7 shows a plot of the detected range history for the two scatterers using NB detection. Clearly, the two scatterers are completely contained within one range bin. Fig. 11.8 shows the same; however, in this case WB detection is utilized. The two scatterers are now completely resolved as two distinct scatterers, except during the times where both point scatterers fall within the same range bin.

11.4. 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.

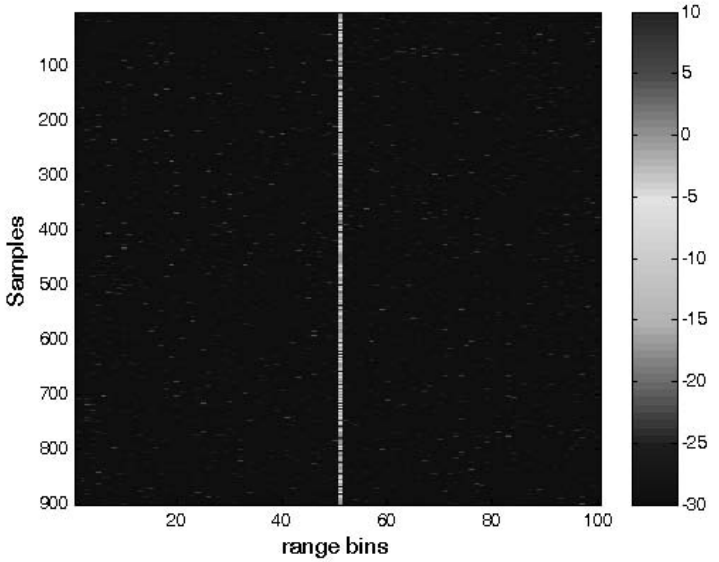

Figure 11.7. NB detection of the two scatterers shown in Fig. 11.2.

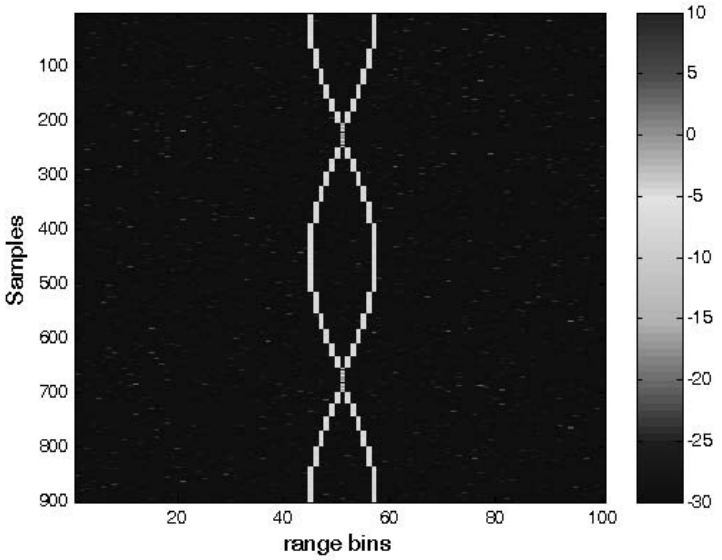

Figure 11.8. WB detection of the two scatterers shown in Fig. 11.2.

11.4.1. Polarization

The x and y electric field components for a wave traveling along the positive z direction are given by

$$E_x = E_1 \sin(\omega t - kz) \tag{11.7}$$

$$E_y = E_2 \sin(\omega t - kz + \delta) \tag{11.8}$$

where $k = 2\pi/\lambda$, ω is the wave frequency, the angle δ is the time phase angle 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. 11.9.

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. (11.7) and (11.8) can be combined to give the instantaneous total electric field,

$$\vec{E} = \hat{a}_x E_1 \sin(\omega t - kz) + \hat{a}_y E_2 \sin(\omega t - kz + \delta) \tag{11.9}$$

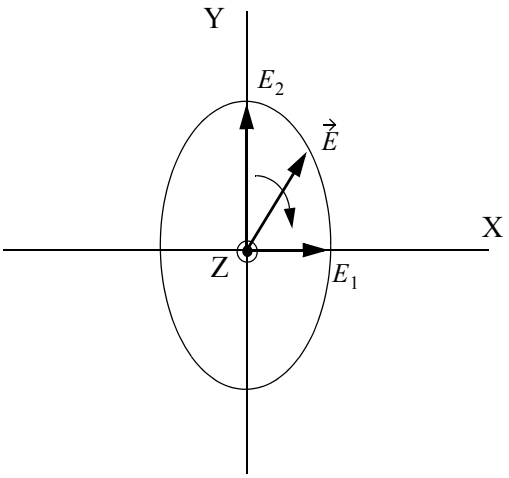

Figure 11.9. Electric field components along the x and y directions. The positive z direction is out of the page.

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. (11.9) can be rewritten as

$$\frac{E_x^2}{E_1^2} - \frac{2E_x E_y \cos\delta}{E_1 E_2} + \frac{E_y^2}{E_2^2} = (\sin\delta)^2 \qquad \textbf{(11.10)}$$

Note that Eq. (11.10) has no dependency on ωt.

In the most general case, the polarization ellipse may have any orientation, as illustrated in Fig. 11.10. The angle ξ is called the tilt angle of the ellipse. In this case, AR is given by

$$AR = \frac{OA}{OB} \qquad (1 \le AR \le \infty) \qquad \textbf{(11.11)}$$

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.

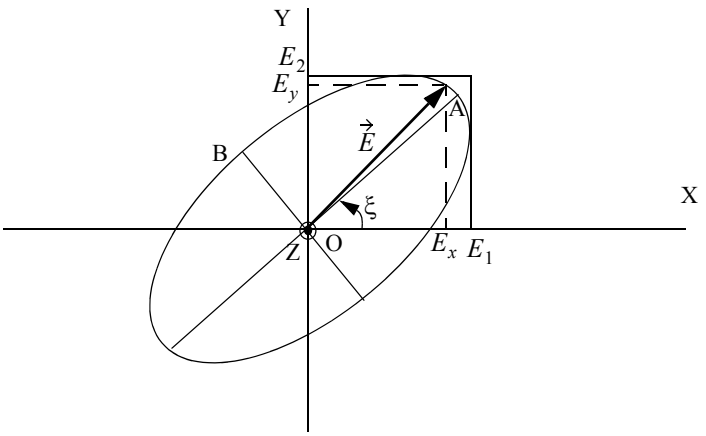

Figure 11.10. Polarization ellipse in the general case.

In general, an arbitrarily polarized electric field may be written as the sum of two circularly polarized fields. More precisely,

$$\vec{E} = \overrightarrow{E_R} + \overrightarrow{E_L} \tag{11.12}$$

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

$$\overrightarrow{E_R} = \overrightarrow{E_V} + j\overrightarrow{E_H} \tag{11.13}$$

$$\overrightarrow{E_L} = \overrightarrow{E_V} - j\overrightarrow{E_H} \tag{11.14}$$

where $\overrightarrow{E_V}$ and $\overrightarrow{E_H}$ are the fields with vertical and horizontal polarizations, respectively. Combining Eqs. (11.13) and (11.14) yields

$$E_R = \frac{E_H - jE_V}{\sqrt{2}} \tag{11.15}$$

$$E_L = \frac{E_H + jE_V}{\sqrt{2}} \tag{11.16}$$

Using matrix notation Eqs. (11.15) and (11.16) can be rewritten as

$$\begin{bmatrix} E_R \\ E_L \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \begin{bmatrix} E_H \\ E_V \end{bmatrix} = [T] \begin{bmatrix} E_H \\ E_V \end{bmatrix} \tag{11.17}$$

$$\begin{bmatrix} E_H \\ E_V \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ j & -j \end{bmatrix} \begin{bmatrix} E_R \\ E_L \end{bmatrix} = [T]^{-1} \begin{bmatrix} E_H \\ E_V \end{bmatrix} \tag{11.18}$$

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 which is RCP becomes LCP when reflected, and a wave which 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.

Example:

Plot the locus of the electric field vector for the following cases:

$$case1:\ \vec{E}(t,z) = \hat{a}_x \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) + \hat{a}_y \sqrt{3} \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right)$$

$$case\ 2:\quad \vec{E}(t,z) = \hat{a}_x \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) + \hat{a}_y \sin\left(\omega_0 t + \frac{2\pi z}{\lambda}\right)$$

$$case\ 3:\quad \vec{E}(t,z) = \hat{a}_x \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) + \hat{a}_y \cos\left(\omega_0 t + \frac{2\pi z}{\lambda} + \frac{\pi}{6}\right)$$

$$case\ 4:\ \vec{E}(t,z) = \hat{a}_x \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) + \hat{a}_y \sqrt{3} \cos\left(\omega_0 t + \frac{2\pi z}{\lambda} + \frac{\pi}{3}\right)$$

Solution:

The MATLAB program "example11_1.m" was developed to calculate and plot the loci of the electric fields. Figs. 11.11 through 11.14 show the desired electric fields' loci. See listing 11.3 in Section 11.9.

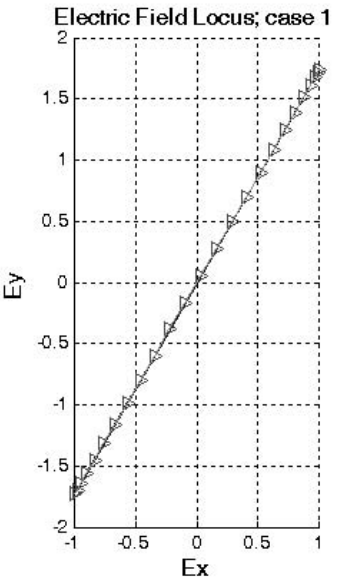

Figure 11.11. Linearly polarized electric field.

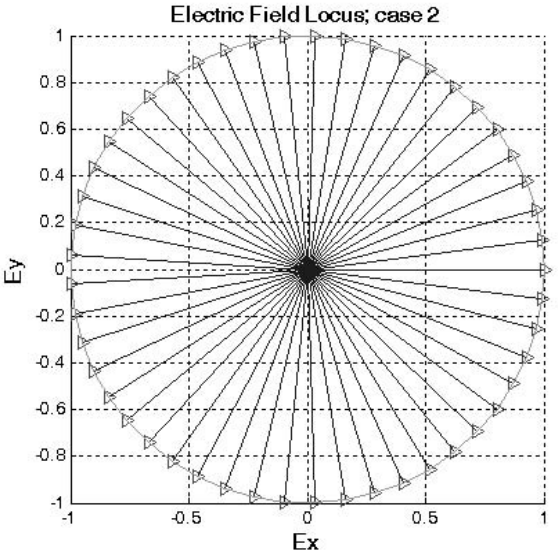

Figure 11.12. Circularly polarized electric field.

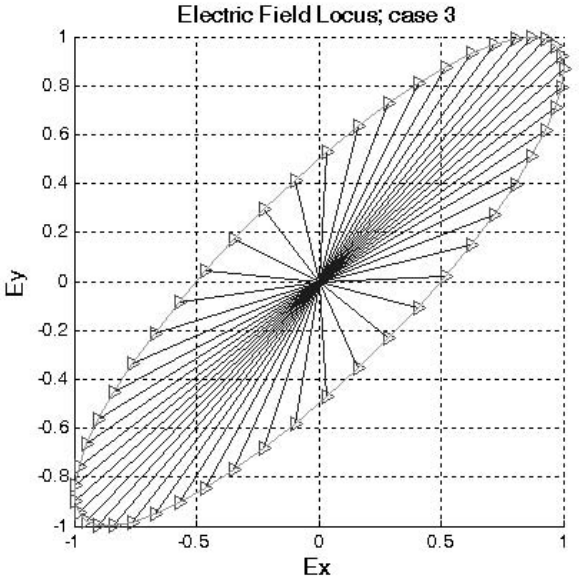

Figure 11.13. Elliptically polarized electric field.

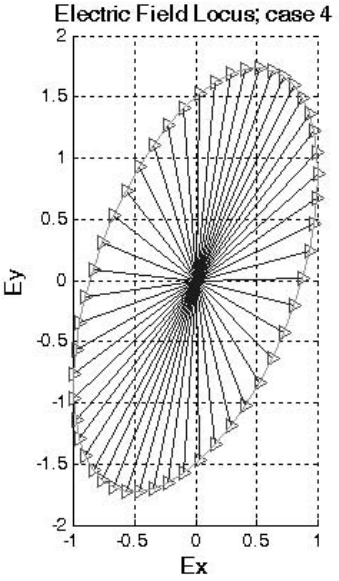

Figure 11.14. Elliptically polarized electric field.

11.4.2. 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

$$\begin{bmatrix} E_1^s \\ E_2^s \end{bmatrix} = [S] \begin{bmatrix} E_1^i \\ E_2^i \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \begin{bmatrix} E_1^i \\ E_2^i \end{bmatrix} \tag{11.19}$$

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. (11.3), the backscattered RCS is related to the scattering matrix components by the following relation:

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = 4\pi R^2 \begin{bmatrix} |s_{11}|^2 & |s_{12}|^2 \\ |s_{21}|^2 & |s_{22}|^2 \end{bmatrix} \tag{11.20}$$

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 for ways to calculate the scattering matrix $[S]$.

Rewriting Eq. (11.20) in terms of the different possible orthogonal polarizations yields

$$\begin{bmatrix} E_H^s \\ E_V^s \end{bmatrix} = \begin{bmatrix} s_{HH} & s_{HV} \\ s_{VH} & s_{VV} \end{bmatrix} \begin{bmatrix} E_H^i \\ E_V^i \end{bmatrix} \tag{11.21}$$

$$\begin{bmatrix} E_R^s \\ E_L^s \end{bmatrix} = \begin{bmatrix} s_{RR} & s_{RL} \\ s_{LR} & s_{LL} \end{bmatrix} \begin{bmatrix} E_R^i \\ E_L^i \end{bmatrix} \tag{11.22}$$

By using the transformation matrix $[T]$ in Eq. (11.17), the circular scattering elements can be computed from the linear scattering elements

$$\begin{bmatrix} s_{RR} & s_{RL} \\ s_{LR} & s_{LL} \end{bmatrix} = [T] \begin{bmatrix} s_{HH} & s_{HV} \\ s_{VH} & s_{VV} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} [T]^{-1} \tag{11.23}$$

and the individual components are

$$\begin{aligned} s_{RR} &= \frac{-s_{VV} + s_{HH} - j(s_{HV} + s_{VH})}{2} \\ s_{RL} &= \frac{s_{VV} + s_{HH} + j(s_{HV} - s_{VH})}{2} \\ s_{LR} &= \frac{s_{VV} + s_{HH} - j(s_{HV} - s_{VH})}{2} \\ s_{LL} &= \frac{-s_{VV} + s_{HH} + j(s_{HV} + s_{VH})}{2} \end{aligned} \tag{11.24}$$

Similarly, the linear scattering elements are given by

$$\begin{bmatrix} s_{HH} & s_{HV} \\ s_{VH} & s_{VV} \end{bmatrix} = [T]^{-1} \begin{bmatrix} s_{RR} & s_{RL} \\ s_{LR} & s_{LL} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} [T] \tag{11.25}$$

and the individual components are

$$
\begin{aligned}
s_{HH} &= \frac{-s_{RR} + s_{RL} + s_{LR} - s_{LL}}{2} \\
s_{VH} &= \frac{j(s_{RR} - s_{LR} + s_{RL} - s_{LL})}{2} \\
s_{HV} &= \frac{-j(s_{RR} + s_{LR} - s_{RL} - s_{LL})}{2} \\
s_{VV} &= \frac{s_{RR} + s_{LL} + js_{RL} + s_{LR}}{2}
\end{aligned}
\tag{11.26}
$$

11.5. RCS of Simple Objects

This section presents examples of 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. 11.15.

In this section, it is assumed that the radar is always illuminating an object from the positive z-direction.

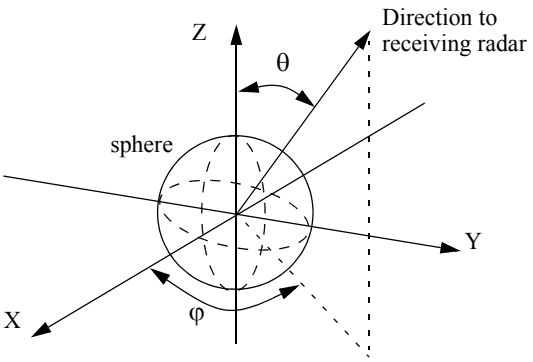

Figure 11.15. Direction of antenna receiving backscattered waves.

11.5.1. 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

$$\frac{\sigma}{\pi r^2} = \left(\frac{j}{kr}\right)\sum_{n=1}^{\infty}(-1)^n(2n+1)\left[\left(\frac{krJ_{n-1}(kr)-nJ_n(kr)}{krH_{n-1}^{(1)}(kr)-nH_n^{(1)}(kr)}\right)-\left(\frac{J_n(kr)}{H_n^{(1)}(kr)}\right)\right] \tag{11.27}$$

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

$$H_n^{(1)}(kr) = J_n(kr) + jY_n(kr) \tag{11.28}$$

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. 11.16a and 11.16b. These plots can be reproduced using the function *"rcs_sphere.m"* given in Listing 11.4 in Section 11.9.

In Fig. 11.16, three regions are identified. First is the optical region (corresponds to a large sphere). In this case,

$$\sigma = \pi r^2 \qquad r \gg \lambda \tag{11.29}$$

Second is the Rayleigh region (small sphere). In this case,

$$\sigma \approx 9\pi r^2 (kr)^4 \qquad r \ll \lambda \tag{11.30}$$

The region between the optical and Rayleigh regions is oscillatory in nature and is called the Mie or resonance region.

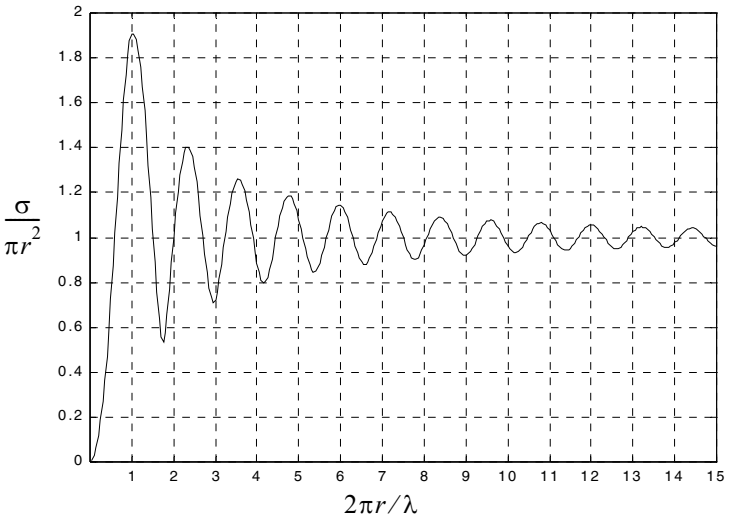

Figure 11.16a. Normalized backscattered RCS for a perfectly conducting sphere.

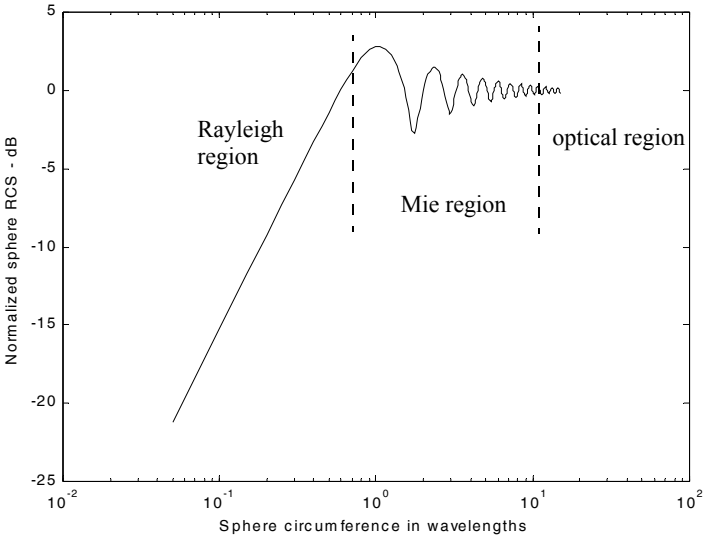

Figure 11.16b. Normalized backscattered RCS for a perfectly conducting sphere using semi-log scale.

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.

11.5.2. Ellipsoid

An ellipsoid centered at (0,0,0) is shown in Fig. 11.17. It is defined by the following equation:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1 \tag{11.31}$$

One widely accepted approximation for the ellipsoid backscattered RCS is given by

$$\sigma = \frac{\pi a^2 b^2 c^2}{\left(a^2(\sin\theta)^2(\cos\varphi)^2 + b^2(\sin\theta)^2(\sin\varphi)^2 + c^2(\cos\theta)^2\right)^2} \tag{11.32}$$

When $a = b$, the ellipsoid becomes roll symmetric. Thus, the RCS is independent of φ, and Eq. (11.32) is reduced to

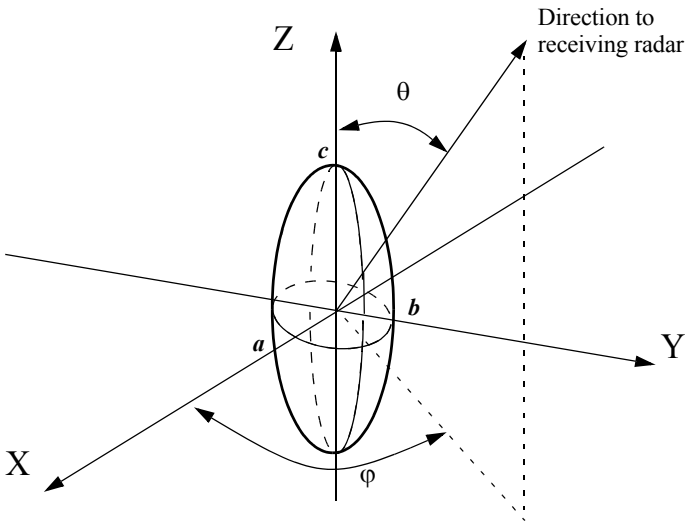

Figure 11.17. Ellipsoid.

$$\sigma = \frac{\pi b^4 c^2}{(a^2(\sin\theta)^2 + c^2(\cos\theta)^2)^2} \tag{11.33}$$

and for the case when $a = b = c$,

$$\sigma = \pi c^2 \tag{11.34}$$

Note that Eq. (11.34) defines the backscattered RCS of a sphere. This should be expected, since under the condition $a = b = c$ the ellipsoid becomes a sphere. Fig. 11.18a shows the backscattered RCS for an ellipsoid versus θ for $\varphi = 45°$. This plot can be generated using MATLAB program *"fig11_18a.m"* given in Listing 11.5 in Section 11.9. 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.

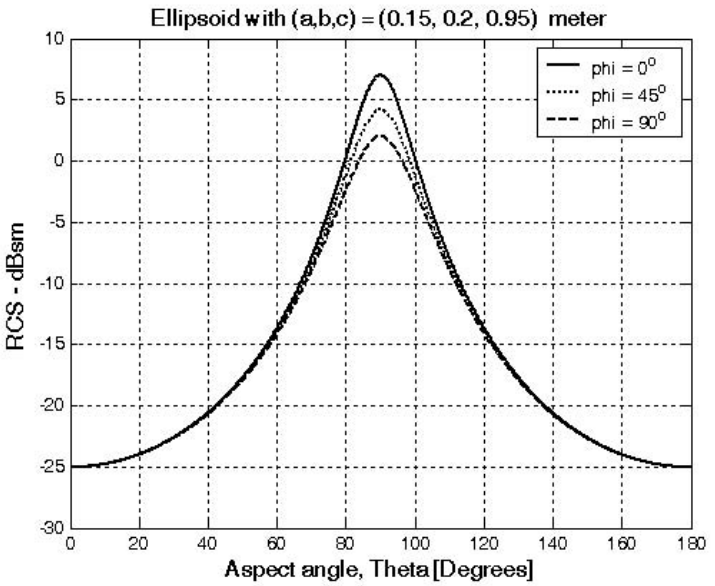

Figure 11.18a. Ellipsoid backscattered RCS versus aspect angle.

MATLAB Function "rcs_ellipsoid.m"

The function *"rcs_ellipsoid.m"* computes and plots the RCS of an ellipsoid versus aspect angle. It is given in Listing 11.6 in Section 11.9, and 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*

Fig. 11.18b shows the GUI workspace associated with function. To execute this GUI type *"rcs_ellipsoid_gui"* from the MATLAB Command window.

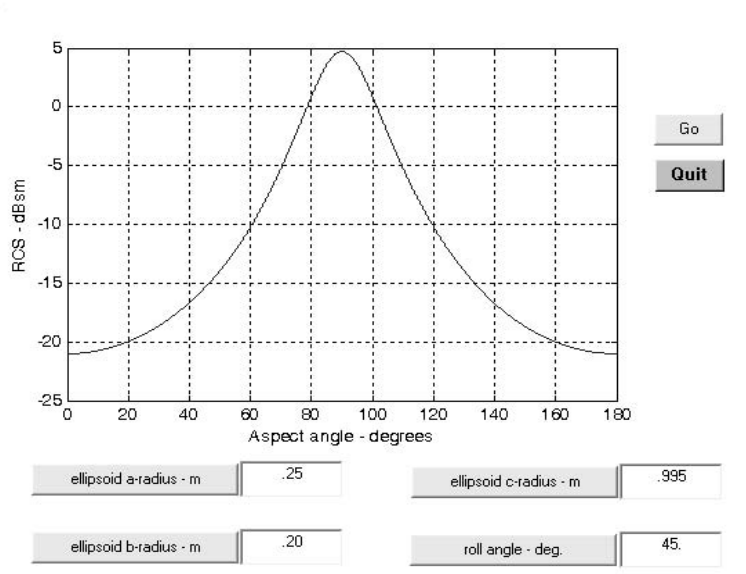

Figure 11.18b. GUI workspace associated with the function *"rcs_ellipsoid.m".*

11.5.3. Circular Flat Plate

Fig. 11.19 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

$$\sigma = \frac{4\pi^3 r^4}{\lambda^2} \qquad \theta = 0° \tag{11.35}$$

For non-normal incidence, two approximations for the circular flat plate backscattered RCS for any linearly polarized incident wave are

$$\sigma = \frac{\lambda r}{8\pi \sin\theta (\tan(\theta))^2} \tag{11.36}$$

$$\sigma = \pi k^2 r^4 \left(\frac{2J_1(2kr\sin\theta)}{2kr\sin\theta} \right)^2 (\cos\theta)^2 \tag{11.37}$$

where $k = 2\pi/\lambda$, and $J_1(\beta)$ is the first order spherical Bessel function evaluated at β. The RCS corresponding to Eqs. (11.35) through (11.37) is shown in Fig. 11.20. These plots can be reproduced using MATLAB function *"rcs_circ_gui.m"*.

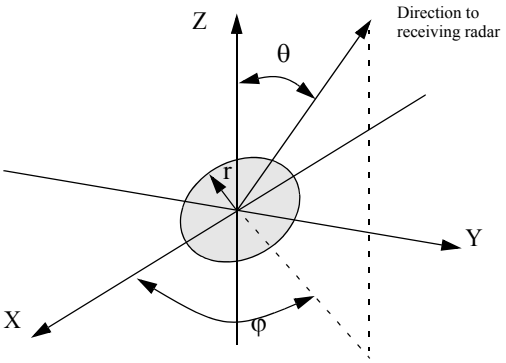

Figure 11.19. Circular flat plate.

MATLAB Function "rcs_circ_plate.m"

The function *"rcs_circ_plate.m"* calculates and plots the backscattered RCS from a circular plate. It is given in Listing 11.7 in Section 11.9; its 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*

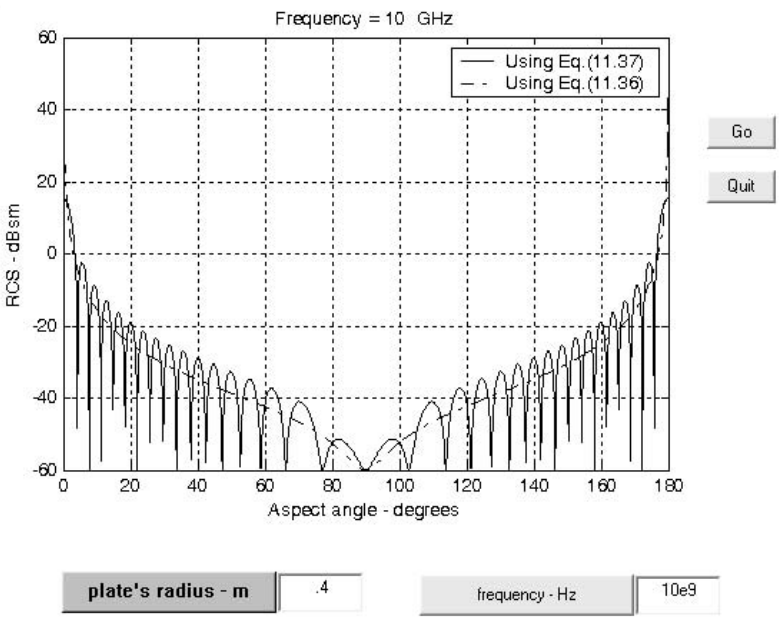

Figure 11.20. Backscattered RCS for a circular flat plate.

11.5.4. Truncated Cone (Frustum)

Figs. 11.21 and 11.22 show the geometry associated with a frustum. The half cone angle α is given by

$$\tan\alpha = \frac{(r_2 - r_1)}{H} = \frac{r_2}{L} \tag{11.38}$$

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

$$\theta_n = 90° - \alpha \tag{11.39}$$

Alternatively, normal incidence occurs at

$$\theta_n = 90° + \alpha \tag{11.40}$$

At normal incidence, one approximation for the backscattered RCS of a truncated cone due to a linearly polarized incident wave is

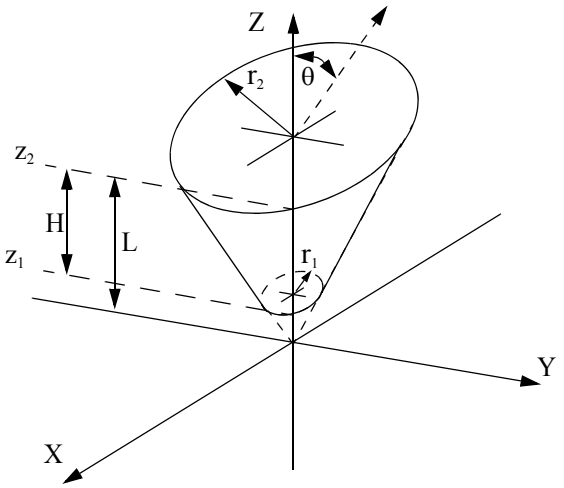

Figure 11.21. Truncated cone (frustum).

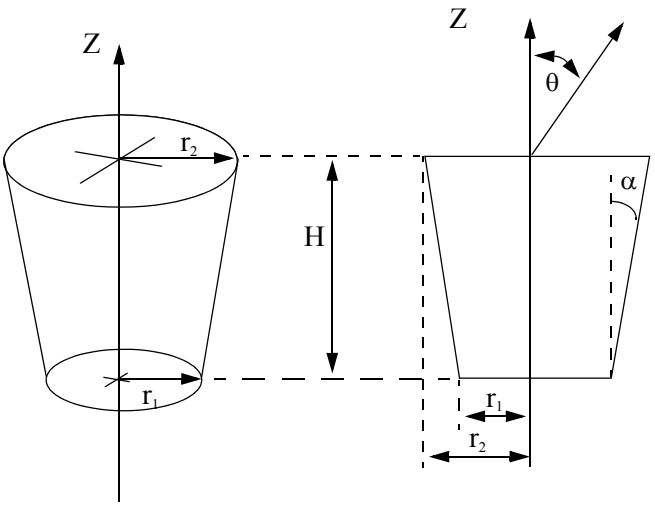

Figure 11.22. Definition of half cone angle.

$$\sigma_{\theta_n} = \frac{8\pi(z_2^{3/2} - z_1^{3/2})^2}{9\lambda \sin\theta_n} \tan\alpha(\sin\theta_n - \cos\theta_n \tan\alpha)^2 \quad \textbf{(11.41)}$$

where λ is the wavelength, and z_1, z_2 are defined in Fig. 11.21. Using trigonometric identities, Eq. (11.41) can be reduced to

$$\sigma_{\theta_n} = \frac{8\pi(z_2^{3/2} - z_1^{3/2})^2}{9\lambda} \frac{\sin\alpha}{(\cos\alpha)^4} \quad \textbf{(11.42)}$$

For non-normal incidence, the backscattered RCS due to a linearly polarized incident wave is

$$\sigma = \frac{\lambda z \tan\alpha}{8\pi \sin\theta} \left(\frac{\sin\theta - \cos\theta \tan\alpha}{\sin\theta \tan\alpha + \cos\theta} \right)^2 \quad \textbf{(11.43)}$$

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. (11.43) (assuming the radar illuminates the frustum starting from the large end) is reduced to

$$\sigma = \frac{\lambda z \tan\alpha}{8\pi \sin\theta} (\tan(\theta - \alpha))^2 \quad \textbf{(11.44)}$$

When the radar illuminates the frustum starting from the small end (i.e., the radar is in the negative z direction in Fig. 11.21), Eq. (11.44) should be modified to

$$\sigma = \frac{\lambda z \tan\alpha}{8\pi \sin\theta} (\tan(\theta + \alpha))^2 \quad \textbf{(11.45)}$$

For example, consider a frustum defined by $H = 20.945cm$, $r_1 = 2.057cm$, $r_2 = 5.753cm$. It follows that the half cone angle is $10°$. Fig. 11.23a shows a plot of its RCS when illuminated by a radar in the positive z direction. Fig. 11.23b shows the same thing, except in this case, the radar is in the negative z direction. Note that for the first case, normal incidence occur at $100°$, while for the second case it occurs at $80°$. These plots can be reproduced using MATLAB function *"rcs_frustum_gui.m"* given in Listing 11.8 in Section 11.9.

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)

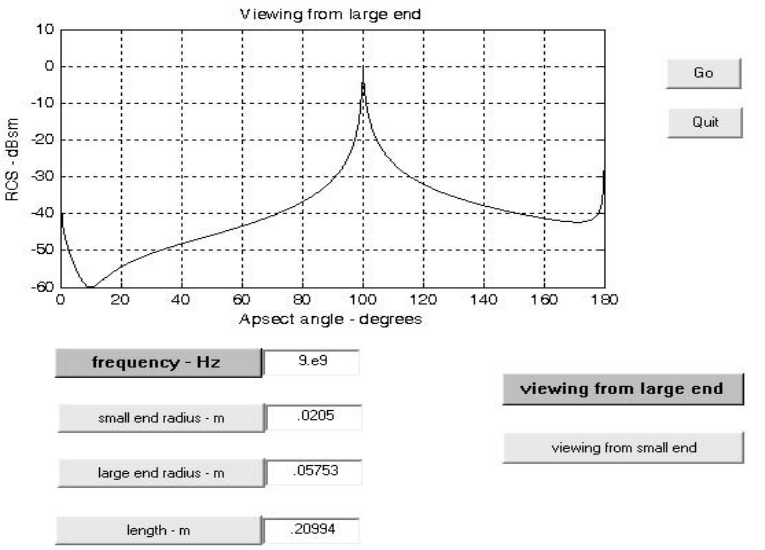

Figure 11.23a. Backscattered RCS for a frustum.

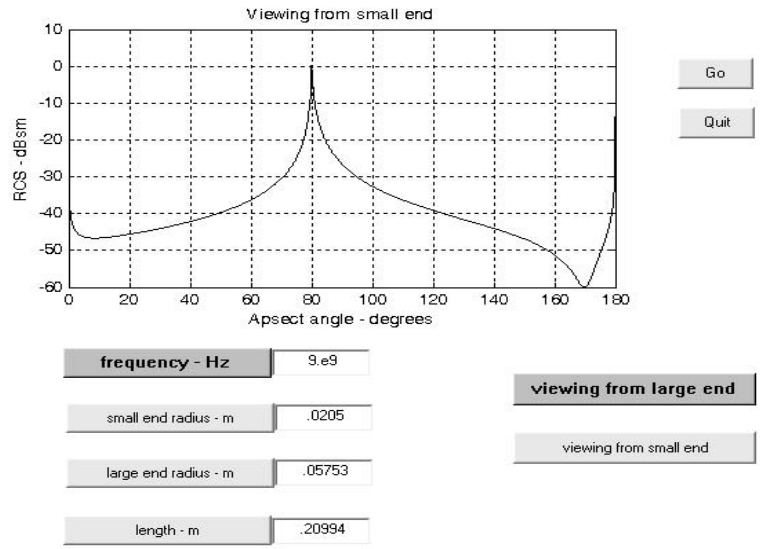

Figure 11.23b. Backscattered RCS for a frustum.

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*

11.5.5. Cylinder

Fig. 11.24 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

$$\sigma_{\theta_n} = \frac{2\pi H^2 r_2^2 r_1^2}{\lambda[r_1^2(\cos\varphi)^2 + r_2^2(\sin\varphi)^2]^{1.5}} \tag{11.46}$$

$$\sigma = \frac{\lambda r_2^2 r_1^2 \sin\theta}{8\pi\ (\cos\theta)^2[r_1^2(\cos\varphi)^2 + r_2^2(\sin\varphi)^2]^{1.5}} \tag{11.47}$$

For a circular cylinder of radius r, then due to roll symmetry, Eqs. (11.46) and (11.47), respectively, reduce to

$$\sigma_{\theta_n} = \frac{2\pi H^2 r}{\lambda} \tag{11.48}$$

$$\sigma = \frac{\lambda r \sin\theta}{8\pi(\cos\theta)^2} \tag{11.49}$$

Fig. 11.25a shows a plot of the cylinder backscattered RCS for a symmetrical cylinder. Fig. 11.25b shows the backscattered RCS for an elliptical cylinder. These plots can be reproduced using MATLAB function *"rcs_cylinder.m"* given in Listing 11.9 in Section 11.9.

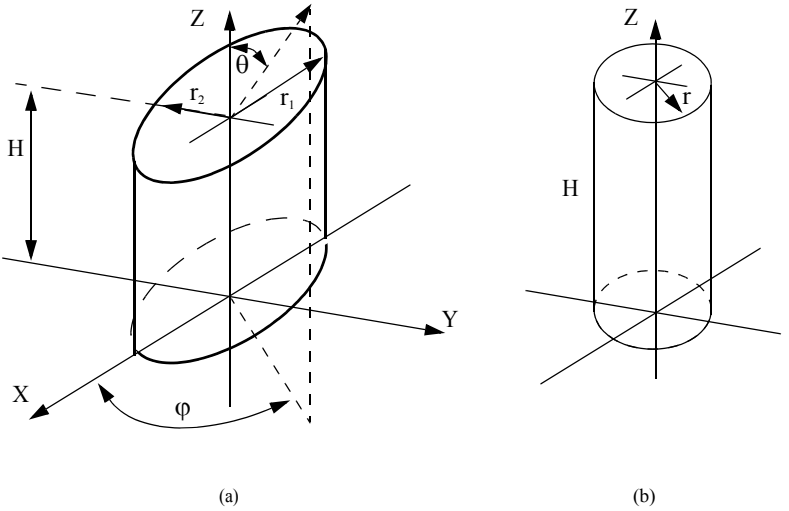

Figure 11.24. (a) Elliptical cylinder; (b) circular cylinder.

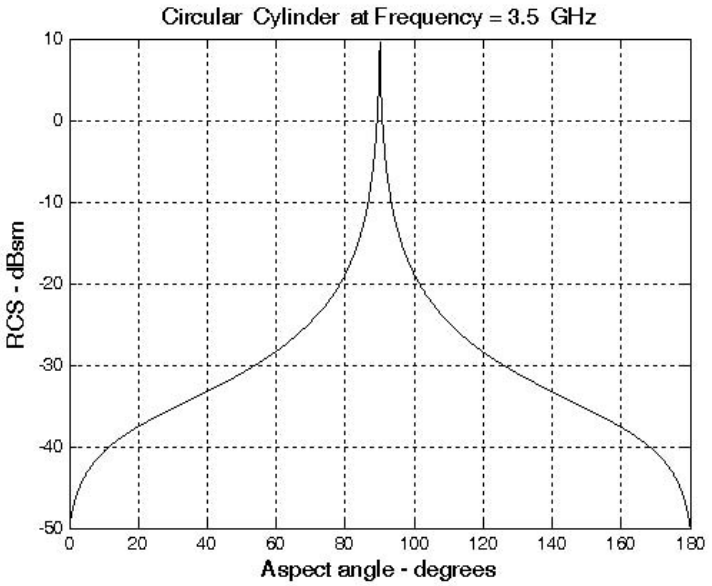

Figure 11.25a. Backscattered RCS for a symmetrical cylinder, $r = 0.125m$ **and** $H = 1m$ **.**

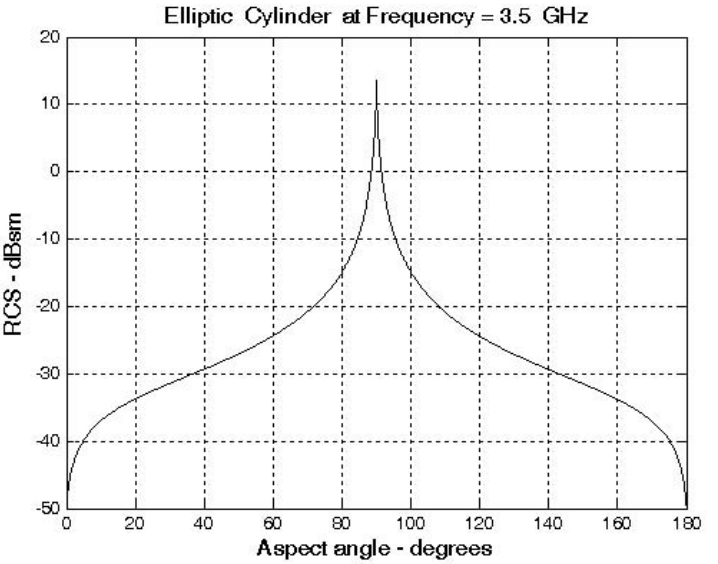

Figure 11.25b. Backscattered RCS for an elliptical cylinder, $r_1 = 0.125m$**,** $r_2 = 0.05m$**, and** $H = 1m$**.**

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*
CylinderType	*'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*

11.5.6. Rectangular Flat Plate

Consider a perfectly conducting rectangular thin flat plate in the x-y plane as shown in Fig. 11.26. 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

$$\sigma_V = \frac{b^2}{\pi}\left|\sigma_{1V} - \sigma_{2V}\left[\frac{1}{\cos\theta} + \frac{\sigma_{2V}}{4}(\sigma_{3V} + \sigma_{4V})\right]\sigma_{5V}^{-1}\right|^2 \tag{11.50}$$

$$\sigma_H = \frac{b^2}{\pi}\left|\sigma_{1H} - \sigma_{2H}\left[\frac{1}{\cos\theta} - \frac{\sigma_{2H}}{4}(\sigma_{3H} + \sigma_{4H})\right]\sigma_{5H}^{-1}\right|^2 \tag{11.51}$$

where $k = 2\pi/\lambda$ and

$$\sigma_{1V} = \cos(k a\sin\theta) - j\frac{\sin(k a\sin\theta)}{\sin\theta} = (\sigma_{1H})^* \tag{11.52}$$

$$\sigma_{2V} = \frac{e^{j(ka - \pi/4)}}{\sqrt{2\pi}(ka)^{3/2}} \tag{11.53}$$

$$\sigma_{3V} = \frac{(1 + \sin\theta)e^{-jk a\sin\theta}}{(1 - \sin\theta)^2} \tag{11.54}$$

$$\sigma_{4V} = \frac{(1 - \sin\theta)e^{jk a\sin\theta}}{(1 + \sin\theta)^2} \tag{11.55}$$

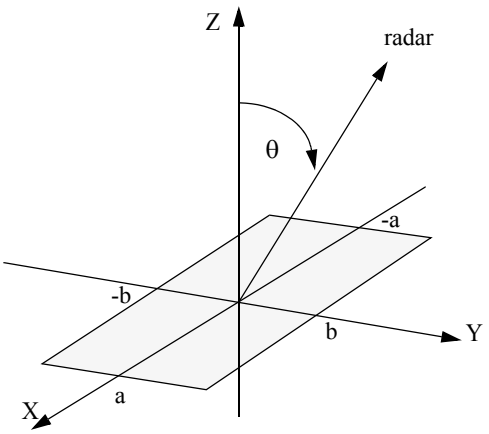

Figure 11.26. Rectangular flat plate.

$$\sigma_{5V} = 1 - \frac{e^{j(2ka - \pi/2)}}{8\pi(ka)^3} \tag{11.56}$$

$$\sigma_{2H} = \frac{4e^{j(ka + \pi/4)}}{\sqrt{2\pi}(ka)^{1/2}} \tag{11.57}$$

$$\sigma_{3H} = \frac{e^{-jk\sin\theta}}{1 - \sin\theta} \tag{11.58}$$

$$\sigma_{4H} = \frac{e^{jk\sin\theta}}{1 + \sin\theta} \tag{11.59}$$

$$\sigma_{5H} = 1 - \frac{e^{j(2ka + (\pi/2))}}{2\pi(ka)} \tag{11.60}$$

Eqs. (11.50) and (11.51) 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

$$\sigma_H \to 0$$

$$\sigma_V = \frac{ab^2}{\lambda}\left\{\left[1 + \frac{\pi}{2(2a/\lambda)^2}\right] + \left[1 - \frac{\pi}{2(2a/\lambda)^2}\right]\cos\left(2ka - \frac{3\pi}{5}\right)\right\} \tag{11.61}$$

The backscattered RCS for a perfectly conducting thin rectangular plate for incident waves at any θ, φ can be approximated by

$$\sigma = \frac{4\pi a^2 b^2}{\lambda^2}\left(\frac{\sin(ak\sin\theta\cos\varphi)}{ak\sin\theta\cos\varphi}\ \frac{\sin(bk\sin\theta\sin\varphi)}{bk\sin\theta\sin\varphi}\right)^2(\cos\theta)^2 \tag{11.62}$$

Eq. (11.62) is independent of the polarization, and is only valid for aspect angles $\theta \le 20°$. Fig. 11.27 shows an example for the backscattered RCS of a rectangular flat plate, for both vertical (Fig. 11.27a) and horizontal (Fig. 11.27b) polarizations, using Eqs. (11.50), (11.51), and (11.62). In this example, $a = b = 10.16cm$ and wavelength $\lambda = 3.33cm$. This plot can be reproduced using MATLAB function *"rcs_rect_plate"* given in Listing 11.10.

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)

1. Ross, R. A., Radar Cross Section of Rectangular Flat Plate as a Function of Aspect Angle, *IEEE Trans*., AP-14,320, 1966.

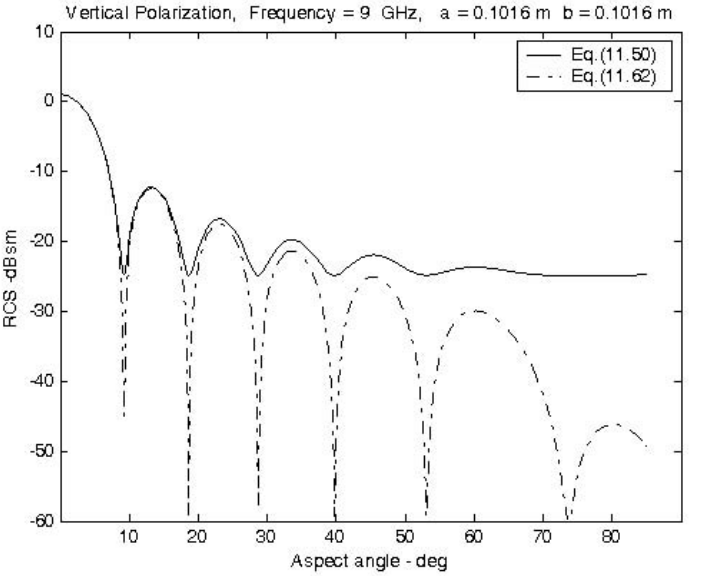

Figure 11.27a. Backscattered RCS for a rectangular flat plate.

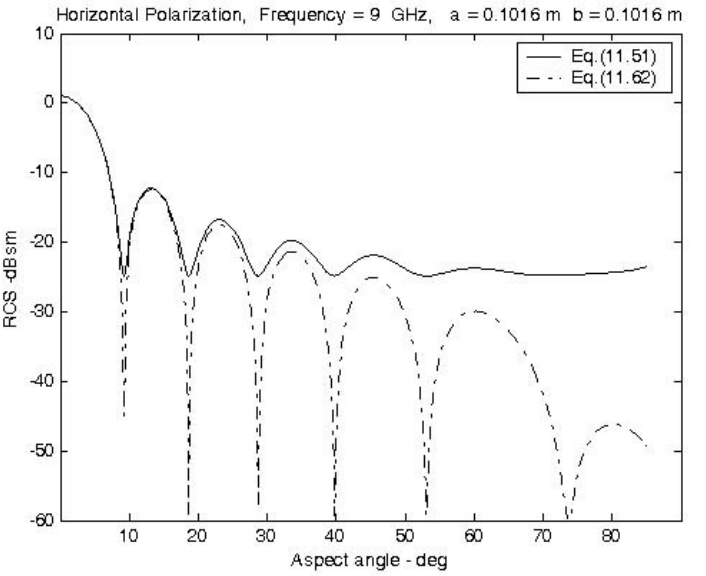

Figure 11.27b. Backscattered RCS for a rectangular flat plate.

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*

Fig. 11.27c shows the GUI workspace associated with this function.

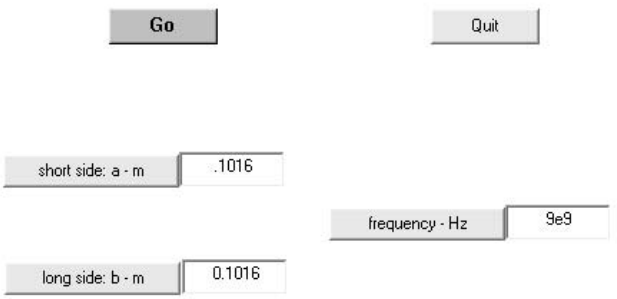

Figure 11.27c. GUI workspace associated with the function *"rcs_rect_plate.m"*.

11.5.7. Triangular Flat Plate

Consider the triangular flat plate defined by the isosceles triangle as oriented in Fig. 11.28. The backscattered RCS can be approximated for small aspect angles ($\theta \leq 30°$) by

$$\sigma = \frac{4\pi A^2}{\lambda^2}(\cos\theta)^2\sigma_0 \tag{11.63}$$

$$\sigma_0 = \frac{[(\sin\alpha)^2 - (\sin(\beta/2))^2]^2 + \sigma_{01}}{\alpha^2 - (\beta/2)^2} \tag{11.64}$$

$$\sigma_{01} = 0.25(\sin\varphi)^2[(2a/b)\cos\varphi\sin\beta - \sin\varphi\sin 2\alpha]^2 \tag{11.65}$$

where $\alpha = k\sin\theta\cos\varphi$, $\beta = kb\sin\theta\sin\varphi$, and $A = ab/2$. For waves incident in the plane $\varphi = 0$, the RCS reduces to

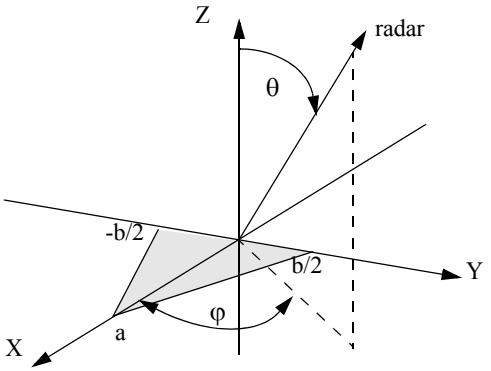

Figure 11.28. Coordinates for a perfectly conducting isosceles triangular plate.

$$\sigma = \frac{4\pi A^2}{\lambda^2}(\cos\theta)^2\left[\frac{(\sin\alpha)^4}{\alpha^4} + \frac{(\sin 2\alpha - 2\alpha)^2}{4\alpha^4}\right] \tag{11.66}$$

and for incidence in the plane $\varphi = \pi/2$

$$\sigma = \frac{4\pi A^2}{\lambda^2}(\cos\theta)^2\left[\frac{(\sin(\beta/2))^4}{(\beta/2)^4}\right] \tag{11.67}$$

Fig. 11.29 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 function *"rcs_isosceles.m"* given in Listing 11.11 in Section 11.9.

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*

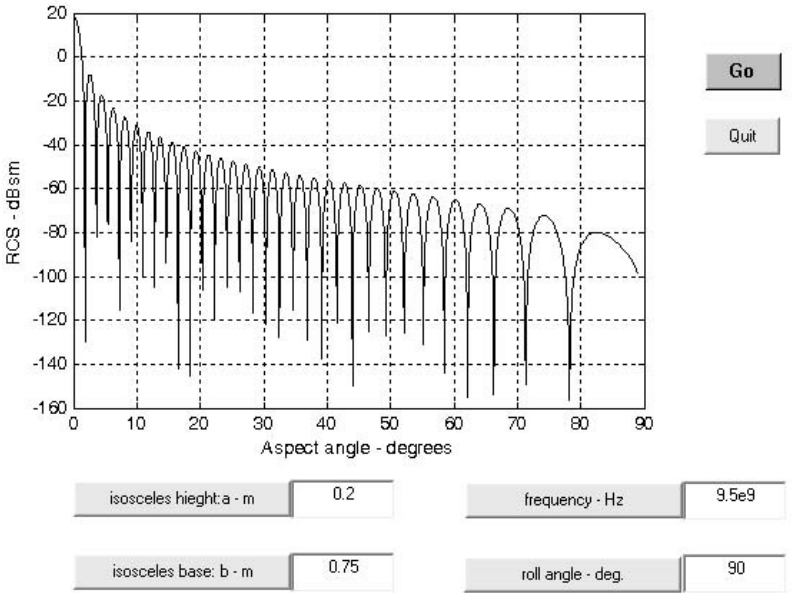

Figure 11.29. Backscattered RCS for a perfectly conducting triangular flat plate, $a = 20cm$ and $b = 75cm$.

11.6. Scattering From a Dielectric-Capped Wedge

The geometry of a dielectric-capped wedge is shown in Fig. 11.30. It is required to find to the field expressions for the problem of scattering by a 2-D perfect electric conducting (PEC) wedge capped with a dielectric cylinder. Using the cylindrical coordinates system, the excitation due to an electric line current of complex amplitude I_0 located at (ρ_0, φ_0) results in TM^z incident field with the electric field expression given by

$$E_z^i = -I_e \frac{\omega\mu_0}{4} H_0^{(2)}\left(k\left|\boldsymbol{\rho} - \boldsymbol{\rho}_0\right|\right) \qquad \textbf{(11.68)}$$

The problem is divided into three regions, I, II, and III shown in Fig. 11.30. The field expressions may be assumed to take the following forms:

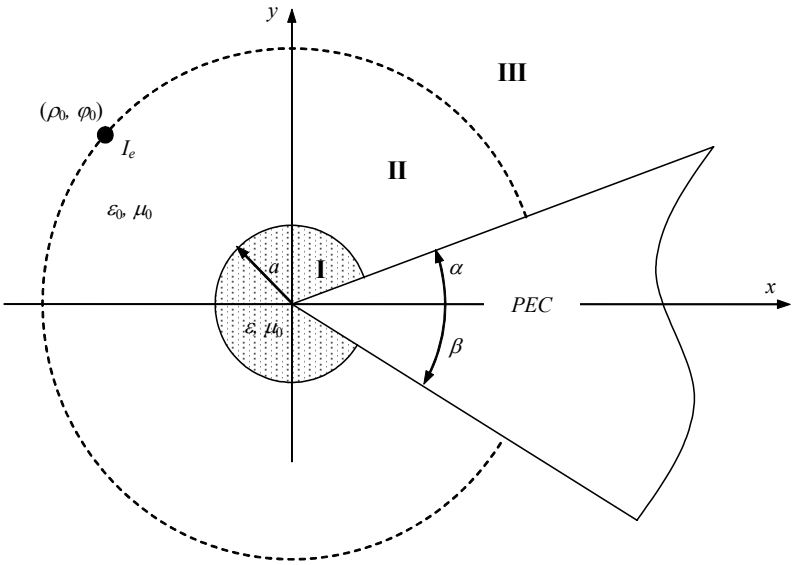

Figure 11.30.Scattering from dielectric-capped wedge.

$$
\begin{aligned}
E_z^{\mathrm{I}} &= \sum_{n=0}^{\infty} a_n J_\nu\left(k_1\rho\right)\sin\nu\left(\varphi-\alpha\right)\sin\nu\left(\varphi_0-\alpha\right) \\
E_z^{\mathrm{II}} &= \sum_{n=0}^{\infty}\left(b_n J_\nu\left(k\rho\right)+c_n H_\nu^{(2)}\left(k\rho\right)\right)\sin\nu\left(\varphi-\alpha\right)\sin\nu\left(\varphi_0-\alpha\right) \\
E_z^{\mathrm{III}} &= \sum_{n=0}^{\infty} d_n H_\nu^{(2)}\left(k\rho\right)\sin\nu\left(\varphi-\alpha\right)\sin\nu\left(\varphi_0-\alpha\right)
\end{aligned}
\qquad \textbf{(11.69)}
$$

where

$$
\nu = \frac{n\pi}{2\pi-\alpha-\beta} \qquad \textbf{(11.70)}
$$

while $J_\nu(x)$ is the Bessel function of order ν and argument x and $H_\nu^{(2)}$ is the Hankel function of the second kind of order ν and argument x. From Maxwell's equations, the magnetic field component H_φ is related to the electric field component E_z for a TM^z wave by

$$
H_\varphi = \frac{1}{j\omega\mu}\frac{\partial E_z}{\partial\rho} \qquad \textbf{(11.71)}
$$

Thus, the magnetic field component H_φ in the various regions may be written as

$$\begin{aligned} H_\varphi^{\mathrm{I}} &= \frac{k_1}{j\omega\mu_0}\sum_{n=0}^{\infty} a_n J_\nu'(k_1\rho)\sin\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha) \\ H_\varphi^{\mathrm{II}} &= \frac{k}{j\omega\mu_0}\sum_{n=0}^{\infty}\left(b_n J_\nu'(k\rho)+c_n H_\nu^{(2)\prime}(k\rho)\right)\sin\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha) \\ H_\varphi^{\mathrm{III}} &= \frac{k}{j\omega\mu_0}\sum_{n=0}^{\infty} d_n H_\nu^{(2)\prime}(k\rho)\sin\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha) \end{aligned} \tag{11.72}$$

Where the prime indicated derivatives with respect to the full argument of the function. The boundary conditions require that the tangential electric field components vanish at the PEC surface. Also, the tangential field components should be continuous across the air-dielectric interface and the virtual boundary between region II and III, except for the discontinuity of the magnetic field at the source point. Thus,

$$E_z = 0 \quad \text{at} \qquad \varphi = \alpha, 2\pi-\beta \tag{11.73}$$

$$\left.\begin{aligned} E_z^{\mathrm{I}} &= E_z^{\mathrm{II}} \\ H_\varphi^{\mathrm{I}} &= H_\varphi^{\mathrm{II}} \end{aligned}\right\} \quad \text{at} \qquad \rho = a \tag{11.74}$$

$$\left.\begin{aligned} E_z^{\mathrm{II}} &= E_z^{\mathrm{III}} \\ H_\varphi^{\mathrm{II}} - H_\varphi^{\mathrm{III}} &= -J_e \end{aligned}\right\} \quad \text{at} \qquad \rho = \rho_0 \tag{11.75}$$

The current density J_e may be given in Fourier series expansion as

$$J_e = \frac{I_e}{\rho_0}\delta(\varphi-\varphi_0) = \frac{2}{2\pi-\alpha-\beta}\frac{I_e}{\rho_0}\sum_{n=0}^{\infty}\sin\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha) \tag{11.76}$$

The boundary condition on the PEC surface is automatically satisfied by the φ dependence of the electric field Eq. (11.72). From the boundary conditions in Eq. (11.73)

$$\begin{aligned} &\sum_{n=0}^{\infty} a_n J_\nu(k_1 a)\sin\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha) = \\ &\qquad \sum_{n=0}^{\infty}\left(b_n J_\nu(ka)+c_n H_\nu^{(2)}(ka)\right)\sin\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha) \end{aligned} \tag{11.77}$$

$$\frac{k_1}{j\omega\mu_0}\sum_{n=0}^{\infty} a_n J_v'(k_1 a)\sin v(\varphi-\alpha)\sin v(\varphi_0-\alpha) = \frac{k}{j\omega\mu_0}\sum_{n=0}^{\infty}\left(b_n J_v'(ka)+c_n H_v^{(2)\prime}(ka)\right)\sin v(\varphi-\alpha)\sin v(\varphi_0-\alpha) \tag{11.78}$$

From the boundary conditions in Eq. (11.75), we have

$$\sum_{n=0}^{\infty}\left(b_n J_v(k\rho_0)+c_n H_v^{(2)}(k\rho_0)\right)\sin v(\varphi-\alpha)\sin v(\varphi_0-\alpha) = \sum_{n=0}^{\infty} d_n H_v^{(2)}(k\rho_0)\sin v(\varphi-\alpha)\sin v(\varphi_0-\alpha) \tag{11.79}$$

$$\frac{k}{j\omega\mu_0}\sum_{n=0}^{\infty}\left(b_n J_v'(k\rho_0)+c_n H_v^{(2)\prime}(k\rho_0)\right)\sin v(\varphi-\alpha)\sin v(\varphi_0-\alpha) = \frac{k}{j\omega\mu_0}\sum_{n=0}^{\infty} d_n H_v^{(2)\prime}(k\rho_0)\sin v(\varphi-\alpha)\sin v(\varphi_0-\alpha) - \frac{2}{2\pi-\alpha-\beta}\frac{I_e}{\rho_0}\sum_{n=0}^{\infty}\sin v(\varphi-\alpha)\sin v(\varphi_0-\alpha) \tag{11.80}$$

Since Eqs. (11.77) and (11.80) hold for all φ, the series on the left and right hand sides should be equal term by term. More precisely,

$$a_n J_v(k_1 a) = b_n J_v(ka) + c_n H_v^{(2)}(ka) \tag{11.81}$$

$$\frac{k_1}{\mu_0} a_n J_v'(k_1 a) = \frac{k}{\mu_0}\left(b_n J_v'(ka) + c_n H_v^{(2)\prime}(ka)\right) \tag{11.82}$$

$$b_n J_v(k\rho_0) + c_n H_v^{(2)}(k\rho_0) = d_n H_v^{(2)}(k\rho_0) \tag{11.83}$$

$$b_n J_v'(k\rho_0) + c_n H_v^{(2)\prime}(k\rho_0) = d_n H_v^{(2)\prime}(k\rho_0) - \frac{2j\eta_0}{2\pi-\alpha-\beta}\frac{I_e}{\rho_0} \tag{11.84}$$

From Eqs. (11.81) and (11.83), we have

$$a_n = \frac{1}{J_v(k_1 a)}\left[b_n J_v(ka) + c_n H_v^{(2)}(ka)\right] \tag{11.85}$$

$$d_n = c_n + b_n \frac{J_v(k\rho_0)}{H_v^{(2)}(k\rho_0)} \tag{11.86}$$

Multiplying Eq. (11.83) by $H_\nu^{(2)'}$ and Eq. (11.84) by $H_\nu^{(2)}$, and by subtraction and using the Wronskian of the Bessel and Hankel functions, we get

$$b_n = -\frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta} H_\nu^{(2)}\left(k\rho_0\right) \tag{11.87}$$

Substituting b_n in Eqs. (11.81) and (11.82) and solving for c_n yield

$$c_n = \frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\left[H_\nu^{(2)}\left(k\rho_0\right)\frac{kJ_\nu'\left(ka\right)J_\nu\left(k_1a\right)-k_1J_\nu\left(ka\right)J_\nu'\left(k_1a\right)}{kH_\nu^{(2)'}\left(ka\right)J_\nu\left(k_1a\right)-k_1H_\nu^{(2)}\left(ka\right)J_\nu'\left(k_1a\right)}\right] \tag{11.88}$$

From Eqs. (11.86) through (11.88), d_n may be given by

$$d_n = \frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\left[H_\nu^{(2)}\left(k\Im_0\right)\frac{kJ_\nu'\left(ka\right)J_\nu\left(k_1a\right)-k_1J_\nu\left(ka\right)J_\nu'\left(k_1a\right)}{kH_\nu^{(2)'}\left(ka\right)J_\nu\left(k_1a\right)-k_1H_\nu^{(2)}\left(ka\right)J_\nu'\left(k_1a\right)}-J_\nu\left(k\rho_0\right)\right] \tag{11.89}$$

which can be written as

$$d_n = \frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\left\{\frac{\begin{array}{r}kJ_\nu\left(k_1a\right)\left[J_\nu'\left(ka\right)H_\nu^{(2)}\left(k\rho_0\right)-H_\nu^{(2)'}\left(ka\right)J_\nu\left(k\rho_0\right)\right]+\mathrm{K}\\ k_1J_\nu'\left(k_1a\right)\left[H_\nu^{(2)}\left(ka\right)J_\nu\left(k\rho_0\right)-J_\nu\left(ka\right)H_\nu^{(2)}\left(k\rho_0\right)\right]\end{array}}{kH_\nu^{(2)'}\left(ka\right)J_\nu\left(k_1a\right)-k_1H_\nu^{(2)}\left(ka\right)J_\nu'\left(k_1a\right)}\right\} \tag{11.90}$$

Substituting for the Hankel function in terms of Bessel and Neumann functions, Eq. (11.90) reduces to

$$d_n = -j\frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\left\{\frac{\begin{array}{r}kJ_\nu\left(k_1a\right)\left[J_\nu'\left(ka\right)Y_\nu\left(k\rho_0\right)-Y_\nu'\left(ka\right)J_\nu\left(k\rho_0\right)\right]+\mathrm{K}\\ k_1J_\nu'\left(k_1a\right)\left[Y_\nu\left(ka\right)J_\nu\left(k\rho_0\right)-J_\nu\left(ka\right)Y_\nu\left(k\rho_0\right)\right]\end{array}}{kH_\nu^{(2)'}\left(ka\right)J_\nu\left(k_1a\right)-k_1H_\nu^{(2)}\left(ka\right)J_\nu'\left(k_1a\right)}\right\} \tag{11.91}$$

With these closed form expressions for the expansion coeffiecients a_n, b_n, c_n and d_n, the field components E_z and H_φ can be determined from Eq. (11.69) and Eq. (11.72), respectively. Alternatively, the magnetic field component H_ρ can be computed from

$$H_\rho = -\frac{1}{j\omega\mu}\frac{1}{\rho}\frac{\partial E_z}{\partial\phi} \tag{11.92}$$

Thus, the H_ρ expressions for the three regions defined in Fig. 11.30 become

$$H_\rho^{\mathrm{I}} = -\frac{1}{j\omega\mu\rho}\sum_{n=0}^{\infty} a_n \nu J_\nu(k_1\rho)\cos\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha)$$

$$H_\rho^{\mathrm{II}} = -\frac{1}{j\omega\mu\rho}\sum_{n=0}^{\infty} \nu\left(b_n J_\nu(k\rho)+c_n H_\nu^{(2)}(k\rho)\right)\cos\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha) \qquad (11.93)$$

$$H_\rho^{\mathrm{III}} = -\frac{1}{j\omega\mu\rho}\sum_{n=0}^{\infty} d_n \nu H_\nu^{(2)}(k\rho)\cos\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha)$$

11.6.1. Far Scattered Field

In region III, the scattered field may be found as the difference between the total and incident fields. Thus, using Eqs. (11.68) and (11.69) and considering the far field condition ($\rho \to \infty$) we get

$$E_z^{\mathrm{III}} = E_z^i + E_z^s = \sqrt{\frac{2j}{\pi k\rho}}e^{-jk\rho}\sum_{n=0}^{\infty} d_n j^\nu \sin\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha)$$
$$E_z^i = -I_e\frac{\omega\mu_0}{4}\sqrt{\frac{2j}{\pi k\rho}}e H_\rho = -\frac{1}{j\omega\mu}\frac{1}{\rho}\frac{\partial E_z}{\partial\phi} \qquad (11.94)$$

Note that d_n can be written as

$$d_n = -\frac{\omega\mu_0 I_e}{4}\,\%d_n \qquad (11.95)$$

where

$$\%d_n = j\frac{4\pi}{2\pi-\alpha-\beta}\left\{\frac{kJ_\nu(k_1a)\left[J_\nu'(ka)Y_\nu(k\rho_0)-Y_\nu'(ka)J_\nu(k\rho_0)\right]+\mathrm{K}}{kH_\nu^{(2)\prime}(ka)J_\nu(k_1a)-k_1H_\nu^{(2)}(ka)J_\nu'(k_1a)}\right\} \qquad (11.96)$$

Substituting Eq. (11.95) into Eq. (11.94), the scattered field $f(\varphi)$ is

$$E_z^s = \frac{-\omega\mu_0 I_e}{4}\sqrt{\frac{2j}{\pi k\rho}}e^{-jk\rho} \qquad (11.97)$$

$$\left(\sum_{n=0}^{\infty} d_n j^{\nu}\sin\nu(\varphi-\alpha)\sin\nu(\varphi_0-\alpha) - e^{jk\rho_0\cos(\varphi-\varphi_0)}\right)$$

11.6.2. Plane Wave Excitation

For plane wave excitation ($\rho_0 \to \infty$), the expression in Eqs. (11.87) and (11.88) reduce to

$$\begin{aligned} b_n &= -\frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\, j^{\nu}\sqrt{\frac{2j}{\pi k\rho_0}}e^{-jk\rho_0} \\ c_n &= \frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\, j^{\nu}\sqrt{\frac{2j}{\pi k\rho_0}}e^{-jk\rho_0}\,\frac{kJ_{\nu}'(ka)J_{\nu}(k_1a)-k_1J_{\nu}(ka)J_{\nu}'(k_1a)}{kH_{\nu}^{(2)\prime}(ka)J_{\nu}(k_1a)-k_1H_{\nu}^{(2)}(ka)J_{\nu}'(k_1a)} \end{aligned} \qquad (11.98)$$

where the complex amplitude of the incident plane wave, E_0, can be given by

$$E_0 = -I_e\frac{\omega\mu_0}{4}\sqrt{\frac{2j}{\pi k\rho_0}}e^{-jk\rho_0} \qquad (11.99)$$

In this case, the field components can be evaluated in regions I and II only.

11.6.3. Special Cases

Case I: $\alpha = \beta$ (reference at bisector); The definition of ν reduces to

$$\nu = \frac{n\pi}{2(\pi-\beta)} \qquad (11.100)$$

and the same expression will hold for the coefficients (with $\alpha = \beta$).

Case II: $\alpha = 0$ (reference at face); the definition of ν takes on the form

$$\nu = \frac{n\pi}{2\pi-\beta} \qquad (11.101)$$

and the same expression will hold for the coefficients (with $\alpha = 0$).

Case III: $k_1 \to \infty$ (PEC cap); Fields at region I will vanish, and the coefficients will be given by

$$
\begin{aligned}
b_n &= -\frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta} H_\nu^{(2)}\left(k\rho_0\right) \\
c_n &= \frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta} H_\nu^{(2)}\left(k\rho_0\right)\frac{J_\nu\left(ka\right)}{H_\nu^{(2)}\left(ka\right)} \\
d_n &= j\frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\frac{Y_\nu\left(ka\right)J_\nu\left(k\rho_0\right)-J_\nu\left(ka\right)Y_\nu\left(k\rho_0\right)}{H_\nu^{(2)}\left(ka\right)} \\
a_n &= \frac{1}{J_\nu\left(k_1 a\right)}\left[b_n J_\nu\left(ka\right)+c_n H_\nu^{(2)}\left(ka\right)\right]=0
\end{aligned}
\tag{11.102}
$$

Note that the expressions of b_n and c_n will yield zero tangential electric field at $\rho = a$ when substituted in Eq.(11.69).

Case IV: $a \to 0$ (no cap); The expressions of the coefficients in this case may be obtained by setting $k_1 = k$, or by taking the limit as a approaches zero. Thus,

$$
\begin{aligned}
c_n &= \frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\left[H_\nu^{(2)}\left(k\rho_0\right)\frac{kJ_\nu'\left(ka\right)J_\nu\left(ka\right)-kJ_\nu\left(ka\right)J_\nu'\left(ka\right)}{kH_\nu^{(2)\prime}\left(ka\right)J_\nu\left(ka\right)-kH_\nu^{(2)}\left(ka\right)J_\nu'\left(ka\right)}\right]=0 \\
b_n &= -\frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}H_\nu^{(2)}\left(k\rho_0\right) \\
a_n &= \frac{1}{J_\nu\left(ka\right)}\left[b_n J_\nu\left(ka\right)+c_n H_\nu^{(2)}\left(ka\right)\right]=b_n \\
d_n &= \frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}\left\{\begin{array}{r}kJ_\nu\left(k_1 a\right)\left[J_\nu'\left(ka\right)H_\nu^{(2)}\left(k\rho_0\right)-H_\nu^{(2)\prime}\left(ka\right)J_\nu\left(k\rho_0\right)\right]+\mathrm{K} \\ \dfrac{k_1 J_\nu'\left(k_1 a\right)\left[H_\nu^{(2)}\left(ka\right)J_\nu\left(k\rho_0\right)-J_\nu\left(ka\right)H_\nu^{(2)}\left(k\rho_0\right)\right]}{kH_\nu^{(2)\prime}\left(ka\right)J_\nu\left(k_1 a\right)-k_1 H_\nu^{(2)}\left(ka\right)J_\nu'\left(k_1 a\right)}\end{array}\right\} \\
&= -\frac{\pi\omega\mu_0 I_e}{2\pi-\alpha-\beta}J_\nu\left(k\rho_0\right)
\end{aligned}
\tag{11.103}
$$

Case V: $a \to 0$ and $\alpha = \beta = 0$ (semi-infinite PEC plane); In this case, the coefficients in Eq. (11.103) become valid with the exception that the values of ν reduce to $n/2$. Once, the electric field component E_z in the different regions is computed, the corresponding magnetic field component H_φ can be computed using Eq. (11.71) and the magnetic field component H_ρ may be computed as

$$
H_\rho = -\frac{1}{j\omega\mu}\frac{1}{\rho}\frac{\partial E_z}{\partial\phi} \tag{11.104}
$$

MATLAB Program "Capped_WedgeTM.m"

The MATLAB program *"Capped_WedgeTM.m"* given in listing 11.12, along with the following associated functions *"DielCappedWedgeTMFields_Ls.m"*, *"DielCappedWedgeTMFields_PW"*, *"polardb.m"*, *"dbesselj.m"*, *"dbesselh.m",* and *"dbessely.m"* given in the following listings, calculates and plots the far field of a capped wedge in the presence of an electric line source field. The near field distribution is also computed for both line source or plane wave excitation. All near field components are computed and displayed, in separate windows, using 3-D output format. The program is also capable of analyzing the field variations due to the cap parameters. The user can execute this MATLAB program from the MATLAB command window and manually change the input parameters in the designated section in the program in order to perform the desired analysis. Alternatively, the *"Capped_Wedge_GUI.m"* function along with the *"Capped_Wedge_GUI.fig"* file can be used to simplify the data entry procedure.

A sample of the data entry screen of the *"Capped_Wedge_GUI"* program is shown in Fig. 11.31 for the case of a line source exciting a sharp conducting wedge. The corresponding far field pattern is shown in Fig. 11.32. When keeping all the parameters in Fig. 11.31 the same except that selecting a dielectric or conducting cap, one obtains the far field patterns in Figs. 11.33 and 11.34, respectively. It is clear from these figures how the cap parameters affect the direction of the maximum radiation of the line source in the presence of the wedge. The distribution of the components of the fields in the near field for these three cases (sharp edge, dielectric capped edge, and conducting capped edge) is computed and shown in Figs. 11.35 to 11.43. The near field distribution for an incident plane wave field on these three types of wedges is also computed and shown in Figs. 11.44 to 11.52. These near field distributions clearly demonstrated the effect or cap parameters in altering the sharp edge singular behavior. To further illustrate this effect, the following set of figures (Figs. (11.53) to (11.55)) presents the near field of the electric component of plane wave incident on a half plane with a sharp edge, dielectric capped edge, and conducting capped edge.

The user is encouraged to experiment with this program as there are many parameters that can be altered to change the near and far field characteristic due to the scattering from a wedge structure.

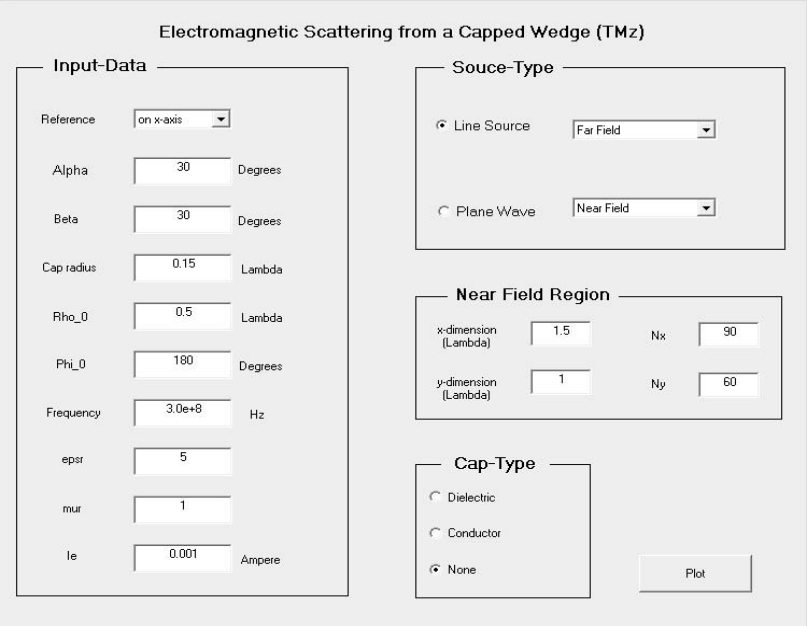

Figure 11.31. The parameters for computing the far field pattern of a 60 degrees wedge excited by a line source

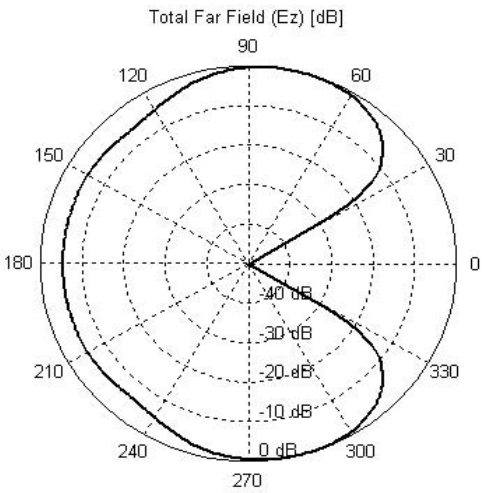

Figure 11.32. The far field pattern of a line source near a conducting wedge with sharp edge characterized by the parameters in Fig. 11.31.

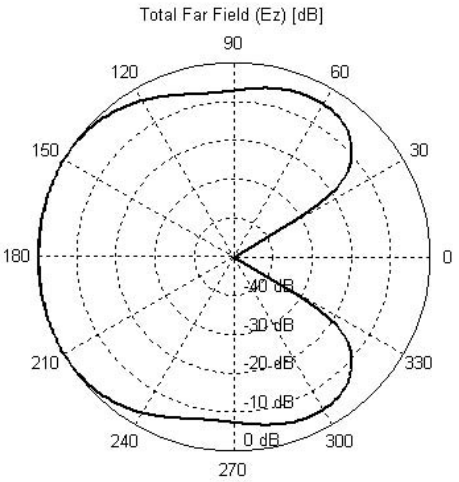

Figure 11.33. The far field pattern of a line source near a conducting wedge with a dielectric capped edge characterized by the parameters in Fig. 11.31.

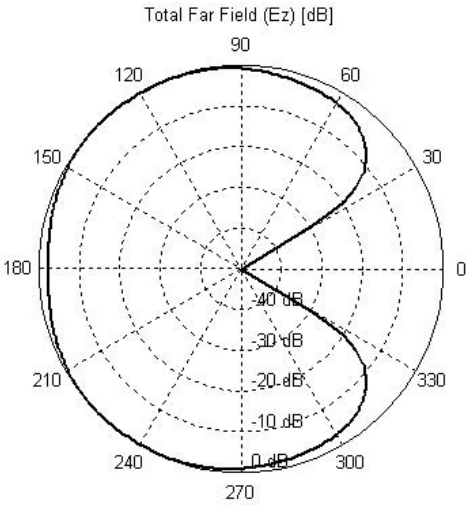

Figure 11.34. The far field pattern of a line source near a conducting wedge with a conducting capped edge characterized by the parameters in Fig. 11.31.

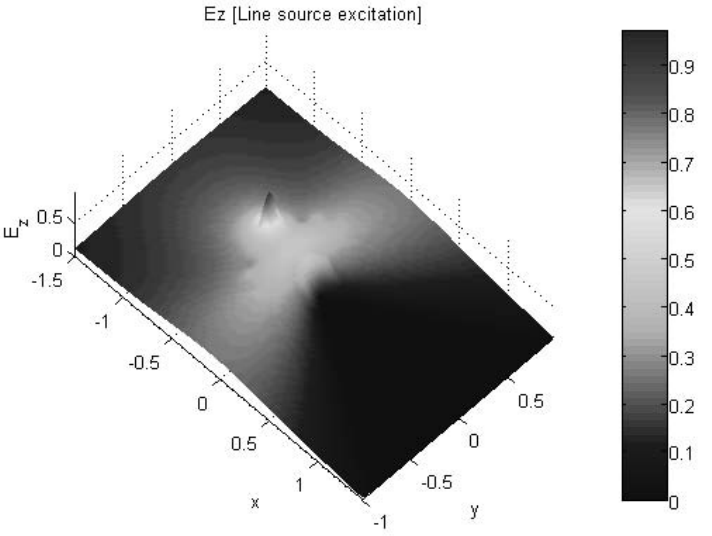

Figure 11.35. The E_z near field pattern of a line source near a conducting wedge with a sharp edge characterized by the parameters in Fig. 11.31.

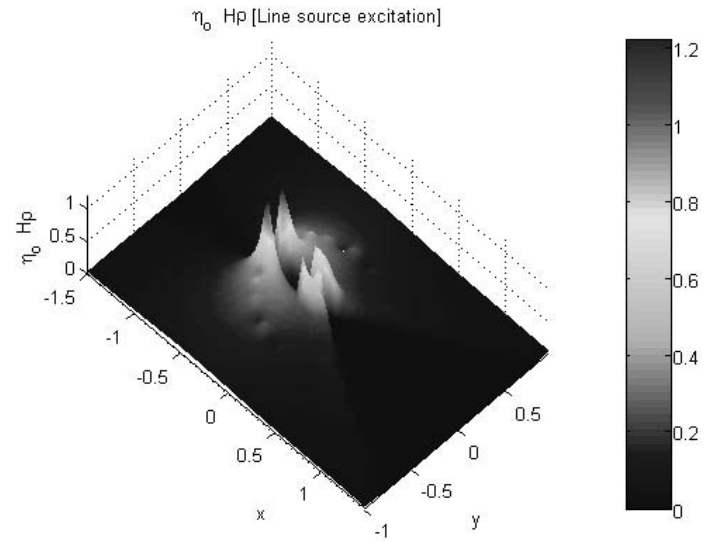

Figure 11.36. The H_ρ near field pattern of a line source near a conducting wedge with a sharp edge characterized by the parameters in Fig. 11.31.

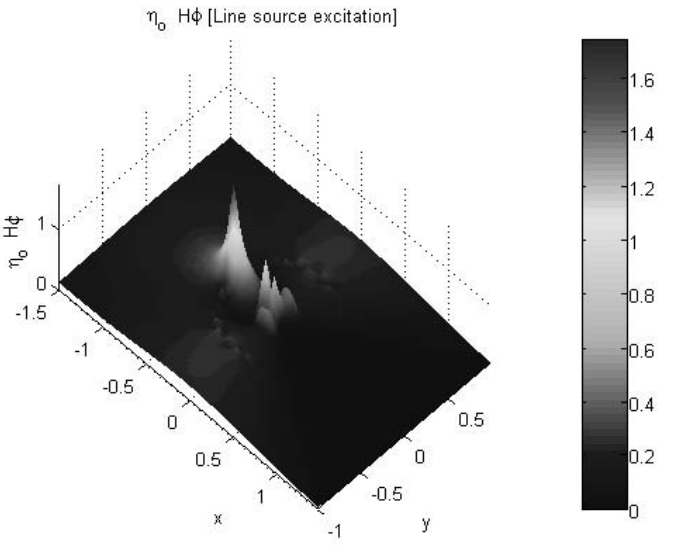

Figure 11.37. The H_φ near field pattern of a line source near a conducting wedge with a sharp edge characterized by the parameters in Fig. 11.31.

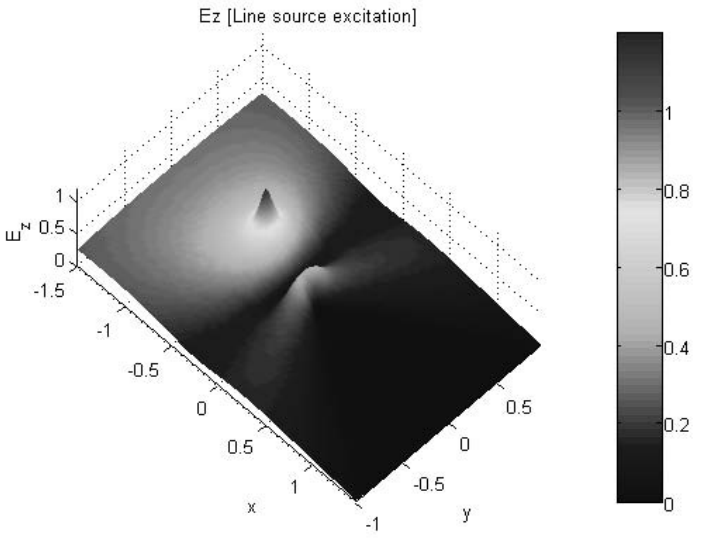

Figure 11.38. The E_z near field pattern of a line source near a conducting wedge with a dielectric cap edge characterized by Fig. 11.31.

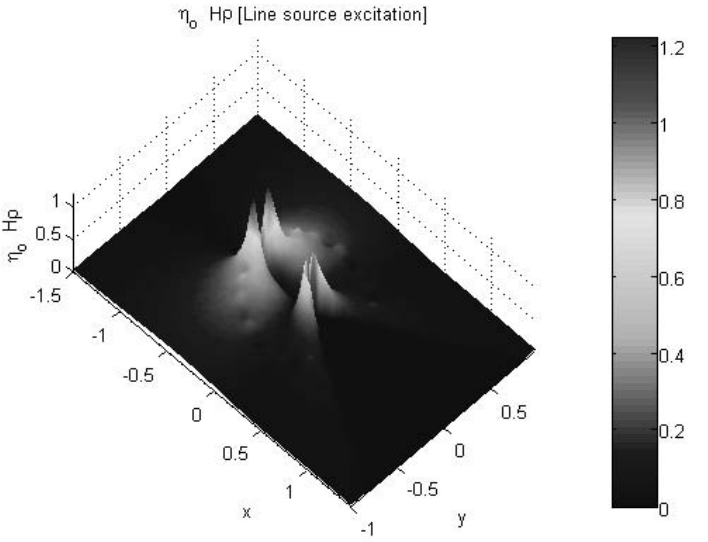

Figure 11.39. The H_ρ near field pattern of a line source near a conducting wedge with a dielectric cap edge characterized by Fig. 11.31.

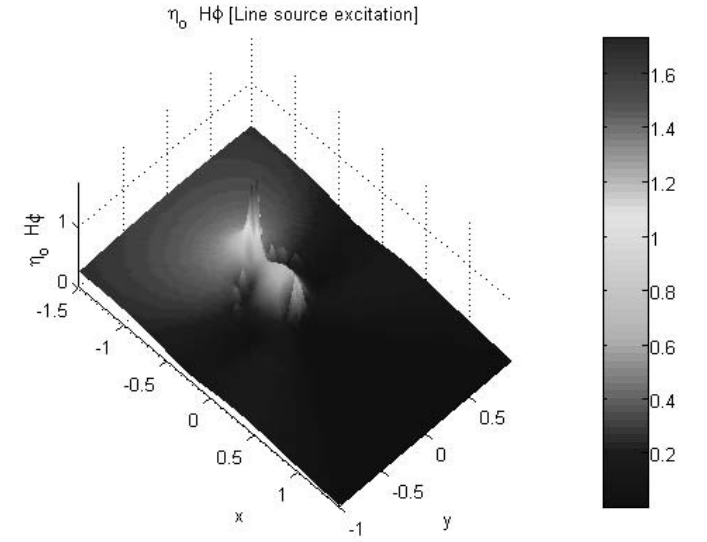

Figure 11.40. The H_ϕ near field pattern of a line source near a conducting wedge with a dielectric cap edge characterized by Fig. 11.31.

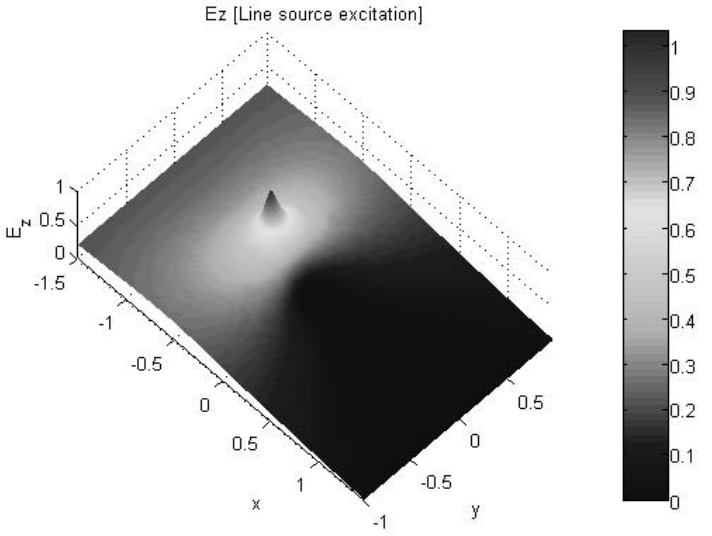

Figure 11.41. The E_z near field pattern of a line source near a conducting wedge with a conducting capped edge characterized by Fig. 11.31.

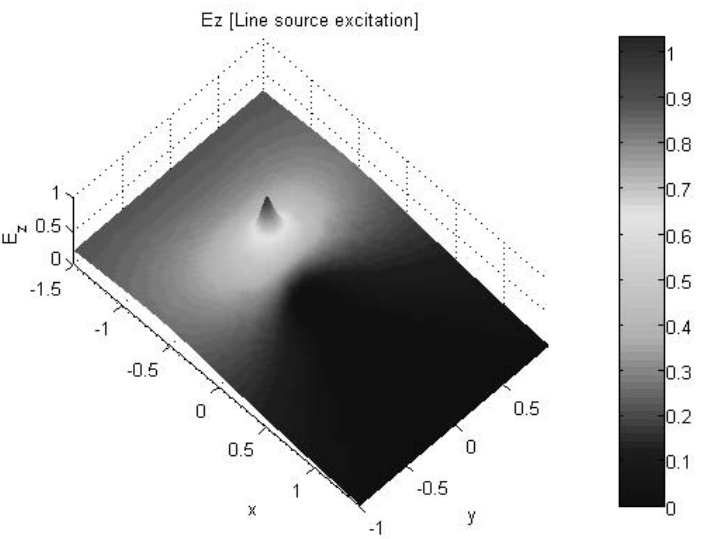

Figure 11.42. The H_ρ near field pattern of a line source near a conducting wedge with a conducting capped edge characterized by Fig. 11.31.

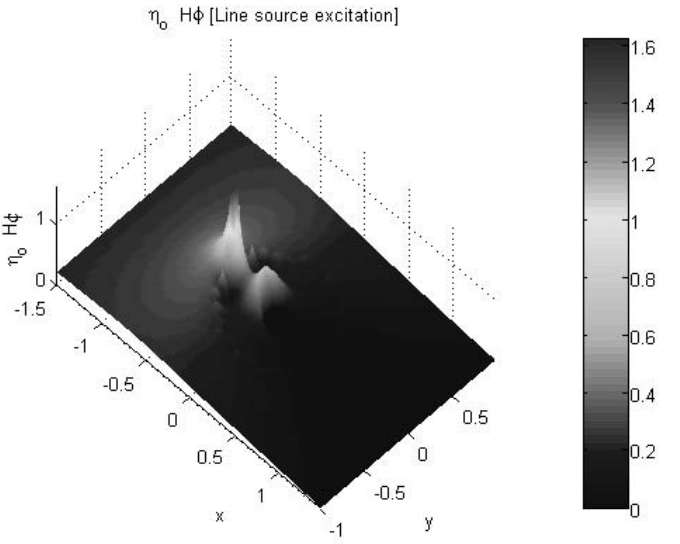

Figure 11.43. The H_φ near field pattern of a line source near a conducting wedge with a conducting capped edge characterized by Fig. 11.31.

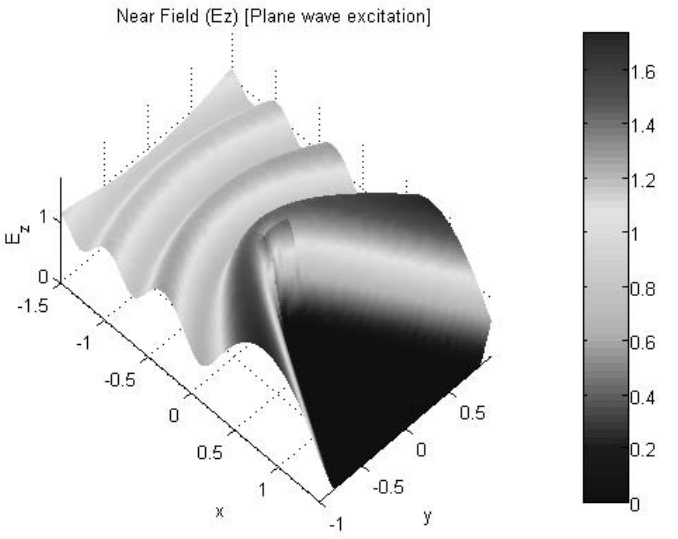

Figure 11.44. The E_z near field pattern of a plane wave incident on a conducting wedge with a sharp edge characterized by Fig. 11.31.

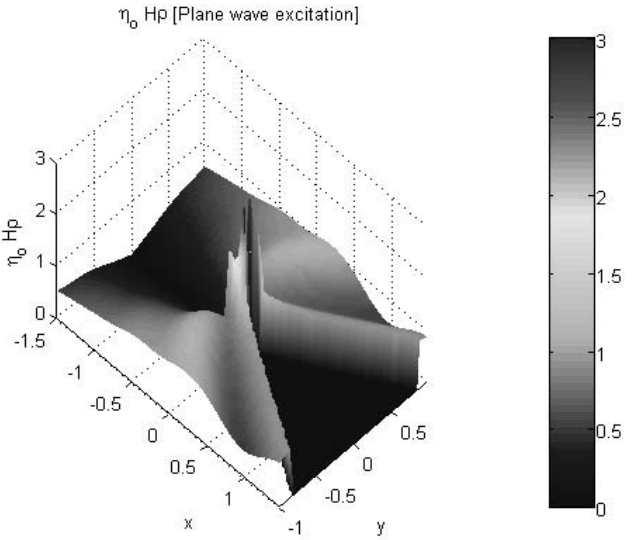

Figure 11.45. The H_ρ near field pattern of a plane wave incident on a conducting wedge with a sharp edge characterized by Fig. 11.31.

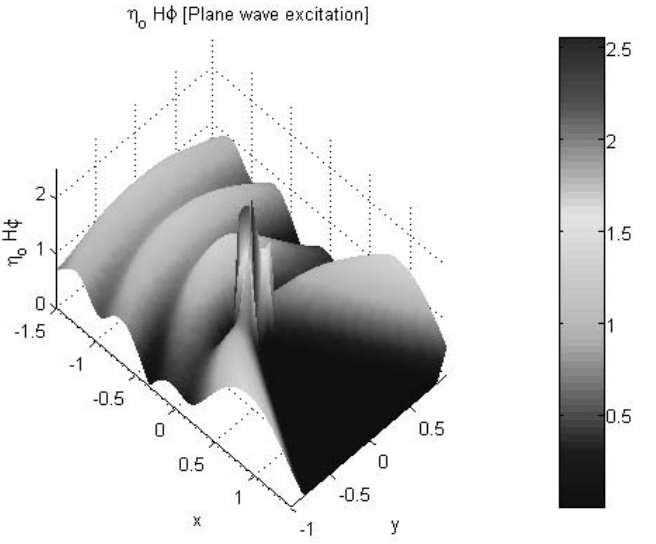

Figure 11.46. The H_φ near field pattern of a plane wave incident on a conducting wedge with a sharp edge characterized by Fig. 11.31.

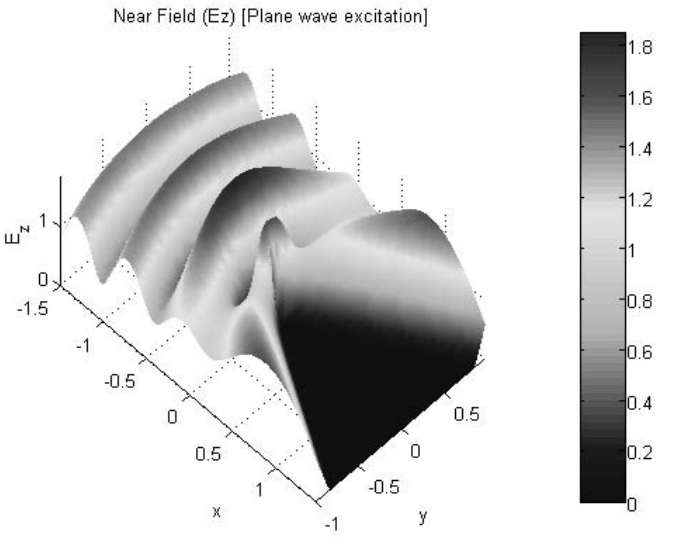

Figure 11.47. The E_z near field pattern of a plane wave incident on a conducting wedge with a dielectric edge characterized by Fig. 11.31.

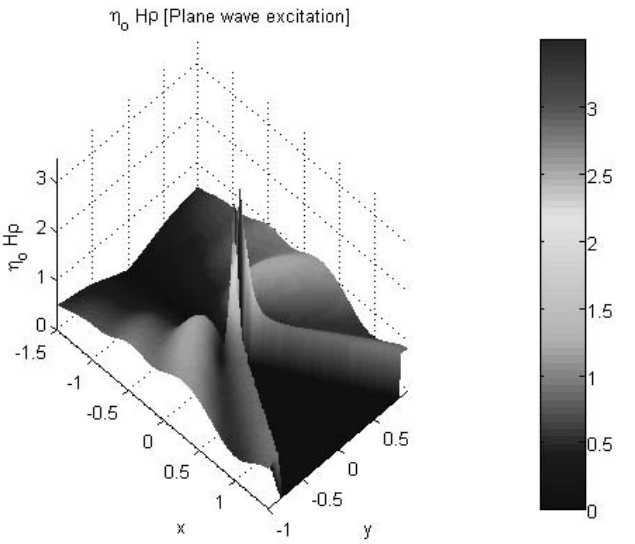

Figure 11.48. The H_ρ near field pattern of a plane wave incident on a conducting wedge with a dielectric edge characterized by Fig. 11.31.

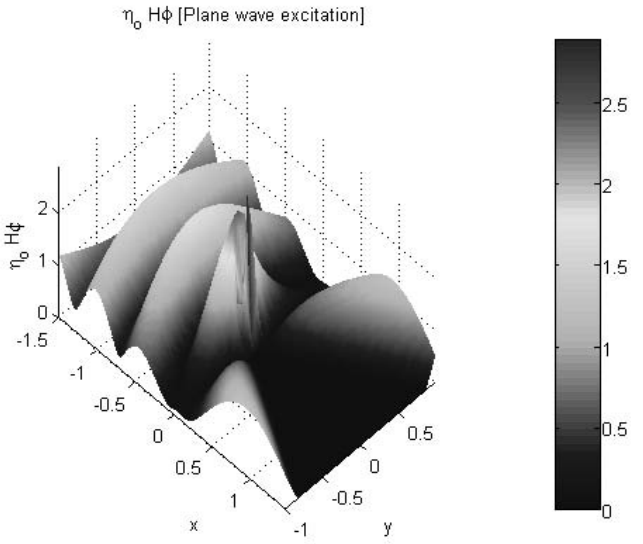

Figure 11.49. The H_φ near field pattern of a plane wave incident on a conducting wedge with dielectric capped edge characterized by Fig. 11.31.

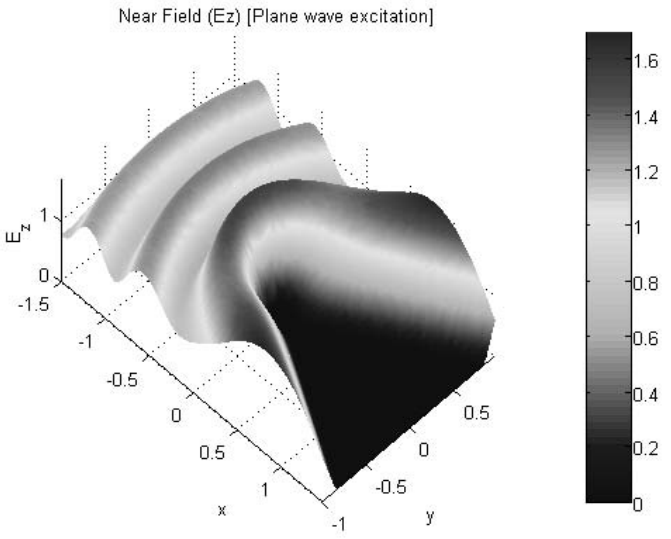

Figure 11.50. The E_z near field pattern of a plane wave incident on a conducting wedge with a conducting capped edge characterized by Fig. 11.31.

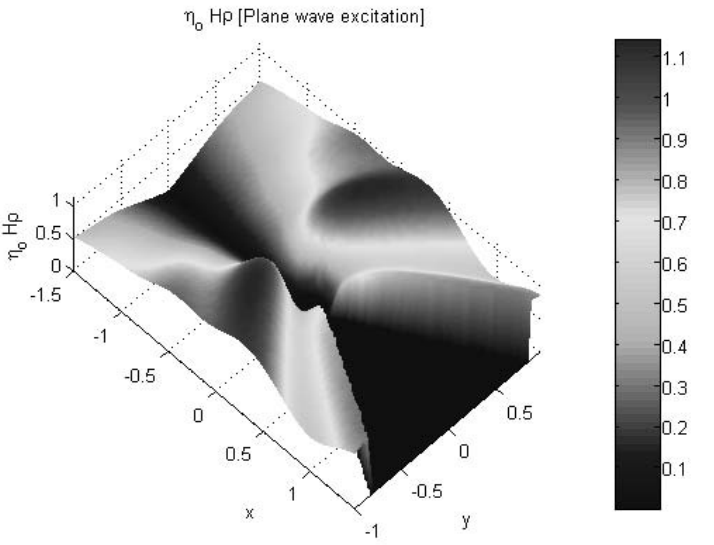

Figure 11.51. The H_ρ near field pattern of a plane wave incident on a conducting wedge with a conducting capped edge characterized by Fig. 11.31.

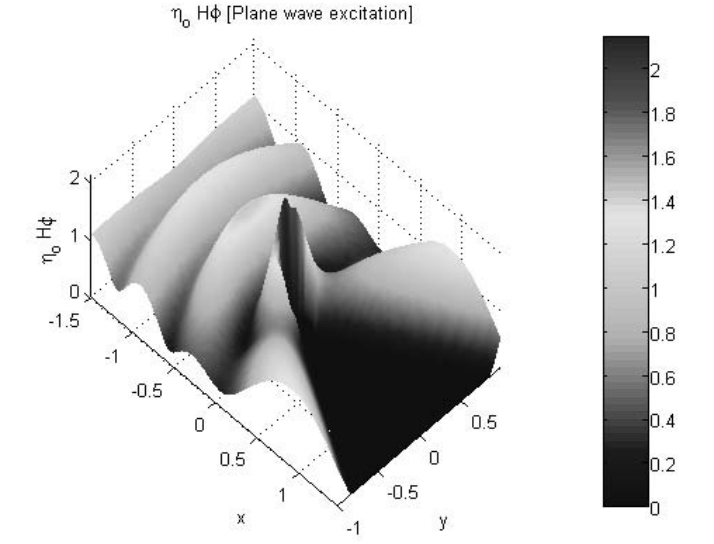

Figure 11.52. The H_φ near field pattern of a plane wave incident on a conducting wedge with a conducting capped edge characterized by Fig. 11.31.

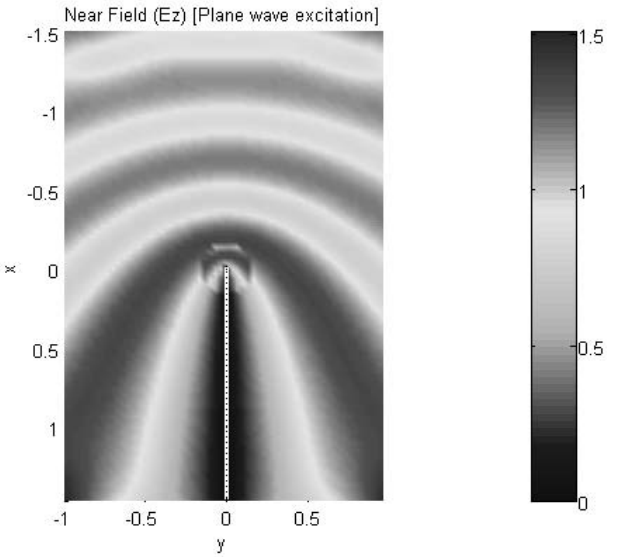

Figure 11.53. The E_z near field pattern of a plane wave incident on a half plane with sharp edge. All other parameters are as in Fig. 11.31.

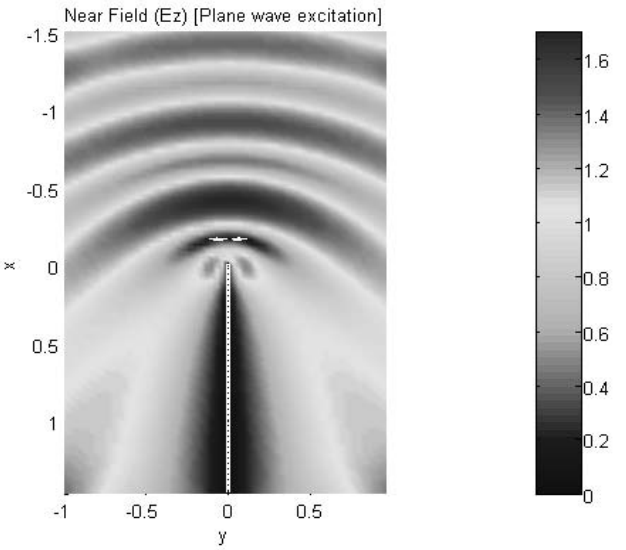

Figure 11.54. E_z near field pattern of a plane wave incident on a half plane with a dielectric capped edge. All other parameters are as in Fig. 11.31.

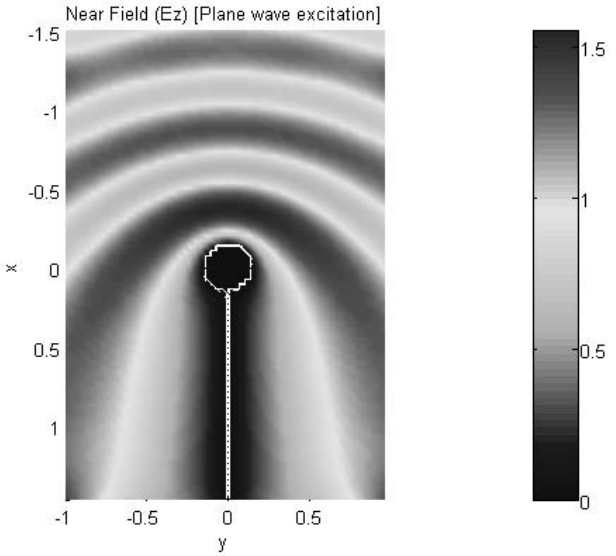

Figure 11.55. E_z **near field pattern of a plane wave incident on a half plane with a conducting capped edge. All other parameters are as in Fig. 11.31.**

11.7. 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 11.3, 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 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 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. 11.56. 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 reproduced this plot using the MATLAB program *"rcs_cyliner_complex.m"* given in Listing 11.19 in Section 11.9.

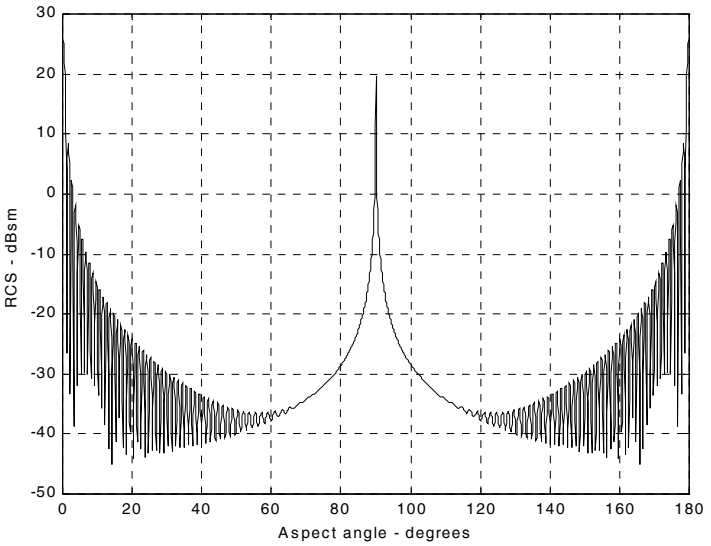

Figure 11.56. Backscattered RCS for a cylinder with flat plates.

11.8. RCS Fluctuations and Statistical Models

In most practical radar systems there is relative motion between the radar and an observed target. Therefore, the RCS measured by the radar fluctuates over a period of time as a function of frequency and the target aspect angle. This observed RCS is referred to as the radar dynamic cross section. Up to this point, all RCS formulas discussed in this chapter assumed a stationary target, where in this case, the backscattered RCS is often called static RCS.

Dynamic RCS may fluctuate in amplitude and/or in phase. Phase fluctuation is called glint, while amplitude fluctuation is called scintillation. Glint causes the far field backscattered wavefronts from a target to be non-planar. For most

radar applications, glint introduces linear errors in the radar measurements, and thus it is not of a major concern. However, in cases where high precision and accuracy are required, glint can be detrimental. Examples include precision instrumentation tracking radar systems, missile seekers, and automated aircraft landing systems. For more details on glint, the reader is advised to visit cited references listed in the bibliography.

Radar cross-section scintillation can vary slowly or rapidly depending on the target size, shape, dynamics, and its relative motion with respect to the radar. Thus, due to the wide variety of RCS scintillation sources, changes in the radar cross section are modeled statistically as random processes. The value of an RCS random process at any given time defines a random variable at that time. Many of the RCS scintillation models were developed and verified by experimental measurements.

11.8.1. RCS Statistical Models - Scintillation Models

This section presents the most commonly used RCS statistical models. Statistical models that apply to sea, land, and volume clutter, such as the Weibull and Log-normal distributions, will be discussed in a later chapter. The choice of a particular model depends heavily on the nature of the target under examination.

Chi-Square of Degree $2m$

The Chi-square distribution applies to a wide range of targets; its *pdf* is given by

$$f(\sigma) = \frac{m}{\Gamma(m)\sigma_{av}}\left(\frac{m\sigma}{\sigma_{av}}\right)^{m-1} e^{-m\sigma/\sigma_{av}} \qquad \sigma \geq 0 \tag{11.105}$$

where $\Gamma(m)$ is the gamma function with argument m, and σ_{av} is the average value. As the degree gets larger the distribution corresponds to constrained RCS values (narrow range of values). The limit $m \to \infty$ corresponds to a constant RCS target (steady-target case).

Swerling I and II (Chi-Square of Degree 2)

In Swerling I, the RCS samples measured by the radar are correlated throughout an entire scan, but are uncorrelated from scan to scan (slow fluctuation). In this case, the *pdf* is

$$f(\sigma) = \frac{1}{\sigma_{av}} \exp\left(-\frac{\sigma}{\sigma_{av}}\right) \qquad \sigma \geq 0 \tag{11.106}$$

where σ_{av} denotes the average RCS overall target fluctuation. Swerling II target fluctuation is more rapid than Swerling I, but the measurements are pulse to

pulse uncorrelated. Swerlings I and II apply to targets consisting of many independent fluctuating point scatterers of approximately equal physical dimensions.

Swerling III and IV (Chi-Square of Degree 4)

Swerlings III and IV have the same *pdf*, and it is given by

$$f(\sigma) = \frac{4\sigma}{\sigma_{av}^2}\exp\left(-\frac{2\sigma}{\sigma_{av}}\right) \qquad \sigma \geq 0 \tag{11.107}$$

The fluctuations in Swerling III are similar to Swerling I; while in Swerling IV they are similar to Swerling II fluctuations. Swerlings III and IV are more applicable to targets that can be represented by one dominant scatterer and many other small reflectors. Fig. 11.57 shows a typical plot of the *pdf*s for Swerling cases. This plot can be reproduced using MATLAB program *"Swerling_models.m"* given in Listing 11.20 in Section 11.9.

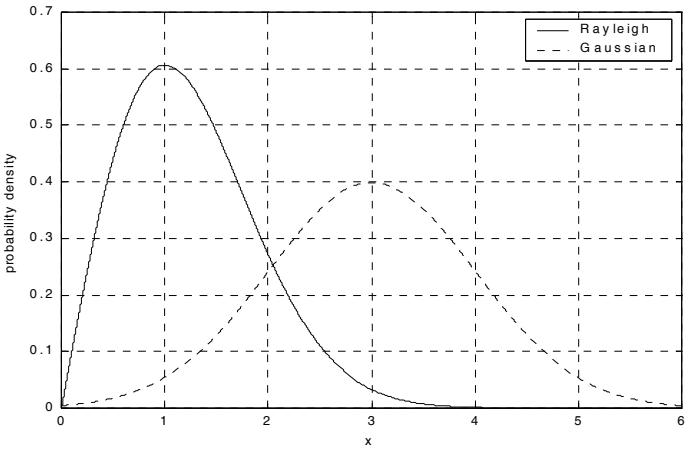

Figure 11.57. Probability densities for Swerling targets.

11.9. MATLAB Program and Function Listings

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

Listing 11.1. MATLAB Function "rcs_aspect.m"

```
function [rcs] = rcs_aspect (scat_spacing, freq)
% This function demonstrates the effect of aspect angle on RCS.
% Plot scatterers separated by scat_spacing meter. Initially the two scatterers
% are aligned with radar line of sight. The aspect angle is changed from
% 0 degrees to 180 degrees and the equivalent RCS is computed.
% Plot of RCS versus aspect is generated.
eps = 0.00001;
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 = (11.0 * scat_spacing / wavelength) .* cos(aspect_radians);
% Compute RCS (rcs = RCS_scat1 + RCS_scat2)
% Scat1 is taken as phase reference point
rcs = abs(1.0 + cos((11.0 * pi) .* elec_spacing) ...
          + i * sin((11.0 * pi) .* elec_spacing));
rcs = rcs + eps;
rcs = 20.0*log10(rcs); % RCS in dBsm
% Plot RCS versus aspect angle
figure (1);
plot (aspect_degrees,rcs,'k');
grid;
xlabel ('aspect angle - degrees');
ylabel ('RCS in dBsm');
%title (' Frequency is 3GHz; scatterer spacing is 0.5m');
```

Listing 11.2. MATLAB Function "rcs_frequency.m"

```
function [rcs] = rcs_frequency (scat_spacing, frequ, freql)
% This program demonstrates the dependency of RCS on wavelength
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
elec_spacing = 2.0 * scat_spacing ./ wavelength;
rcs = abs (  1 + cos((11.0 * pi) .* elec_spacing) ...
          + i * sin((11.0 * pi) .* elec_spacing));
```

```
rcs = rcs + eps;
rcs = 20.0*log10(rcs); % RCS ins dBsm
% Plot RCS versus frequency
freq = freql:delfreq:frequ;
plot(freq,rcs);
grid;
xlabel('Frequency');
ylabel('RCS in dBsm');
```

Listing 11.3. MATLAB Program "example11_1.m"

```
clear all
close all
N = 50;
wct = linspace(0,2*pi,N);
% Case 1
ax1 = cos(wct);
ay1 = sqrt(3) .* cos(wct);
M1 = moviein(N);
figure(1)
xc =0;
yc=0;
axis image
hold on
for ii = 1:N
  plot(ax1(ii),ay1(ii),'>r');
  line([xc ax1(ii)],[yc ay1(ii)]);
  plot(ax1,ay1,'g');
  M1(ii) = getframe;
end
grid
xlabel('Ex')
ylabel('Ey')
title('Electric Field Locus; case1')
% case 2
ax3 = cos(wct);
ay3 = sin(wct);
M3 = moviein(N);
figure(3)
axis image
hold on
for ii = 1:N
  plot(ax3(ii),ay3(ii),'>r');
  line([xc ax3(ii)],[yc ay3(ii)]);
```

```
  plot(ax3,ay3,'g');
  M3(ii) = getframe;
end
grid
xlabel('Ex')
ylabel('Ey')
title('Electric Field Locus; case 2')
rho = sqrt(ax3.^2 + ay3.^2);
major_axis = 2*max(rho);
minor_axis = 2*min(rho);
aspect3 = 10*log10(major_axis/minor_axis)
alpha3 = (180/pi) * atan2(ay3(1),ax3(1))
% Case 3
ax4 = cos(wct);
ay4 = cos(wct+(pi/6));
M4 = moviein(N);
figure(4)
axis image
hold on
for ii = 1:N
  plot(ax4(ii),ay4(ii),'>r');
  line([xc ax4(ii)],[yc ay4(ii)]);
  plot(ax4,ay4,'g')
  M4(ii) = getframe;
end
grid
xlabel('Ex')
ylabel('Ey')
title('Electric Field Locus; case 3')
rho = sqrt(ax4.^2 + ay4.^2);
major_axis = 2*max(rho);
minor_axis = 2*min(rho);
aspect4 = 10*log10(major_axis/minor_axis)
alpha4 = (180/pi) * atan2(ay4(1),ax4(1))
end
% Case 4
ax6 = cos(wct);
ay6 = sqrt(3) .* cos(wct+(pi/3));
M6 = moviein(N);
figure(6)
axis image

hold on
for ii = 1:N
```

```
   plot(ax6(ii),ay6(ii),'>r');
   line([xc ax6(ii)],[yc ay6(ii)]);
   plot(ax6,ay6,'g')
   M6(ii) = getframe;
end
grid
xlabel('Ex')
ylabel('Ey')
title('Electric Field Locus; case 4')
rho = sqrt(ax6.^2 + ay6.^2);
major_axis = 2*max(rho);
minor_axis = 2*min(rho);
aspect6 = 10*log10(major_axis/minor_axis)
alpha6 = (180/pi) * atan2(ay6(1),ax6(1))
```

Listing 11.4. MATLAB Program "rcs_sphere.m"

```
% This program calculates the back-scattered RCS for a perfectly
% conducting sphere using Eq.(11.7), and produces plots similar to Fig.2.9
% Spherical Bessel functions are computed using series approximation and
recursion.
clear all
eps   = 0.00001;
index = 0;
% kr limits are [0.05 - 15] ===> 300 points
for kr = 0.05:0.05:15
   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 = (11.*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:.05:15;
plot (n,rcs,'k');
set (gca,'xtick',[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]);
%xlabel ('Sphere circumference in wavelengths');
%ylabel ('Normalized sphere RCS');
grid;
figure (2);
plot (n,sphere_rcsdb,'k');
set (gca,'xtick',[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]);
xlabel ('Sphere circumference in wavelengths');
ylabel ('Normalized sphere RCS - dB');
grid;
figure (3);
semilogx (n,sphere_rcsdb,'k');
xlabel ('Sphere circumference in wavelengths');
ylabel ('Normalized sphere RCS - dB');
```

Listing 11.5. MATLAB Function "rcs_ellipsoid.m"

```
function [rcs] = rcs_ellipsoid (a, b, c, phi)
% This function computes and plots the ellipsoid RCS versus aspect angle.
% The roll angle phi is fixed,
eps = 0.00001;
sin_phi_s = sin(phi)^2;
cos_phi_s = cos(phi)^2;
% Generate aspect angle vector
theta = 0.:.05:180.0;
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);
figure (1);
plot ((theta * 180.0 / pi),rcs_db,'k');
xlabel ('Aspect angle - degrees');
ylabel ('RCS - dBsm');
%title ('phi = 45 deg, (a,b,c) = (.15,.20,.95) meter')
grid;
```

Listing 11.6. MATLAB Program "fig11_18a.m"

```
% Use this program to reproduce Fig. 11.18a
%This program computes the back-scattered RCS for an ellipsoid.
% The angle phi is fixed to three values 0, 45, and 90 degrees
% The angle theta is varied from 0-180 deg.
% A plot of RCS versus theta is generated
% Last modified on July 16, 2003
clear all;
% ===   Input parameters   ===
a = .15;          % 15 cm
b = .20;          % 20 cm
c = .95 ;         % 95 cm
% ===   End of Input parameters   ===
as = num2str(a);
bs = num2str(b);
cs = num2str(c);
eps = 0.00001;
dtr = pi/180;
for q = 1:3
   if q == 1
      phir = 0;       % the first value of the angle phi
   elseif q == 2
      phir = pi/4;    % the second value of the angle phi
   elseif q == 3
      phir = pi/2;    % the third value of the angle phi
   end
   sin_phi_s = sin(phir)^2;
   cos_phi_s = cos(phir)^2;
   % Generate aspect angle vector
   theta = 0.:.05:180;
   thetar = theta * dtr;
   if(a ~= b & a ~= c)
```

```
      rcs(q,:) = (pi * a^2 * b^2 * c^2) ./ (a^2 * cos_phi_s .* (sin(thetar).^2) + ...
         b^2 * sin_phi_s .* (sin(thetar).^2) + ...
         c^2 .* (cos(thetar).^2)).^2 ;
   elseif(a == b & a ~= c)
      rcs(q,:) = (pi * b^4 * c^2) ./ ( b^2 .* (sin(thetar).^2) + ...
         c^2 .* (cos(thetar).^2)).^2 ;
   elseif (a == b & a ==c)
      rcs(q,:) = pi * c^2;
   end
end
rcs_db = 10.0 * log10(rcs);
figure (1);
plot(theta,rcs_db(1,:),'b',theta,rcs_db(2,:),'r:',theta,rcs_db(3,:),'g--','line-
width',1.5);
xlabel ('Aspect angle, Theta [Degrees]');
ylabel ('RCS - dBsm');
title (['Ellipsoid with (a,b,c) = (', [as],', ', [bs],', ', [cs], ')  meter'])
legend ('phi = 0^o','phi = 45^o','phi = 90^o')
grid;
```

Listing 11.7. MATLAB Function "rcs_circ_plate.m"

```
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 ('Aspect angle - degrees');
ylabel ('RCS - dBsm');
legend('Using Eq.(11.37)','Using Eq.(11.36)')
freqGH = num2str(freq*1.e-9);
title (['Frequency = ',[freqGH],'  GHz']);
```

Listing 11.8. MATLAB Function "rcs_frustum.m"

```
function [rcs] = rcs_frustum (r1, r2, h, freq, indicator)
% This program computes the monostatic RCS for a frustum.
% Incident linear Polarization is assumed.
% To compute RCP or LCP RCS one must use Eq. (11.24)
% When viewing from the small end of the frustum
% normal incidence occurs at aspect pi/2 - half cone angle
% When viewing from the large end, normal incidence occurs at
% pi/2 + half cone angle.
% RCS is computed using Eq. (11.43). This program assumes a geometry
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_incedence = (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));
grid;
xlabel ('Aspect angle - degrees');
ylabel ('RCS - dBsm');
if(indicator ==1)
   title ('Viewing from large end');
else
   title ('Viewing from small end');
end
```

Listing 11.9. MATLAB Function "rcs_cylinder.m"

```
function [rcs] = rcs_cylinder(r1, r2, h, freq, phi, CylinderType)
% rcs_cylinder.m
% This program computes monostatic RCS for a finite length
% cylinder of either curricular or elliptical cross-section.
% Plot of RCS versus aspect angle theta is generated at a specified
% input angle phi
% Last modified on July 16, 2003
 r = r1;          % radius of the circular cylinder
eps =0.00001;
dtr = pi/180;
phir = phi*dtr;
freqGH = num2str(freq*1.e-9);
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
% Plot the results
delta= 180/(index-1);
angle = 0:delta:180;
plot(angle,10*log10(rcs),'k','linewidth',1.5);
grid;
xlabel ('Aspect angle, Theta [Degrees]');;
ylabel ('RCS - dBsm');
title  ([[CylinderType],'  Cylinder','  at Frequency = ',[freqGH],'  GHz']);
```

Listing 11.10. MATLAB Function "rcs_rect_plate.m"

```
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(11.50)through (11.60). Also Physical
% Optics approximation Eq.(11.62) 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.(11.62)
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(2)
plot (theta_deg, rcsdb_v,'k',theta_deg,rcsdb_po,'k -.');
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 ('RCS -dBsm');
xlabel ('Aspect angle - deg');
legend('Eq.(11.50)','Eq.(11.62)')
figure(3)
plot (theta_deg, rcsdb_h,'k',theta_deg,rcsdb_po,'k -.');
set(gca,'xtick',[10:10:85]);
title (['Horizontal Polarization,  ','Frequency = ',[freqGH],'  GHz, ', '  a = ',
[A], ' m','  b = ',[B],' m']);
ylabel ('RCS -dBsm');
xlabel ('Aspect angle - deg');
legend('Eq.(11.51)','Eq.(11.62)')
```

Listing 11.11. MATLAB Function "rcs_isosceles.m"

```
function [rcs] = rcs_isosceles (a, b, freq, phi)
% This program calculates the backscattered RCS for a perfectly
% conducting triangular flat plate, using Eqs. (11.63) through (11.65)
% The default case is to assume phi = pi/2. These equations are
% valid for aspect angles less than 30 degrees
% compute area of plate
A = a * b / 2.;
lambda = 3.e+8 / freq;
phi = pi / 2.;
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 .* ((11. * a / b) * cos(phi) .* ...
    sin(beta) - sin(phi) .* sin(11. .* 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')
xlabel ('Aspect angle - degrees');
ylabel ('RCS - dBsm')
%title ('freq = 9.5GHz, phi = pi/2');
grid;
```

Listing 11.12. MATLAB Program "Capped_WedgeTM.m"

```
   % Program to calculate the near field of a sharp conducting wedge
% due to an incident field from a line source or a plane wave
% By: Dr. Atef Elsherbeni -- atef@olemiss.edu
% This program uses 6 other functions
% Last modified July 24, 2003
clear all
close all
img = sqrt(-1);
rtd = 180/pi;   dtr = pi/180;
mu0 = 4*pi*1e-7;              % Permeability of free space
eps0 = 8.854e-12;             % Permittivity of free space
%  =====   Input parameters   =====
alphad = 30;                  % above x Wedge angle
betad = 30;                   % Below x wedge angle
reference = 'on x-axis';        % Reference condition 'top face' or 'bisector' or
'on x-axis'
CapType = 'Diel';             % Cap Type 'Cond', 'diel' or 'None'
ar = .15;                     % Cap radius in lambda
rhop = 0.5;                   % radial Position of the line source in terms of lambda
phipd = 180;                  % angular position of the line source
Ie = .001;                    % Amplitude of the current source
freq = 2.998e8;               % frequency
mur = 1;
epsr = 1;
ax = 1.5;    by = 1;          % area for near field calculations
nx = 30;       ny = 20;       % Number of points for near field calculations
%  =====  End of Input Data   =====
```

```
alpha = alphad*dtr;
beta = betad *dtr;

switch reference
   case 'top face'
      alpha = 0;
      vi = pi/(2*pi-beta);
   case 'bisector'
      beta = alpha;
      vi = pi/(2*pi-2*beta);
   case 'on x-axis'
      vi = pi/(2*pi-alpha-beta);
end
phip = phipd*dtr;
etar = sqrt(mur/epsr);
mu = mu0*mur;
eps = eps0*epsr;
lambda = 2.99e8/freq;
k = 2*pi/lambda;              % free space wavenumber
ka = k*ar;
k1 = k*sqrt(mur*epsr);        % wavenumber inside dielectric
k1a = k1*ar;
krhop = k*rhop;
omega =2*pi*freq;
%  <<< Far field Calculations of Ez component >>>
%  ===  Line source excitation  ===
Nc =round(1+2*k*rhop);        % number of terms for series summation
Term  = pi*omega*mu0/(2*pi-alpha-beta);
Term0D =  img*4*pi/(2*pi-alpha-beta);
Term0C = -img*4*pi/(2*pi-alpha-beta);
Term0  =     4*pi/(2*pi-alpha-beta);
for ip = 1:360
   phii = (ip -1)*dtr;
   xphi(ip) = ip-1;
   if phii > alpha  & phii < 2*pi-beta %  outside the wedge region
      EzFLs(ip) = 0;
      for m = 1:Nc
         v = m*vi;
         ssterm = (img^v)*sin(v*(phip-alpha))*sin(v*(phii-alpha));
         switch CapType
            case 'Diel'
               Aterm = k * besselj(v,k1a)*(dbesselj(v,ka)*bessely(v,krhop)...
                     -dbessely(v,ka)*besselj(v,krhop)) ...
                     +k1*dbesselj(v,k1a)*( bessely(v,ka)*besselj(v,krhop)...
```

```
                -besselj(v,ka)*bessely(v,krhop));
            Bterm =k*dbesselh(v,2,ka)*besselj(v,k1a) ...
                -k1*besselh(v,2,ka)*dbesselj(v,k1a);
            EzLS(m) = Term0D*ssterm*Aterm/Bterm;
         case 'Cond'
            Aterm = bessely(v,ka)*besselj(v,krhop) ...
                - besselj(v,ka)*bessely(v,krhop);
            Bterm = besselh(v,2,ka);
            EzLS(m) = Term0C*ssterm*Aterm/Bterm;
         case 'None'
            EzLS(m) = Term0*ssterm*besselj(v,krhop);
       end
    end
    EzFLs(ip) = abs(sum(EzLS));
  else
    EzFLs(ip)=0;
  end
end
EzFLs = EzFLs/max(EzFLs);

figure(1);
plot(xphi,EzFLs,'linewidth',1.5);
xlabel('Observation angle \phi^o');
ylabel('Ez');
axis ([0 360 0 1])
title('Total Far Field (Ez) [Line source excitation]');

figure(2)
polardb(xphi*dtr,EzFLs,'k')
title ('Total Far Field (Ez) [dB]')

%  <<<  Near field observation points  >>>
delx = 2*ax/nx; dely = 2*by/ny;
xi = -ax;  yi = -by;          % Initial values for x and y
for i = 1:nx
  for j = 1:ny
    x(i,j) = xi + (i-1)*delx;
    y(i,j) = yi + (j-1) *dely;
    rho(i,j) = sqrt(x(i,j)^2+y(i,j)^2);
    phi(i,j) = atan2(y(i,j),x(i,j));
    if phi(i,j) < 0
      phi(i,j) = phi(i,j) + 2*pi;
    end
    if rho(i,j) <= 0.001
```

```
            rho(i,j) = 0.001;
        end
    end
end

%   Line source excitation, near field calculations

%   ====   Line source coefficients    ====
Nc =round(1+2*k*max(max(rho)));          % number of terms for series sum-
mation
Term   = Ie*pi*omega*mu0/(2*pi-alpha-beta);
for m = 1:Nc
    v = m*vi;
    switch CapType
        case 'Diel'
            b(m) = -Term * besselh(v,2,krhop);
            c(m) = -b(m) * (k*dbesselj(v,ka)*besselj(v,k1a) ...
                  -k1*besselj(v,ka)*dbesselj(v,k1a)) ...
                  /(k*dbesselh(v,2,ka)*besselj(v,k1a) ...
                  -k1*besselh(v,2,ka)*dbesselj(v,k1a));
            d(m) = c(m) + b(m) * besselj(v,krhop) ...
                / besselh(v,2,krhop);
            a(m) = ( b(m) * besselj(v,ka)+c(m) ...
                * besselh(v,2,ka))/besselj(v,k1a);
        case 'Cond'
            b(m) = -Term * besselh(v,2,krhop);
            c(m) = -b(m) * besselj(v,ka)/besselh(v,2,ka);
            d(m) = c(m) + b(m) * besselj(v,krhop) ...
                / besselh(v,2,krhop);
            a(m) = 0;
        case 'None'
            b(m) = -Term * besselh(v,2,krhop);
            c(m) = 0;
            d(m) = -Term * besselj(v,krhop);
            a(m) = b(m);
    end
end
   termhphi = sqrt(-1)*omega*mu0;
termhrho = -termhphi;
for i = 1:nx
    for j = 1:ny
        for m = 1:Nc
            v = m*vi;  % Equation
```

```
            [Ezt,Hphit,Hrhot] =
DielCappedWedgeTMFields_Ls(v,m,rho(i,j),phi(i,j),rhop, ...
                        phip,ar,k,k1,alpha,beta,a,b,c,d);
            Eztt(m) = Ezt;
            Hphitt(m) = Hphit;
            Hrhott(m) = Hrhot;
        end
        SEz(i,j) = sum(Eztt);
        SHphi(i,j) = sum(Hphitt)/termhphi;
        SHrho(i,j) = sum(Hrhott)/termhrho;
    end
end
figure(3);
surf(x,y,abs(SEz));
axis ('equal');
view(45,60);
shading interp;
xlabel('x');
ylabel('y');
zlabel('E_z');
title('Ez [Line source excitation]');
colorbar; colormap(copper);  % colormap(jet);
figure(4);
surf(x,y,377*abs(SHrho));
axis ('equal');
view(45,60);
shading interp;
xlabel('x');
ylabel('y');
zlabel('\eta_o  H\rho');
title('\eta_o  H\rho [Line source excitation]');
colorbar; colormap(copper);  % colormap(jet);
figure(5);
surf(x,y,377*abs(SHphi));
axis ('equal');
view(45,60);
shading interp;
xlabel('x');
ylabel('y');
zlabel('\eta_o  H\phi');
title('\eta_o  H\phi [Line source excitation]')
colorbar; colormap(copper);  % colormap(jet);
%   ===   Plane wave excitation, near field calculations   ===
```

```
Nc =round(1+2*k*max(max(rho)));          % number of terms for series summation
Term   = 4*pi/(2*pi-alpha-beta);
for m = 1:Nc
   v = m*vi;
   switch CapType
      case 'Diel'
         b(m) = Term * img^v;
         c(m) = -b(m) * (k*dbesselj(v,ka)*besselj(v,k1a)...
               -k1*besselj(v,ka)*dbesselj(v,k1a)) ...
              / (k*dbesselh(v,2,ka)*besselj(v,k1a) ...
               -k1*besselh(v,2,ka)*dbesselj(v,k1a));
         a(m) = ( b(m) * besselj(v,ka)+c(m) * besselh(v,2,ka))/besselj(v,k1a);
      case 'Cond'
         b(m) = -Term * img^v;
         c(m) = -b(m) * besselj(v,ka)/besselh(v,2,ka);
         a(m) = 0;
      case 'None'
         b(m) = -Term * img^v;
         c(m) = 0;
         a(m) = b(m);
   end
end
termhphi = sqrt(-1)*omega*mu0;
termhrho = -termhphi;
for i = 1:nx
   for j = 1:ny
      for m = 1:Nc
         v = m*vi;  % Equation
          [Ezt,Hphit,Hrhot] =
DielCappedWedgeTMFields_PW(v,m,rho(i,j),phi(i,j), ...
                        php,ar,k,k1,alpha,beta,a,b,c);
         Eztt(m) = Ezt;
         Hphitt(m) = Hphit;
         Hrhott(m) = Hrhot;
end
      EzPW(i,j) = sum(Eztt);
      HphiPW(i,j) = sum(Hphitt)/termhphi;
      HrhoPW(i,j) = sum(Hrhott)/termhrho;
  end
end
figure(6);
surf(x,y,abs(EzPW));
axis ('equal');
```

```
view(45,60);
shading interp;
xlabel('x');
ylabel('y');
zlabel('E_z');
colorbar; colormap(copper);  % colormap(jet);
title('Near Field (Ez) [Plane wave excitation]');
figure(7);
surf(x,y,377*abs(HrhoPW));
axis ('equal');
view(45,60);
shading interp;
xlabel('x');
ylabel('y');
zlabel('\eta_o H\rho');
title('\eta_o H\rho [Plane wave excitation]');
colorbar; colormap(copper);  % colormap(jet);
figure(8);
surf(x,y,377*abs(HphiPW));
axis ('equal');
view(45,60);
shading interp;
xlabel('x');
ylabel('y');
zlabel('\eta_o H\phi');
title('\eta_o H\phi [Plane wave excitation]');
colorbar; colormap(copper);  % colormap(jet);
```

Listing 11.13. MATLAB Function "DielCappedWedgeTMFields_Ls.m"

```
function [Ezt,Hphit,Hrhot] =
DielCappedWedgeTMFields_Ls(v,m,rhoij,phiij,rhop,php,ar,k,k1,alpha,beta,a,
b,c,d);
% Function to calculate the near field components of a capped wedge
% with a line source excitation at one near field point
% This function is to be called by the Main program:
Diel_Capped_WedgeTM.m
% By: Dr. Atef Elsherbeni -- atef@olemiss.edu
% Last modified July 23, 2003
Ezt = 0;  Hrhot = 0;  Hphit = 0;    % Initialization
if phiij > alpha  & phiij < 2*pi-beta %  outside the wedge region
   krho = k*rhoij;
   k1rho = k1*rhoij;
```

```
   jvkrho = besselj(v,krho);
   hvkrho = besselh(v,2,krho);
   jvk1rho = besselj(v,k1rho);
   djvkrho = dbesselj(v,krho);
   djvk1rho = dbesselj(v,k1rho);
   dhvkrho = dbesselh(v,2,krho);
   ssterm = sin(v*(phip-alpha))*sin(v*(phiij-alpha));
   scterm = sin(v*(phip-alpha))*cos(v*(phiij-alpha));
     if rhoij <= ar   % field point location is inside the cap region
      Ezt = a(m)*jvk1rho*ssterm;
      Hphit = k1*a(m)*djvk1rho*ssterm;
      Hrhot = v*a(m)*jvk1rho*scterm/rhoij;
   elseif rhoij <= rhop   % field point location is between cap and the line
source location
      Ezt = (b(m)*jvkrho+c(m)*hvkrho)*ssterm;
      Hphit = k*(b(m)*djvkrho+c(m)*dhvkrho)*ssterm;
      Hrhot = v*(b(m)*jvkrho+c(m)*hvkrho)*scterm/rhoij;
   elseif rhoij > rhop % field point location is greater than the line source loca-
tion
      Ezt = d(m)*hvkrho*ssterm;
      Hphit = k*d(m)*dhvkrho*ssterm;
      Hrhot = v*d(m)*hvkrho*scterm/rhoij;
   end
else
   Ezt = 0;    Hrhot = 0;  Hphit = 0;  % inside wedge region
End
```

Listing 11.14. MATLAB Function "DielCappedWedgeTMFields_PW.m"

```
function [Ezt,Hphit,Hrhot] =
DielCappedWedgeTMFields_PW(v,m,rhoij,phiij,phip,ar,k,k1,alpha,beta,a,b,c)
;
% Function to calculate the near field components of a capped wedge
% with a line source excitation at one near field point
% This function is to be called by the Main program:
Diel_Capped_WedgeTM.m
% By: Dr. Atef Elsherbeni -- atef@olemiss.edu
% Last modified July 23, 2003
Ezt = 0;  Hrhot = 0;  Hphit = 0;    % Initialization
if phiij > alpha  & phiij < 2*pi-beta %  outside the wedge region
   krho = k*rhoij;
   k1rho = k1*rhoij;
   jvkrho = besselj(v,krho);
```

```
   hvkrho = besselh(v,2,krho);
   jvk1rho = besselj(v,k1rho);
   djvkrho = dbesselj(v,krho);
   djvk1rho = dbesselj(v,k1rho);
   dhvkrho = dbesselh(v,2,krho);
   ssterm = sin(v*(phip-alpha))*sin(v*(phiij-alpha));
   scterm = sin(v*(phip-alpha))*cos(v*(phiij-alpha));
   if rhoij <= ar   % field point location is inside the cap region
      Ezt = a(m)*jvk1rho*ssterm;
      Hphit = k1*a(m)*djvk1rho*ssterm;
      Hrhot = v*a(m)*jvk1rho*scterm/rhoij;
   else   % field point location is between the cap and the line source location
      Ezt = (b(m)*jvkrho+c(m)*hvkrho)*ssterm;
      Hphit = k*(b(m)*djvkrho+c(m)*dhvkrho)*ssterm;
      Hrhot = v*(b(m)*jvkrho+c(m)*hvkrho)*scterm/rhoij;
   end
else
   Ezt = 0;    Hrhot = 0;  Hphit = 0;  % inside wedge region
End
```

Listing 11.15. MATLAB Function "polardb.m"

```
function polardb(theta,rho,line_style)
%   POLARDB  Polar coordinate plot.
%   POLARDB(THETA, RHO) makes a plot using polar coordinates of
%   the angle THETA, in radians, versus the radius RHO in dB.
%   The maximum value of RHO should not exceed 1. It should not be
%   normalized, however (i.e., its max. value may be less than 1).
%   POLAR(THETA,RHO,S) uses the linestyle specified in string S.
%   See PLOT for a description of legal linestyles.
if nargin < 1
   error('Requires 2 or 3 input arguments.')
elseif nargin == 2
   if isstr(rho)
      line_style = rho;
      rho = theta;
      [mr,nr] = size(rho);
      if mr == 1
         theta = 1:nr;
      else
         th = (1:mr)';
         theta = th(:,ones(1,nr));
      end
   else
```

```
        line_style = 'auto';
    end
elseif nargin == 1
    line_style = 'auto';
    rho = theta;
    [mr,nr] = size(rho);
    if mr == 1
        theta = 1:nr;
    else
        th = (1:mr)';
        theta = th(:,ones(1,nr));
    end
end
if isstr(theta) | isstr(rho)
    error('Input arguments must be numeric.');
end
if ~isequal(size(theta),size(rho))
    error('THETA and RHO must be the same size.');
end
% get hold state
cax = newplot;
next = lower(get(cax,'NextPlot'));
hold_state = ishold;
% get x-axis text color so grid is in same color
tc = get(cax,'xcolor');
ls = get(cax,'gridlinestyle');
% Hold on to current Text defaults, reset them to the
% Axes' font attributes so tick marks use them.
fAngle  = get(cax, 'DefaultTextFontAngle');
fName   = get(cax, 'DefaultTextFontName');
fSize   = get(cax, 'DefaultTextFontSize');
fWeight = get(cax, 'DefaultTextFontWeight');
fUnits  = get(cax, 'DefaultTextUnits');
set(cax, 'DefaultTextFontAngle',  get(cax, 'FontAngle'), ...
    'DefaultTextFontName',   get(cax, 'FontName'), ...
    'DefaultTextFontSize',   get(cax, 'FontSize'), ...
    'DefaultTextFontWeight', get(cax, 'FontWeight'), ...
    'DefaultTextUnits','data')
% make a radial grid
    hold on;
    maxrho =1;
    hhh=plot([-maxrho -maxrho maxrho maxrho],[-maxrho maxrho maxrho -
maxrho]);
    set(gca,'dataaspectratio',[1 1 1],'plotboxaspectratiomode','auto')
```

```
    v = [get(cax,'xlim') get(cax,'ylim')];
    ticks = sum(get(cax,'ytick')>=0);
    delete(hhh);
  % check radial limits and ticks
    rmin = 0; rmax = v(4); rticks = max(ticks-1,2);
    if rticks > 5   % see if we can reduce the number
       if rem(rticks,2) == 0
          rticks = rticks/2;
       elseif rem(rticks,3) == 0
          rticks = rticks/3;
       end
    end
  % only do grids if hold is off
  if ~hold_state
  % define a circle
    th = 0:pi/50:2*pi;
    xunit = cos(th);
    yunit = sin(th);
  % now really force points on x/y axes to lie on them exactly
    inds = 1:(length(th)-1)/4:length(th);
    xunit(inds(2:2:4)) = zeros(2,1);
    yunit(inds(1:2:5)) = zeros(3,1);
  % plot background if necessary
    if ~isstr(get(cax,'color')),
      patch('xdata',xunit*rmax,'ydata',yunit*rmax, ...
            'edgecolor',tc,'facecolor',get(gca,'color'),...
            'handlevisibility','off');
    end
  % draw radial circles with dB ticks
    c82 = cos(82*pi/180);
    s82 = sin(82*pi/180);
    rinc = (rmax-rmin)/rticks;
    tickdB=-10*(rticks-1);    % the innermost tick dB value
    for i=(rmin+rinc):rinc:rmax
       hhh = plot(xunit*i,yunit*i,ls,'color',tc,'linewidth',1,...
                  'handlevisibility','off');
       text((i+rinc/20)*c82*0,-(i+rinc/20)*s82, ...
          ['  ' num2str(tickdB) ' dB'],'verticalalignment','bottom',...
          'handlevisibility','off')
       tickdB=tickdB+10;
    end
    set(hhh,'linestyle','-') % Make outer circle solid
  % plot spokes
    th = (1:6)*2*pi/12;
```

```
    cst = cos(th); snt = sin(th);
    cs = [-cst; cst];
    sn = [-snt; snt];
    plot(rmax*cs,rmax*sn,ls,'color',tc,'linewidth',1,...
         'handlevisibility','off')
% annotate spokes in degrees
    rt = 1.1*rmax;
    for i = 1:length(th)
        text(rt*cst(i),rt*snt(i),int2str(i*30),...
             'horizontalalignment','center',...
             'handlevisibility','off');
        if i == length(th)
           loc = int2str(0);
        else
           loc = int2str(180+i*30);
        end
        text(-rt*cst(i),-rt*snt(i),loc,'horizontalalignment','center',...
             'handlevisibility','off')
    end
% set view to 2-D
    view(2);
% set axis limits
    axis(rmax*[-1 1 -1.15 1.15]);
end
% Reset defaults.
set(cax, 'DefaultTextFontAngle', fAngle , ...
    'DefaultTextFontName',   fName , ...
    'DefaultTextFontSize',   fSize, ...
    'DefaultTextFontWeight', fWeight, ...
    'DefaultTextUnits',fUnits );
% Tranfrom data to dB scale
rmin = 0; rmax=1;
rinc = (rmax-rmin)/rticks;
rhodb=zeros(1,length(rho));
for i=1:length(rho)
    if rho(i)==0
        rhodb(i)=0;
    else
        rhodb(i)=rmax+2*log10(rho(i))*rinc;
    end
    if rhodb(i)<=0
        rhodb(i)=0;
    end
end
```

```
% transform data to Cartesian coordinates.
xx = rhodb.*cos(theta);
yy = rhodb.*sin(theta);
% plot data on top of grid
if strcmp(line_style,'auto')
   q = plot(xx,yy);
else
   q = plot(xx,yy,line_style,'linewidth',1.5);
end
if nargout > 0
   hpol = q;
end
if ~hold_state
   set(gca,'dataaspectratio',[1 1 1]), axis off; set(cax,'NextPlot',next);
end
set(get(gca,'xlabel'),'visible','on')
set(get(gca,'ylabel'),'visible','on')
```

Listing 11.16. MATLAB Function "dbesselj.m"

```
function [ res ] = dbesselj( nu,z )
res=besselj(nu-1,z)-besselj(nu,z)*nu/z;
```

Listing 11.17. MATLAB Function "dbessely.m"

```
function [ res ] = dbessely( nu,z )
res=bessely(nu-1,z)-bessely(nu,z)*nu/z;
```

Listing 11.18. MATLAB Function "dbesselh.m"

```
function [ res ] = dbesselh(nu,kind,z)
res=besselh(nu-1,kind,z)-besselh(nu,kind,z)*nu/z;
```

Listing 11.19. MATLAB Program “rcs_cylinder_complex.m”

```
% This program computes the backscattered RCS for a cylinder
% with flat plates.
clear all
index = 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. (11.37)
  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');
grid;
xlabel ('Aspect angle -degrees');
ylabel ('RCS -dBsm');
```

Listing 11.20. MATLAB Program "Swerling_models.m"

```
% This program computes and plots Swerling statistical models
% sigma_bar = 1.5;
clear all
```

```
sigma = 0:0.001:6;
sigma_bar = 1.5;
swer_3_4 = (4. / sigma_bar^2) .* sigma .* ...
   exp(-2. * (sigma ./ sigma_bar));
%t.*exp(-(t.^2)./2.
swer_1_2 = (1. /sigma_bar) .* exp( -sigma ./ sigma_bar);
plot(sigma,swer_1_2,'k',sigma,swer_3_4,'k');
grid;
gtext ('Swerling I,II');
gtext ('Swerling III,IV');
xlabel ('sigma');
ylabel ('Probability density');
title ('sigma-bar = 1.5');
```

Chapter 12

High Resolution Tactical Synthetic Aperture Radar (TSAR)

This chapter is 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.

12.1. 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.

1. Dr. Brian J. Smith is with the US Army Aviation and Missile Command (AMCOM), Redstone Arsenal, Alabama.

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 correspond to azimuth. This chapter is presented in the context of DBS.

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.

12.2. Side Looking SAR Geometry

Fig. 12.1 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

$$\vec{a}'(t) = 0 \times \hat{a}_x + v \times \hat{a}_y + 0 \times \hat{a}_z \qquad \textbf{(12.1)}$$

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,

$$(t = t_a + t_c) \;\; ; \;\; -\frac{T_{ob}}{2} \le t \le \frac{T_{ob}}{2} \qquad \textbf{(12.2)}$$

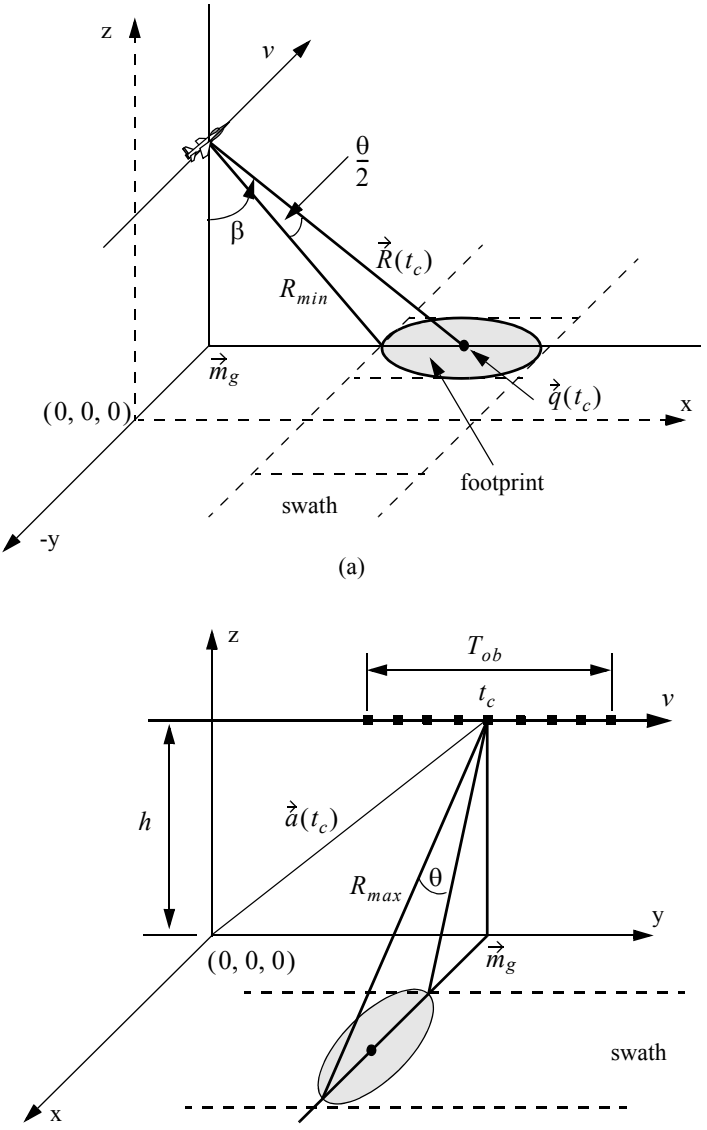

Figure 12.1. Side looking SAR geometry.

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. 12.2. It follows that

$$\begin{aligned} R_{min} &= h/\cos(\beta - \theta/2) \\ R_{max} &= h/\cos(\beta + \theta/2) \\ \left|\vec{R}(t_c)\right| &= h/\cos\beta \end{aligned} \tag{12.3}$$

Notice that the elevation angle β is equal to

$$\beta = 90 - \psi_g \tag{12.4}$$

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. 12.3. 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.

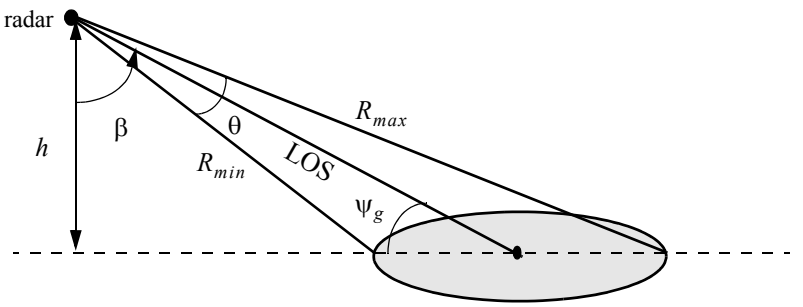

Figure 12.2. Definition of minimum and maximum range.

12.3. SAR Design Considerations

The quality of SAR images is heavily dependent on the size of the map resolution cell shown in Fig. 12.4. The range resolution, ΔR, is computed on the beam LOS, and is given by

$$\Delta R = (c\tau)/2 \tag{12.5}$$

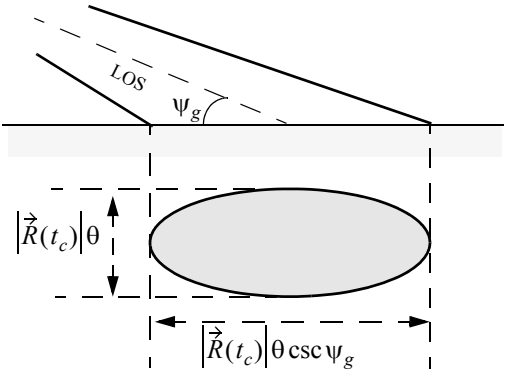

Figure 12.3. Footprint definition.

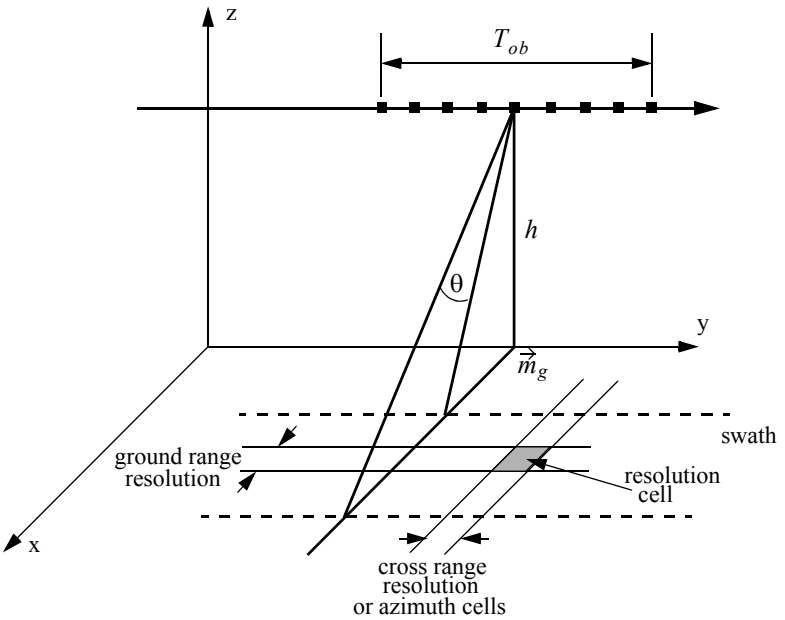

Figure 12.4a. Definition of a resolution cell.

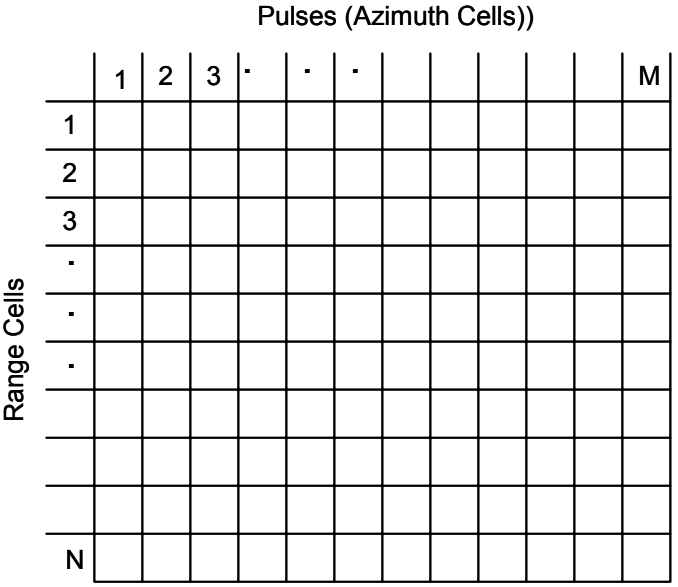

Figure 12.4b. Definition of a resolution cell.

where τ is the pulsewidth. From the geometry in Fig. 12.5 the extent of the range cell ground projection ΔR_g is computed as

$$\Delta R_g = \frac{c\tau}{2}\sec\psi_g \tag{12.6}$$

The azimuth or cross range resolution for a real antenna with a $3dB$ beam-width θ (radians) at range R is

$$\Delta A = \theta R \tag{12.7}$$

However, the antenna beamwidth is proportional to the aperture size,

$$\theta \approx \frac{\lambda}{L} \tag{12.8}$$

where λ is the wavelength and L is the aperture length. It follows that

$$\Delta A = \frac{\lambda R}{L} \tag{12.9}$$

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

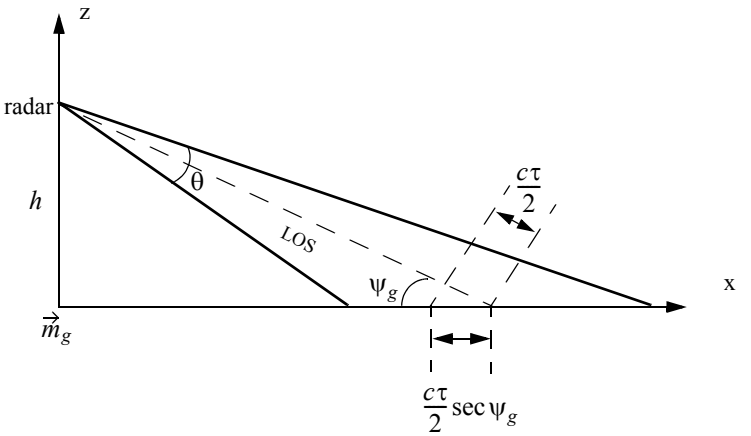

Figure 12.5. Definition of a range cell on the ground.

$$\Delta A = \frac{\lambda R}{2L} \tag{12.10}$$

Furthermore, since the synthetic aperture length L is equal to vT_{ob}, Eq. (12.10) can be rewritten as

$$\Delta A = \frac{\lambda R}{2vT_{ob}} \tag{12.11}$$

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

$$f_d = -\frac{2R'(t)}{\lambda} = \frac{2v_r}{\lambda} \tag{12.12}$$

where $R'(t)$ 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. 12.6 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. 12.7.

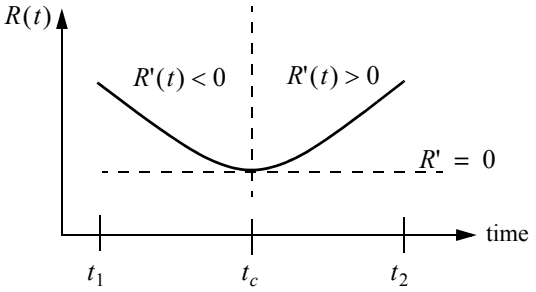

Figure 12.6. Sketch of range versus time for a scatterer.

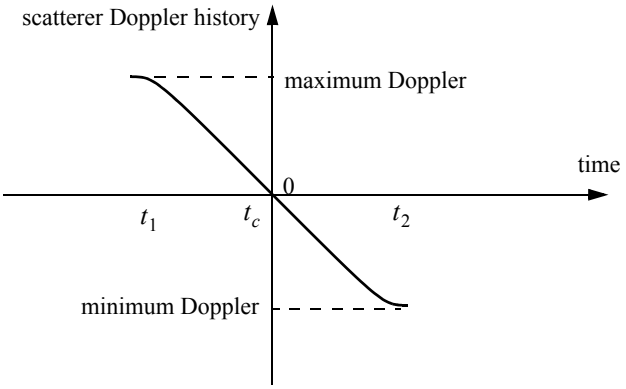

Figure 12.7. Point scatterer Doppler history.

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

$$R_u = R_{max} - R_{min} \tag{12.13}$$

An expression for unambiguous range was derived in Chapter 1, and is repeated here as Eq. (12.14),

$$R_u = \frac{c}{2f_r} \tag{12.14}$$

Combining Eq. (12.14) and Eq. (12.13) yields

$$f_{r_{max}} \leq \frac{c}{2(R_{max} - R_{min})} \tag{12.15}$$

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. 12.8, the maximum and minimum Doppler frequencies are, respectively, given by

$$f_{d_{max}} = \frac{2v}{\lambda} \sin\left(\frac{\theta}{2}\right) \sin\beta \;\; ; \;\; at \;\; t_1 \tag{12.16}$$

$$f_{d_{min}} = -\frac{2v}{\lambda} \sin\left(\frac{\theta}{2}\right) \sin\beta \;\; ; \;\; at \;\; t_2 \tag{12.17}$$

It follows that the maximum Doppler spread is

$$\Delta f_d = f_{d_{max}} - f_{d_{min}} \tag{12.18}$$

Substituting Eqs. (12.16) and (12.17) into Eq. (12.18) and applying the proper trigonometric identities yield

$$\Delta f_d = \frac{4v}{\lambda} \sin\frac{\theta}{2} \sin\beta \tag{12.19}$$

Finally, by using the small angle approximation we get

$$\Delta f_d \approx \frac{4v}{\lambda} \frac{\theta}{2} \sin\beta = \frac{2v}{\lambda} \theta \sin\beta \tag{12.20}$$

Therefore, the minimum PRF is

$$f_{r_{min}} \geq \frac{2v}{\lambda} \theta \sin\beta \tag{12.21}$$

Combining Eqs. (11.15) and (11.21) we get

$$\frac{c}{2(R_{max} - R_{min})} \geq f_r \geq \frac{2v}{\lambda} \theta \sin\beta \tag{12.22}$$

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.

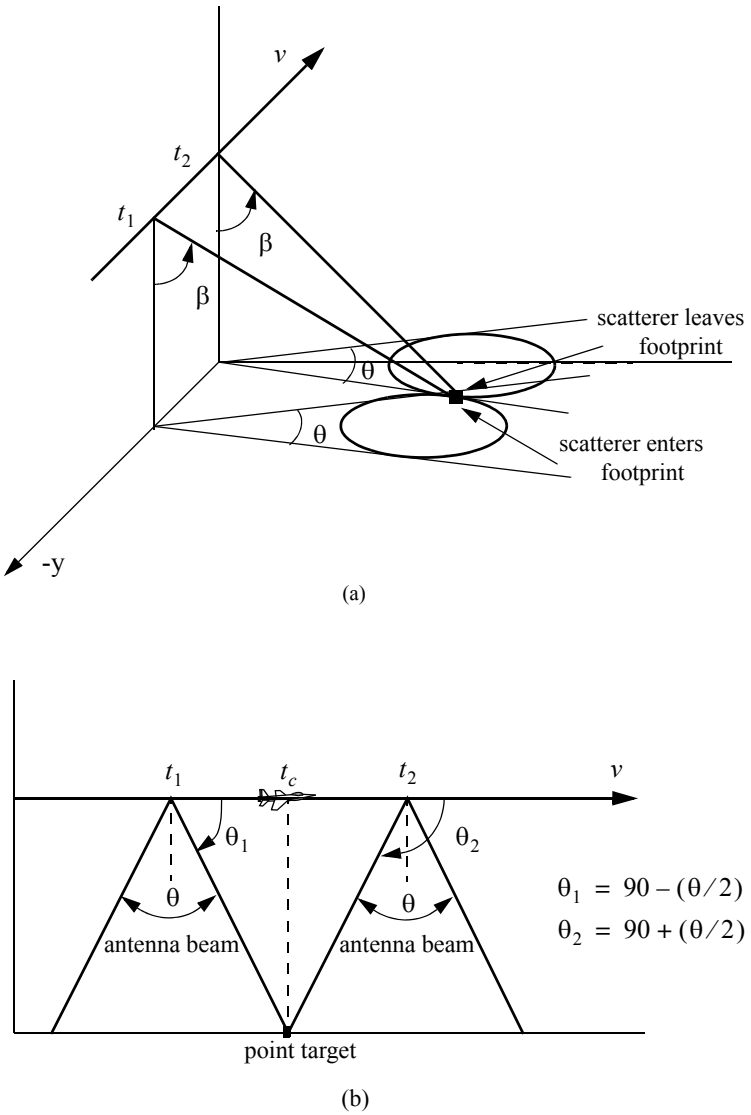

Figure 12.8. Doppler history computation. (a) Full view; (b) top view.

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. (12.20) we get

$$\Delta f_{d_{min}} = \frac{2v}{\lambda} \Delta\theta \sin\beta_k \tag{12.23}$$

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

$$\Delta\theta = \frac{\lambda}{2vT_{ob}\sin\beta_k} \tag{12.24}$$

Substituting L for vT_{ob} yields

$$\Delta\theta = \frac{\lambda}{2L\sin\beta_k} \tag{12.25}$$

Therefore, the SAR azimuth resolution (within the kth range bin) is

$$\Delta A_g = \Delta\theta R_k = R_k \frac{\lambda}{2L\sin\beta_k} \tag{12.26}$$

Note that when $\beta_k = 90°$, Eq. (12.26) is identical to Eq. (12.10).

12.4. SAR Radar Equation

The single pulse radar equation was derived in Chapter 1, and is repeated here as Eq. (12.27),

$$SNR = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R_k^4 k T_0 B L_{Loss}} \tag{12.27}$$

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,

$$\sigma = \sigma^0 \Delta R_g \Delta A_g = \sigma^0 \Delta A_g \frac{c\tau}{2} \sec\psi_g \tag{12.28}$$

where σ^0 is the clutter scattering coefficient, ΔA_g is the azimuth resolution, and Eq. (12.6) was used to replace the ground range resolution. The number of coherently integrated pulses within an observation interval is

$$n = f_r T_{ob} = \frac{f_r L}{v} \tag{12.29}$$

where L is the synthetic aperture size. Using Eq. (12.26) in Eq. (12.29) and rearranging terms yield

$$n = \frac{\lambda R f_r}{2\Delta A_g v} \csc\beta_k \tag{12.30}$$

The radar average power over the observation interval is

$$P_{av} = (P_t / B) f_r \tag{12.31}$$

The SNR for n coherently integrated pulses is then

$$(SNR)_n = nSNR = n \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R_k^4 k T_0 B L_{Loss}} \tag{12.32}$$

Substituting Eqs. (11.31), (11.30), and (11.28) into Eq. (12.32) and performing some algebraic manipulations give the SAR radar equation,

$$(SNR)_n = \frac{P_{av} G^2 \lambda^3 \sigma^0}{(4\pi)^3 R_k^3 k T_0 L_{Loss}} \frac{\Delta R_g}{2v} \csc\beta_k \tag{12.33}$$

Eq. (12.33) 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) function of the ground range resolution; (4) inversely proportional to the velocity v; and (5) proportional to the third power of wavelength.

12.5. 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.

12.6. Side Looking SAR Doppler Processing

Consider the geometry shown in Fig. 12.9, 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

$$s(t) = \cos(2\pi f_0 t - \xi_0) \tag{12.34}$$

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

$$s_i(t, \mu_i) = A_i \cos[2\pi f_0 (t - \tau_i(t, \mu_i)) - \xi_0] \tag{12.35}$$

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

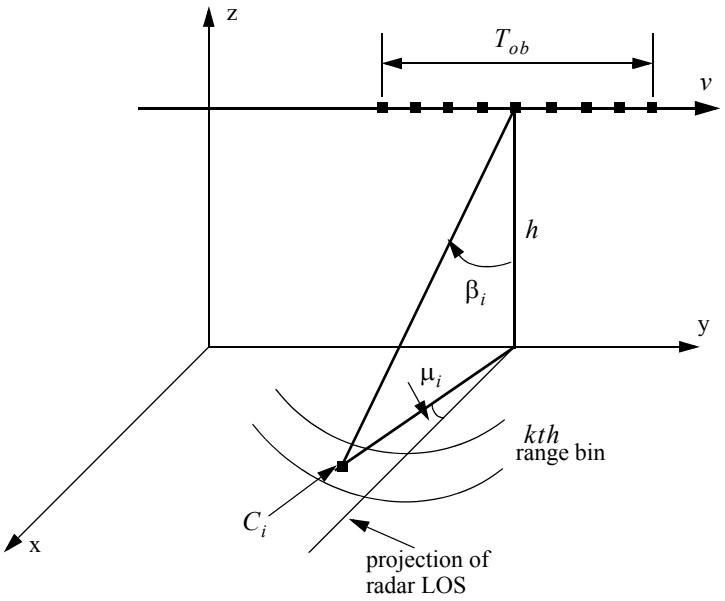

Figure 12.9. A scatterer C_i within the k^{th} range bin.

$$\tau_i(t, \mu_i) = \frac{2r_i(t, \mu_i)}{c} \tag{12.36}$$

where c is the speed of light and $r_i(t, \mu_i)$ is the scatterer slant range. From the geometry in Fig. 12.9, one can write the expression for the slant range to the ith scatterer within the kth range bin as

$$r_i(t, \mu_i) = \frac{h}{\cos\beta_i}\sqrt{1 - \frac{2vt}{h}\cos\beta_i\cos\mu_i\sin\beta_i + \left(\frac{vt}{h}\cos\beta_i\right)^2} \tag{12.37}$$

And by using Eq. (12.36) the round-trip delay can be written as

$$\tau_i(t, \mu_i) = \frac{2}{c}\frac{h}{\cos\beta_i}\sqrt{1 - \frac{2vt}{h}\cos\beta_i\cos\mu_i\sin\beta_i + \left(\frac{vt}{h}\cos\beta_i\right)^2} \tag{12.38}$$

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

$$\tau_i(t, \mu_i) \approx \bar{\tau} + \bar{\tau}_{t\mu}\ \mu_i t + \bar{\tau}_{tt}\ \frac{t^2}{2} \tag{12.39}$$

where the over-bar indicates evaluation at the state $(0, 0)$, and the subscripts denote partial derivatives. For example, $\bar{\tau}_{t\mu}$ means

$$\bar{\tau}_{t\mu} = \frac{\partial^2}{\partial t \partial \mu} \tau_i(t, \mu_i)\Big|_{(t, \mu) = (0, 0)} \tag{12.40}$$

The Taylor series coefficients are

$$\bar{\tau} = \left(\frac{2h}{c}\right) \frac{1}{\cos\beta_i} \tag{12.41}$$

$$\bar{\tau}_{t\mu} = \left(\frac{2v}{c}\right) \sin\beta_i \tag{12.42}$$

$$\bar{\tau}_{tt} = \left(\frac{2v^2}{hc}\right) \cos\beta_i \tag{12.43}$$

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

$$\begin{aligned} s_i(t, \mu_i) &= A_i \cos[\hat{\psi}_i(t, \mu_i) - \xi_0] \\ \hat{\psi}_i(t, \mu_i) &= 2\pi f_0 \left[(1 - \bar{\tau}_{t\mu}\mu_i)t - \bar{\tau} - \bar{\tau}_{tt}\frac{t^2}{2}\right] \end{aligned} \tag{12.44}$$

Observation of Eq. (12.44) 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. 12.10.

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. (12.44). In other words, superposition holds, and the overall echo signal is

$$s_r(t) = \sum_{i=1}^{I} s_i(t, \mu_i) \tag{12.45}$$

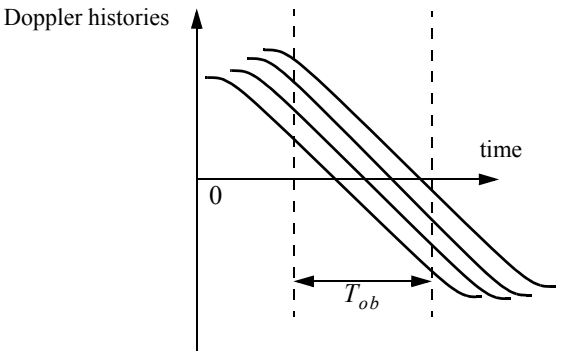

Figure 12.10. Doppler histories for several scatterers within the same range bin.

A signal processing block diagram for the kth range bin is illustrated in Fig. 12.11. It consists of the following steps. First, heterodyning with the carrier frequency is performed to extract the quadrature components.

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

$$\begin{aligned} \tilde{x}_I(t_n) &= \tilde{x}_I(n) = A_i \cos[\tilde{\psi}_i(t_n, \mu_i) - \xi_0] \\ \tilde{x}_Q(t_n) &= \tilde{x}_Q(n) = A_i \sin[\tilde{\psi}_i(t_n, \mu_i) - \xi_0] \end{aligned} \tag{12.46}$$

$$\tilde{\psi}_i(t_n, \mu_i) = \hat{\psi}_i(t_n, \mu_i) - 2\pi f_0 t_n \tag{12.47}$$

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

$$\begin{bmatrix} x_I(n) \\ x_Q(n) \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \tilde{x}_I(n) \\ \tilde{x}_Q(n) \end{bmatrix} \tag{12.48}$$

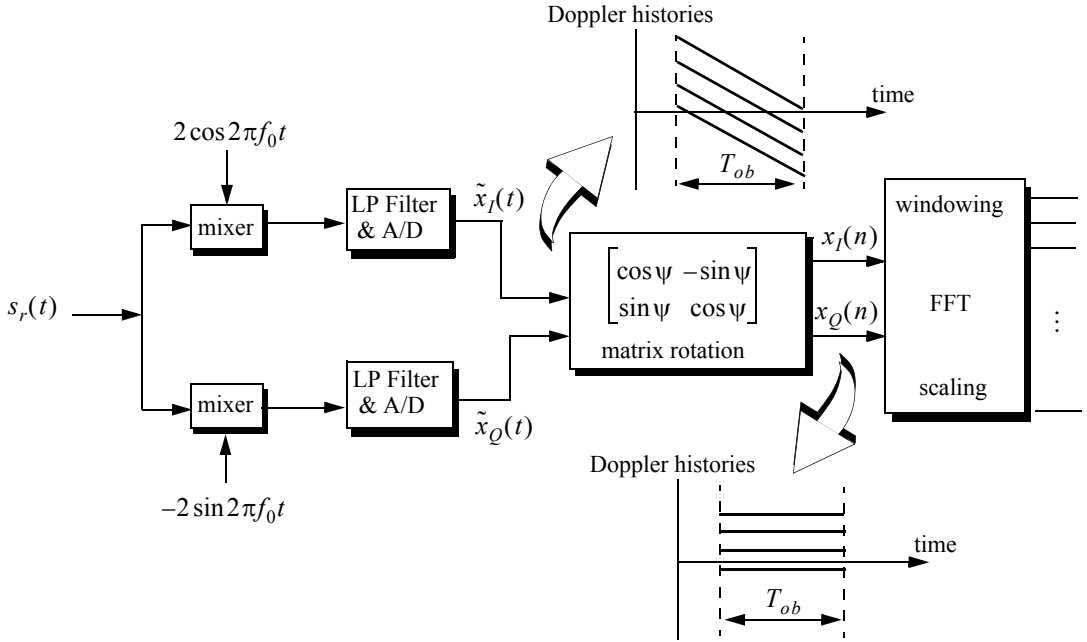

Figure 12.11. Signal processing block diagram for the k^{th} range bin.

12.7. 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

$$f_{di} = \frac{1}{2\pi} \frac{d\psi}{dt} = f_0 \dot{\bar{\tau}}_{t\mu} \mu_i = \frac{2v}{\lambda} \sin\beta_i \mu_i \qquad \textbf{(12.49)}$$

This is the same result derived in Eq. (12.23), with $\mu_i = \Delta\theta$. Therefore, the scatterers separated in Doppler by more than Δf can then be resolved.

Fig. 12.12 shows a two-dimensional SAR image for three point scatterers located 10 Km down-range. In this case, the azimuth and range resolutions are equal to 1 m and the operating frequency is 35GHz. Fig. 12.13 is similar to Fig. 12.12, except in this case the resolution cell is equal to 6 inches. One can clearly see the blurring that occurs in the image. Figs. 12.12 and 12.13 can be reproduced using the program *"fig12_12_13.m"* given in Listing 12.1 in Section 12.10.

12.8. 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.

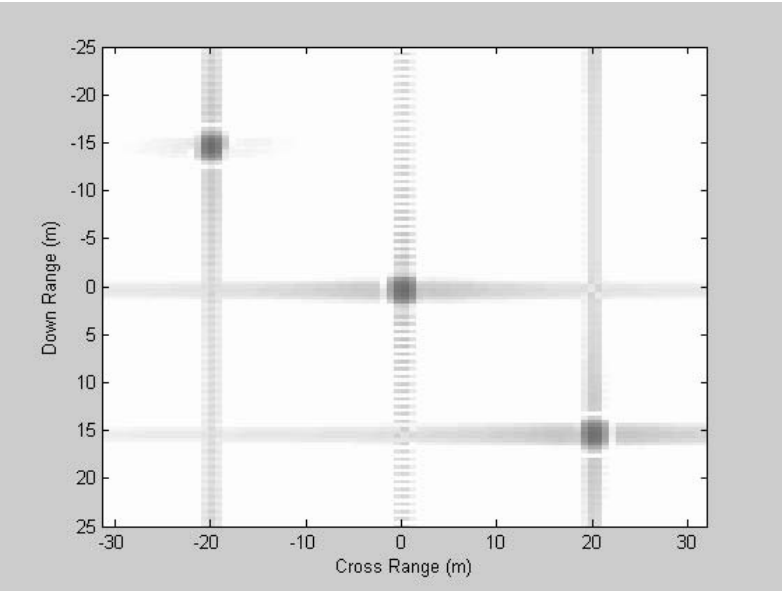

Figure 12.12. Three point scatterer image. Resolution cell is $1m^2$.

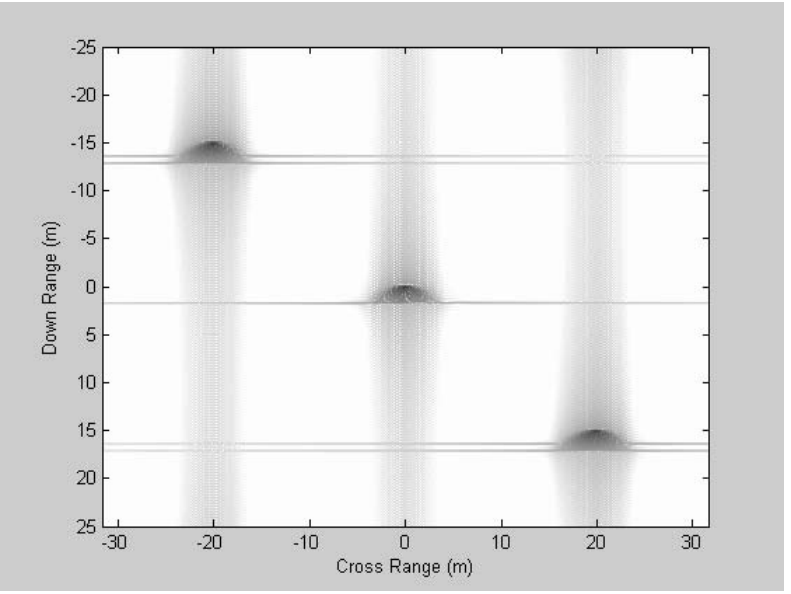

Figure 12.13. Three point scatterer image. Resolution cell is squared inches.

12.9. A Three-Dimensional SAR Imaging Technique

This section presents a new three-dimensional (3-D) Synthetic Aperture Radar (SAR) imaging technique.[1] 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. 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.

12.9.1. 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 wide band 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

1. This section is extracted from: 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.

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.

12.9.2. 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 $\sin\beta_l$. 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

$$x_n = \left(-\frac{N-1}{2} + n\right); n = 0, \ldots N-1 \tag{12.50}$$

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

$$E(x_1, x_2; s_l) = G^2(s_l)\left(\frac{R_0}{R}\right)^4 \sqrt{\sigma_l}\ exp(j\phi(x_1, x_2; s_l)) \tag{12.51}$$

$$\phi(x_1, x_2; s_l) = \frac{2\pi}{\lambda}(x_1 + x_2)(s_l) \tag{12.52}$$

$$s_l = \sin\beta_l \tag{12.53}$$

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

$$E(x_1, x_2) = \sum_{all\ l} [E_I(x_1, x_2; s_l) + jE_Q(x_1, x_2; s_l)] \tag{12.54}$$

where the subscripts (I, Q) denote the quadrature components. Note that the variable part of the phase given in Eq. (12.52) 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 $m = n1 + n2;\ (n1, n2) = \ldots 0, N-1$

$$b(m) = b(m) + E(x_{n1}, x_{n2}) \tag{12.55}$$

$$c(m) = c(m) + 1 \tag{12.56}$$

It follows that

$$\{c(m); m = 0, \ldots 2N-2\} = \left\{ \begin{array}{l} m+1 \ ; \ m = 0, \ldots N-2 \\ N \ ; \ m = N-1 \\ 2N-1-m \quad m = N, \ldots 2N-2 \end{array} \right\} \tag{12.57}$$

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,

$$B(q) = \sum_{m=0}^{N_F-1} b'(m) \cdot exp\left(-j\frac{2\pi qm}{N_F}\right); q = 0, \ldots N_F - 1 \tag{12.58}$$

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

$$\sin\beta_q = \frac{\lambda q}{2Nd}; q = 0, \ldots 2N-1 \tag{12.59}$$

where $\sin\beta_q$ denotes the direction-sine of the q^{th} scatterer, and $N_F = 2N$ is implied in Eq. (12.59).

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. (12.59), 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's 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

$$x_n = \left(-\frac{N-1}{2} + n\right) \quad ;n = 0, \ldots N-1 \tag{12.60}$$

$$y_n = \left(-\frac{N-1}{2} + n\right) \quad ;n = 0, \ldots N-1 \tag{12.61}$$

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

$$\varphi'(x, y) = \frac{2\pi}{\lambda}[x \sin\beta \cos\alpha + y \sin\beta \sin\alpha] \tag{12.62}$$

Therefore, the two-way geometric phase between the (x_1, y_1) and (x_2, y_2) elements is

$$\varphi(x_1, y_1, x_2, y_2) = \frac{2\pi}{\lambda} \sin\beta[(x_1 + x_2)\cos\alpha + (y_1 + y_2)\sin\alpha] \tag{12.63}$$

The two-way electric field for the l^{th} scatterer at (α_l, β_l) is

$$E(x_1, x_2, y_1, y_2; \alpha_l, \beta_l) = G^2(\beta_l)\left(\frac{R_0}{R}\right)^4 \sqrt{\sigma_l}\ exp[j(\varphi(x_1, y_1, x_2, y_2))] \tag{12.64}$$

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

$$E(x_1, x_2, y_1, y_2) = \sum_{all\ scatterers} E(x_1, x_2, y_1, y_2; \alpha_l, \beta_l) \tag{12.65}$$

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. (12.63) 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

$$\{i = x_{n1} + x_{n2}; i = -(N-1), \ldots (N-1)\} \tag{12.66}$$

$$\{j = y_{n1} + y_{n2}; j = -(N-1), \ldots (N-1)\} \tag{12.67}$$

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,

$$\begin{aligned} &for\ \ m_x = n1 + n2\ \ and\ \ m_y = n1 + n2 \\ &n1 = 0, \ldots N-1 \quad , and\ \ n2 = 0, \ldots N-1 \end{aligned} \tag{12.68}$$

$$b(m_x, m_y) = b(m_x, m_y) + E(x_{n1}, y_{n1}, x_{n2}, y_{n2}) \tag{12.69}$$

It follows that

$$c(m_x, m_y) = (N_x - |m_x - (N_x - 1)|) \times (N_y - |m_y - (N_y - 1)|) \tag{12.70}$$

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

$$\alpha_l = \text{atan}\left(\frac{q_l}{p_l}\right) \tag{12.71}$$

$$\sin\beta_l = \frac{\lambda p_l}{2Nd\cos\alpha_l} = \frac{\lambda q_l}{2Nd\sin\alpha_l} \tag{12.72}$$

Derivation of Eq. (12.71) is in Section 12.9.7.

12.9.3. Geometry for DFTSQM SAR Imaging

Fig. 12.14 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 rela-

tive 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. 12.15.

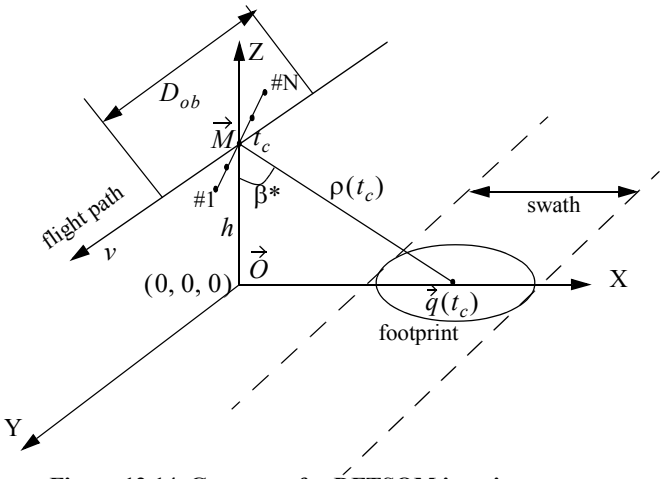

Figure 12.14. Geometry for DFTSQM imaging system.

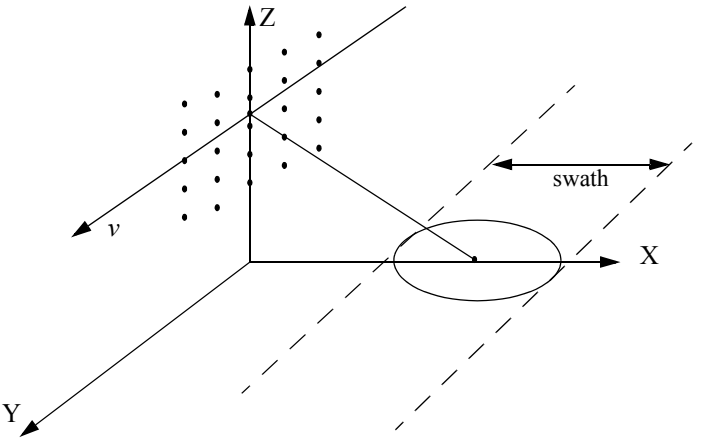

Figure 12.15. Synthesized 2-D array.

The vector $\hat{q}(t_c)$ defines the center of the 3 dB 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 $\hat{a}$ along the real array is

$$\hat{a} = \cos\beta^* \hat{a}_x + \sin\beta^* \hat{a}_z \tag{12.73}$$

where β^* is the elevation angle, or the complement of the depression angle, for the center of the footprint at central time t_c.

12.9.4. Slant Range Equation

Consider the geometry shown in Fig. 12.16 and assume that there is a scatterer $\vec{C}_i$ within the k^{th} range cell. This scatterer is defined by

$$\{ampltiude, phase, elevation, azimuth, height\} = \{a_i, \phi_i, \beta_i, \mu_i, \tilde{h}_i\} \tag{12.74}$$

The scatterer $\vec{C}_i$ (assuming rectangular coordinates) is given by

$$\vec{C}_i = h\tan\beta_i \cos\mu_i \hat{a}_x + h\tan\beta_i \sin\mu_i \hat{a}_y + \tilde{h}_i \hat{a}_z \tag{12.75}$$

$$\beta_i = \beta_k + \varepsilon \tag{12.76}$$

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

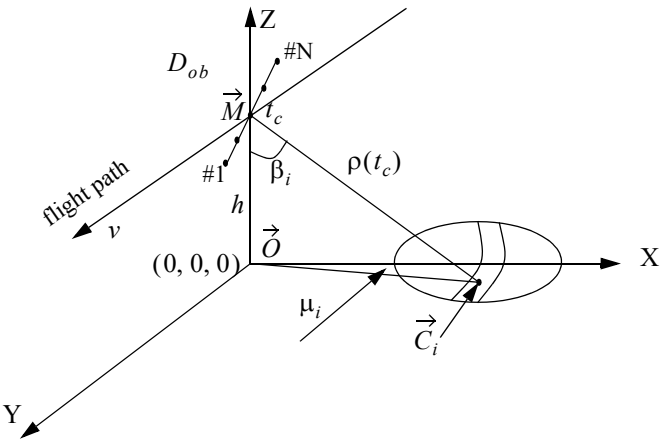

Figure 12.16. Scatterer $\vec{C}_i$ within a range cell.

$$\vec{O}e_n = D_n \cos\beta^* \hat{a}_x + vt\hat{a}_y + (D_n \sin\beta^* + h)\hat{a}_z \tag{12.77}$$

$$D_n = \left(\frac{1-N}{2} + n\right)d \quad ;n = 0, \ldots N-1 \tag{12.78}$$

The range between a scatterer $\vec{C}$ within the k^{th} range cell and the n^{th} element of the real array is

$$r_n^2(t, \varepsilon, \mu, \tilde{h}; D_n) = D_n^2 + v^2t^2 + (h - \tilde{h})^2 + 2D_n \sin\beta^*(h - \tilde{h}) + h\tan(\beta_k + \varepsilon)[h\tan(\beta_k + \varepsilon) - 2D_n \cos\beta^* \cos\mu - 2vt\sin\mu] \tag{12.79}$$

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

$$s = \sin\mu \tag{12.80}$$

$$u = \sin\varepsilon \tag{12.81}$$

Then, one can rewrite Eq. (12.79) as

$$r_n^2(t, s, u, \tilde{h}; D_n) = D_n^2 + v^2t^2 + (h - \tilde{h})^2 + h^2 f^2(u) + 2D_n \sin\beta^*(h - \tilde{h}) - (2D_n h \cos\beta^* f(u)\sqrt{1 - s^2} - 2vhtf(u)s) \tag{12.82}$$

$$f(u) = \tan(\beta_k + \text{asin}\, u) \tag{12.83}$$

Expanding r_n as a third order Taylor series expansion about incremental (t, s, u, h) yields

$$r(t, s, u, \tilde{h}; D_n) = \bar{r} + \bar{r}_{\tilde{h}}\tilde{h} + \bar{r}_u u + \bar{r}_{\tilde{h}\tilde{h}}\frac{\tilde{h}^2}{2} + \bar{r}_{\tilde{h}u}\tilde{h}u + \bar{r}_{ss}\frac{s^2}{2} + \bar{r}_{st}st + \bar{r}_{tt}\frac{t^2}{2} + \bar{r}_{uu}\frac{u^2}{2} + \bar{r}_{\tilde{h}\tilde{h}\tilde{h}}\frac{\tilde{h}^3}{6} + \bar{r}_{\tilde{h}\tilde{h}u}\frac{\tilde{h}^2 u}{2} + \bar{r}_{\tilde{h}st}\tilde{h}st + \bar{r}_{\tilde{h}uu}\frac{\tilde{h}u^2}{2} + \bar{r}_{\tilde{h}ss}\frac{\tilde{h}s^2}{2} + \bar{r}_{uss}\frac{us^2}{2} + \bar{r}_{stu}stu + \bar{r}_{suu}\frac{su^2}{2} + \bar{r}_{t\tilde{h}\tilde{h}}\frac{t\tilde{h}^2}{2} + \bar{r}_{utt}\frac{ut^2}{2} + \bar{r}_{uuu}\frac{u^3}{6} \tag{12.84}$$

where subscripts denote partial derivations, and the over-bar indicates evaluation at the state $(t, s, u, h) = (0, 0, 0, 0)$. Note that

$$\{\bar{r}_s = \bar{r}_t = \bar{r}_{\tilde{h}s} = \bar{r}_{\tilde{h}t} = \bar{r}_{su} = \bar{r}_{tu} = \bar{r}_{\tilde{h}\tilde{h}s} = \bar{r}_{\tilde{h}\tilde{h}t} = \bar{r}_{\tilde{h}su} = \bar{r}_{\tilde{h}tu} = \bar{r}_{sss} = \bar{r}_{sst} = \bar{r}_{stt} = \bar{r}_{ttt} = \bar{r}_{tsu} = 0\} \tag{12.85}$$

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:

$$\left\{ \bar{r}_{\tilde{h}\tilde{h}\tilde{h}}\frac{\tilde{h}^3}{6}, \bar{r}_{\tilde{h}\tilde{h}u}\frac{\tilde{h}^2 u}{2}, \bar{r}_{\tilde{h}uu}\frac{\tilde{h}u^2}{2}, \bar{r}_{\tilde{h}ss}\frac{\tilde{h}s^2}{2}, \right. \tag{12.86}$$

$$\left. \bar{r}_{uss}\frac{us^2}{2}, \bar{r}_{stu}stu, \bar{r}_{suu}\frac{su^2}{2}, \bar{r}_{t\tilde{h}\tilde{h}}\frac{t\tilde{h}^2}{2}, \bar{r}_{utt}\frac{ut^2}{2}, \bar{r}_{uuu}\frac{u^3}{6} \right\}$$

are small and can be neglected. Thus, the range r_n is approximated by

$$r(t, s, u, \tilde{h}; D_n) = \bar{r} + \bar{r}_{\tilde{h}}\tilde{h} + \bar{r}_u u + \bar{r}_{\tilde{h}\tilde{h}}\frac{\tilde{h}^2}{2} + \bar{r}_{\tilde{h}u}\tilde{h}u + \tag{12.87}$$

$$\bar{r}_{ss}\frac{s^2}{2} + \bar{r}_{st}st + \bar{r}_{tt}\frac{t^2}{2} + \bar{r}_{uu}\frac{u^2}{2} + \bar{r}_{\tilde{h}st}\tilde{h}st$$

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

$$\tau_{n_1 n_2} = \frac{1}{c}(r_{n_1}(t, s, u, \tilde{h}; D_{n_1}) + r_{n_2}(t, s, u, \tilde{h}; D_{n_2})) \tag{12.88}$$

where c is the speed of light.

12.9.5. 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

$$\Delta t > N\frac{2R_{mx}}{c} \tag{12.89}$$

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

$$s_n(t_{mn}) = \cos(2\pi f_o t_{mn} + \zeta) \tag{12.90}$$

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

$$s_i(n_1, n_2; t_{mn_1}) = a_i G^2 (\sin\beta_i)(\rho_k(t_c)/\rho(t_c))^4 \cos[2\pi f_o(t_{mn_1} - \tau_{n_1 n_2}) + \zeta - \phi_i] \quad (12.91)$$

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 hereon 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

$$\{(a_i, \phi_i, s_i, u_i, \tilde{h}_i); i = 1, \ldots N_o\} \quad (12.92)$$

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

$$s(n_1, n_2; t_{mn_1}) = \sum_{i=1}^{N_o} s_i(n_1, n_2; t_{mn_1}) \quad (12.93)$$

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

$$\{b_k(0, m), b_k(1, m), b_k(2, m), b_k(3, m), b_k(4, m), b_k(5, m), b_k(6, m), b_k(7, m)\} = \{s(0, 0; t_{mn_0}), s(0, 1; t_{mn_0}) + s(1, 0; t_{mn_1}), s(0, 2; t_{mn_0}) + s(1, 1; t_{mn_1}) + s(2, 0; t_{mn_2}), s(0, 3; t_{mn_0}) + s(1, 2; t_{mn_1}) + s(2, 1; t_{mn_2}) + s(3, 0; t_{mn_3}), s(1, 3; t_{mn_1}) + s(2, 2; t_{mn_2}) + s(3, 1; t_{mn_3}), s(2, 3; t_{mn_2}) + s(3, 2; t_{mn_3}), s(3, 3; t_{mn_3}), 0\} \quad (12.94)$$

12.9.6. 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

$$s_I^{\perp}(n_1, n_2;t) = B\cos\psi^{\perp} \tag{12.95}$$

$$s_Q^{\perp}(n_1, n_2;t) = B\sin\psi^{\perp} \tag{12.96}$$

$$\psi^{\perp} = 2\pi f_0\left\{ t - \frac{1}{c}\left[2\bar{r} + (\bar{r}_{\tilde{h}}(D_{n_1}) + \bar{r}_{\tilde{h}}(D_{n_2}))\tilde{h} + (\bar{r}_u(D_{n_1}) + \bar{r}_u(D_{n_2}))u + \right.\right. \tag{12.97}$$

$$(\bar{r}_{\tilde{h}\tilde{h}}(D_{n_1}) + \bar{r}_{\tilde{h}\tilde{h}}(D_{n_2}))\frac{\tilde{h}^2}{2} + (\bar{r}_{\tilde{h}u}(D_{n_1}) + \bar{r}_{\tilde{h}u}(D_{n_2}))\tilde{h}u +$$

$$(\bar{r}_{ss}(D_{n_1}) + \bar{r}_{ss}(D_{n_2}))\frac{s^2}{2} + 2\bar{r}_{st}st + 2\bar{r}_{tt}\frac{t^2}{2} +$$

$$\left.\left.(\bar{r}_{uu}(D_{n_1}) + \bar{r}_{uu}(D_{n_2}))\frac{u^2}{2} + (\bar{r}_{\tilde{h}st}(D_{n_1}) + \bar{r}_{\tilde{h}st}(D_{n_2}))\tilde{h}st\right] \right\}$$

where B denotes antenna gain, range attenuation, and scatterers' strengths. The subscripts for t have been dropped for notation simplicity. Rearranging Eq. (12.97) and collecting terms yields

$$\psi^{\perp} = \frac{2\pi f_0}{c}\left\{ \{tc - [2\bar{r}_{st}s + (\bar{r}_{\tilde{h}st}(D_{n_1}) + \bar{r}_{\tilde{h}st}(D_{n_2}))\tilde{h}s]t - \bar{r}_{tt}t^2\} - \right. \tag{12.98}$$

$$\left[2\bar{r} + (\bar{r}_{\tilde{h}}(D_{n_1}) + \bar{r}_{\tilde{h}}(D_{n_2}))\tilde{h} + (\bar{r}_u(D_{n_1}) + \bar{r}_u(D_{n_2}))u + \right.$$

$$(\bar{r}_{uu}(D_{n_1}) + \bar{r}_{uu}(D_{n_2}))\frac{u^2}{2} + (\bar{r}_{\tilde{h}\tilde{h}}(D_{n_1}) + \bar{r}_{\tilde{h}\tilde{h}}(D_{n_2}))\frac{\tilde{h}^2}{2} +$$

$$\left.\left.(\bar{r}_{\tilde{h}u}(D_{n_1}) + \bar{r}_{\tilde{h}u}(D_{n_2}))\tilde{h}u + (\bar{r}_{ss}(D_{n_1}) + \bar{r}_{ss}(D_{n_2}))\frac{s^2}{2}\right]\right\}$$

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

$$s_I(n_1, n_2;t) = B\cos\psi \tag{12.99}$$

$$s_Q(n_1, n_2;t) = B\sin\psi \tag{12.100}$$

$$\psi = \psi^{\perp} - 2\pi f_0 t + 2\pi f_0 \bar{r}_{tt} \frac{t^2}{c} \tag{12.101}$$

The instantaneous frequency for the i^{th} scatterer within the k^{th} range cell is computed as

$$f_{di} = \frac{1}{2\pi} \frac{d\psi}{dt} = -\frac{f_0}{c}[2\bar{r}_{st}s + (\bar{r}_{\tilde{h}st}(D_{n_1}) + \bar{r}_{\tilde{h}st}(D_{n_2}))\tilde{h}s] \tag{12.102}$$

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

$$f_{di} = -\left(\frac{2v\sin\beta_k}{\lambda}\right)\left(\frac{\tilde{h}s}{\rho_k^2(t_c)}(h + (D_{n_1} + D_{n_2})\sin\beta^*) - s\right) \tag{12.103}$$

Note that if $\tilde{h} = 0$, then

$$f_{di} = \frac{2v}{\lambda} \sin\beta_k \sin\mu \tag{12.104}$$

which is the Doppler value corresponding to a ground patch (see Eq. (12.49)).

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.

12.9.7. Derivation of Eq. (12.71)

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

$$\hat{u} = \sin\beta\cos\alpha \ \hat{a}_x + \sin\beta\sin\alpha \ \hat{a}_y + \cos\beta \ \hat{a}_z \tag{12.105}$$

The $(n_x, n_y)^{th}$ element of the array can be defined by the vector

$$\hat{e}(n_x, n_y) = \left(n_x - \frac{N-1}{2}\right)d \ \hat{a}_x + \left(n_y - \frac{N-1}{2}\right)d \ \hat{a}_y \tag{12.106}$$

where $(n_x, n_y = 0, \ldots N-1)$. The one-way geometric phase for this element is

$$\varphi'(n_x, n_y) = k(\hat{u} \bullet \hat{e}(n_x, n_y)) \tag{12.107}$$

where $k = 2\pi/\lambda$ is the wave-number, 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

$$\varphi(n_{x1}, n_{y1}, n_{x2}, n_{y2}) = k[\hat{u} \bullet \{\hat{e}(n_{x1}, n_{y1}) + \hat{e}(n_{x2}, n_{y2})\}] \quad \textbf{(12.108)}$$

The cumulative two-way normalized electric field due to all transmissions is

$$E(\hat{u}) = E_t(\hat{u})E_r(\hat{u}) \quad \textbf{(12.109)}$$

where the subscripts t and r, respectively, refer to the transmitted and received electric fields. More precisely,

$$E_t(\hat{u}) = \sum_{n_{xt}=0}^{N-1} \sum_{n_{yt}=0}^{N-1} w(n_{xt}, n_{yt}) exp[jk\{\hat{u} \bullet \hat{e}(n_{xt}, n_{yt})\}] \quad \textbf{(12.110)}$$

$$E_r(\hat{u}) = \sum_{n_{xr}=0}^{N-1} \sum_{n_{yr}=0}^{N-1} w(n_{xr}, n_{yr}) exp[jk\{\hat{u} \bullet \hat{e}(n_{xr}, n_{yr})\}] \quad \textbf{(12.111)}$$

In this case, $w(n_x, n_y)$ denotes the tapering sequence. Substituting Eqs. (12.108), (12.110), and (12.111) into Eq. (12.109) and grouping all fields with the same two-way geometric phase yields

$$E(\hat{u}) = e^{j\delta} \sum_{m=0}^{N_a-1} \sum_{n=0}^{N_a-1} w'(m, n) exp[jkd \sin\beta(m\cos\alpha + n\sin\alpha)] \quad \textbf{(12.112)}$$

$$N_a = 2N - 1 \quad \textbf{(12.113)}$$

$$m = n_{xt} + n_{xr}; m = 0, \ldots 2N-2 \quad \textbf{(12.114)}$$

$$n = n_{yt} + n_{yr}; n = 0, \ldots 2N-2 \quad \textbf{(12.115)}$$

$$\delta = \left(\frac{-d\sin\beta}{2}\right)(N-1)(\cos\alpha + \sin\alpha) \quad \textbf{(12.116)}$$

The two-way array pattern is then computed as

$$|E(\hat{u})| = \left| \sum_{m=0}^{N_a-1} \sum_{n=0}^{N_a-1} w'(m, n) exp[jkd \sin\beta(m\cos\alpha + n\sin\alpha)] \right| \quad \textbf{(12.117)}$$

Consider the two-dimensional DFT transform, $W'(p, q)$, of the array $w'(n_x, n_y)$

$$W'(p,q) = \tag{12.118}$$

$$\sum_{m=0}^{N_a-1}\sum_{n=0}^{N_a-1} w'(m,n)exp\left(-j\frac{2\pi}{N_a}(pm+qn)\right);p,q = 0,\ldots N_a-1$$

Comparison of Eqs. (12.117) and Eq. (12.118) indicates that $|E(\tilde{u})|$ is equal to $|W'(p,q)|$ if

$$-\left(\frac{2\pi}{N_a}\right)p = \frac{2\pi}{\lambda}d\sin\beta\cos\alpha \tag{12.119}$$

$$-\left(\frac{2\pi}{N_a}\right)q = \frac{2\pi}{\lambda}d\sin\beta\sin\alpha \tag{12.120}$$

It follows that

$$\alpha = \tan^{-1}\left(\frac{q}{p}\right) \tag{12.121}$$

12.9.8. Non-Zero Taylor Series Coefficients for the k^{th} Range Cell

$$\bar{r} = \sqrt{D_n^2 + h^2(1+\tan\beta_k) + 2hD_n\sin\beta^* - 2hD_n\cos\beta^*\tan\beta_k} = \rho_k(t_c) \tag{12.122}$$

$$\bar{r}_{\tilde{h}} = \left(\frac{-1}{\bar{r}}\right)(h + D_n\sin\beta^*) \tag{12.123}$$

$$\bar{r}_u = \left(\frac{h}{\bar{r}\cos^2\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) \tag{12.124}$$

$$\bar{r}_{\tilde{h}\tilde{h}} = \left(\frac{1}{\bar{r}}\right) - \left(\frac{1}{\bar{r}^3}\right)(h + D_n\sin\beta^*) \tag{12.125}$$

$$\bar{r}_{\tilde{h}u} = \left(\frac{1}{\bar{r}^3}\right)\left(\frac{h}{\cos^2\beta_k}\right)(h + D_n\tan\beta^*)(h\tan\beta_k - D_n\cos\beta^*) \tag{12.126}$$

$$\bar{r}_{ss} = \left(\frac{-1}{4\bar{r}^3}\right) + \left(\frac{1}{\bar{r}}\right)(h\tan\beta_k - D_n\cos\beta^*) \tag{12.127}$$

$$\bar{r}_{st} = \left(\frac{-1}{\bar{r}}\right)hv\tan\beta_k \tag{12.128}$$

$$\bar{r}_{tt} = \frac{v^2}{\bar{r}} \tag{12.129}$$

$$\bar{r}_{uu} = \left(\frac{h}{\bar{r}\cos^3\beta_k}\right)\left\{\left(\frac{h}{\bar{r}^2\cos\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) + h\left(\left(\frac{1}{\cos\beta_k}\right) + 2\tan\beta_k\sin\beta_k\right) - 2\sin\beta_k D_n\cos\beta^*\right\} \quad \textbf{(12.130)}$$

$$\bar{r}_{\tilde{h}\tilde{h}\tilde{h}} = \left(\frac{3}{\bar{r}^3}\right)(h + D_n\sin\beta^*)\left[\left(\frac{1}{\bar{r}^2}\right)(h + D_n\sin\beta^*)^2 - 1\right] \quad \textbf{(12.131)}$$

$$\bar{r}_{\tilde{h}\tilde{h}u} = \left(\frac{h}{\bar{r}^3\cos^2\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*)\left[\left(\frac{-3}{\bar{r}^2}\right)(h + D_n\sin\beta^*)^2 + 1\right] \quad \textbf{(12.132)}$$

$$\bar{r}_{\tilde{h}st} = \left(\frac{hv\tan\beta_k}{\bar{r}^3}\right)(h + D_n\sin\beta^*) \quad \textbf{(12.133)}$$

$$\bar{r}_{\tilde{h}uu} = \left(\frac{-3}{\bar{r}^5}\right)\left(\frac{h^2}{\cos^4\beta_k}\right)(h + D_n\sin\beta^*)(h\tan\beta_k - D_n\cos\beta^*) \quad \textbf{(12.134)}$$

$$\bar{r}_{\tilde{h}ss} = \left(\frac{-1}{\bar{r}^3}\right)(h\tan\beta_k - D_n\cos\beta^*)(h + D_n\sin\beta^*) \quad \textbf{(12.135)}$$

$$\bar{r}_{uss} = \left(\frac{h}{\bar{r}\cos^2\beta_k}\right)(D_n\cos\beta^*)\left[\left(\frac{1}{\bar{r}^2}\right)(h\tan\beta_k - D_n\cos\beta^*)(h\tan\beta_k) + 1\right] \quad \textbf{(12.136)}$$

$$\bar{r}_{stu} = \left(\frac{-h\tan\beta_k}{\bar{r}^3\cos^2\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) \quad \textbf{(12.137)}$$

$$\bar{r}_{suu} = \left(\frac{hD_n\cos\beta^*}{\bar{r}\cos^2\beta_k}\right)\left[\left(\frac{h\tan\beta_k}{\bar{r}^2}\right)(h\tan\beta_k - D_n\cos\beta^*) + 1\right] \quad \textbf{(12.138)}$$

$$\bar{r}_{\tilde{h}tt} = \left(\frac{v^2h}{\bar{r}^3\cos^2\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) \quad \textbf{(12.139)}$$

$$\bar{r}_{uuu} = \quad \textbf{(12.140)}$$

$$\left(\frac{h}{\bar{r}\cos^4\beta_k}\right)[8h\tan\beta_k + \sin^2\beta_k(h - D_n\cos\beta^*) - 2D_n\cos\beta^*] +$$
$$\left(\frac{3h^2}{\bar{r}^3\cos^5\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*) + \left[\left(\frac{3h^2}{\bar{r}^3\cos^5\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*)\right.$$
$$\left.\left(\frac{1}{2\cos\beta_k} + (h\tan\beta_k - D_n\cos\beta^*)\right)\right] + \left(\frac{3h^3}{\bar{r}^5\cos^6\beta_k}\right)(h\tan\beta_k - D_n\cos\beta^*)$$

12.10. MATLAB Programs and Functions

Listing 12.1. MATLAB Program "fig12_12-13.m"

```
%                  Figures 12.12 and 12.13
%   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
```

Chapter 13 *Signal Processing*

13.1. Signal and System Classifications

In general, electrical signals can represent either current or voltage, and may be classified into two main categories: energy signals and power signals. Energy signals can be deterministic or random, while power signals can be periodic or random. A signal is said to be random if it is a function of a random parameter (such as random phase or random amplitude). Additionally, signals may be divided into low pass or band pass signals. Signals that contain very low frequencies (close to DC) are called low pass signals; otherwise they are referred to as band pass signals. Through modulation, low pass signals can be mapped into band pass signals.

The average power P for the current or voltage signal $x(t)$ over the interval (t_1, t_2) across a 1Ω resistor is

$$P = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} |x(t)|^2 \, dt \tag{13.1}$$

The signal $x(t)$ is said to be a power signal over a very large interval $T = t_2 - t_1$, if and only if it has finite power; it must satisfy the following relation:

$$0 < \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{T/2} |x(t)|^2 \, dt < \infty \tag{13.2}$$

Using Parseval's theorem, the energy E dissipated by the current or voltage signal $x(t)$ across a 1Ω resistor, over the interval (t_1, t_2), is

$$E = \int_{t_1}^{t_2} |x(t)|^2 \, dt \tag{13.3}$$

The signal $x(t)$ is said to be an energy signal if and only if it has finite energy,

$$E = \int_{-\infty}^{\infty} |x(t)|^2 \, dt \quad < \infty \tag{13.4}$$

A signal $x(t)$ is said to be periodic with period T if and only if

$$x(t) = x(t + nT) \qquad for \ all \ t \tag{13.5}$$

where n is an integer.

Example:

Classify each of the following signals as an energy signal, as a power signal, or as neither. All signals are defined over the interval $(-\infty < t < \infty)$*:* $x_1(t) = \cos t + \cos 2t$, $x_2(t) = \exp(-\alpha^2 t^2)$.

Solution:

$$P_{x_1} = \frac{1}{T} \int_{-T/2}^{T/2} (\cos t + \cos 2t)^2 dt = 1 \Rightarrow \ power \ signal$$

Note that since the cosine function is periodic, the limit is not necessary.

$$E_{x_2} = \int_{-\infty}^{\infty} (e^{-\alpha^2 t^2})^2 dt = 2\int_{0}^{\infty} e^{-2\alpha^2 t^2} dt = 2\frac{\sqrt{\pi}}{2\sqrt{2}\alpha} = \frac{1}{\alpha}\sqrt{\frac{\pi}{2}} \ \Rightarrow \ energy \ signal$$

Electrical systems can be linear or nonlinear. Furthermore, linear systems may be divided into continuous or discrete. A system is linear if the input signal $x_1(t)$ produces $y_1(t)$ and $x_2(t)$ produces $y_2(t)$; then for some arbitrary constants a_1 and a_2 the input signal $a_1x_1(t) + a_2x_2(t)$ produces the output $a_1y_1(t) + a_2y_2(t)$. A linear system is said to be shift invariant (or time invariant) if a time shift at its input produces the same shift at its output. More precisely, if the input signal $x(t)$ produces $y(t)$ then the delayed signal $x(t - t_0)$ produces the output $y(t - t_0)$. The impulse response of a Linear Time Invariant (LTI) system, $h(t)$, is defined to be the system's output when the input is an impulse (delta function).

13.2. The Fourier Transform

The Fourier Transform (FT) of the signal $x(t)$ is

$$F\{x(t)\} = X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\, dt \tag{13.6}$$

or

$$F\{x(t)\} = X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\, dt \tag{13.7}$$

and the Inverse Fourier Transform (IFT) is

$$F^{-1}\{X(\omega)\} = x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}\, d\omega \tag{13.8}$$

or

$$F^{-1}\{X(f)\} = x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}\, df \tag{13.9}$$

where, in general, t represents time, while $\omega = 2\pi f$ and f represent frequency in radians per second and Hertz, respectively. In this book we will use both notations for the transform, as appropriate (i.e., $X(\omega)$ and $X(f)$).

A detailed table of the FT pairs is listed in Appendix 13A. The FT properties are (the proofs are left as an exercise):

1. *Linearity:*

$$F\{a_1x_1(t) + a_2x_2(t)\} = a_1X_1(\omega) + a_2X_2(\omega) \tag{13.10}$$

2. *Symmetry: If* $F\{x(t)\} = X(\omega)$ *then*

$$2\pi X(-\omega) = \int_{-\infty}^{\infty} X(t)e^{-j\omega t}dt \tag{13.11}$$

3. *Shifting: For any real time* t_0

$$F\{x(t \pm t_0)\} = e^{\pm j\omega t_0}X(\omega) \tag{13.12}$$

4. *Scaling: If* $F\{x(t)\} = X(\omega)$ *then*

$$F\{x(at)\} = \frac{1}{|a|}X\left(\frac{\omega}{a}\right) \tag{13.13}$$

5. *Central Ordinate:*

$$X(0) = \int_{-\infty}^{\infty} x(t)dt \tag{13.14}$$

$$x(0) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)d\omega \tag{13.15}$$

6. *Frequency Shift: If* $F\{x(t)\} = X(\omega)$ *then*

$$F\{e^{\pm\omega_0 t}x(t)\} = X(\omega \mp \omega_0) \tag{13.16}$$

7. *Modulation: If* $F\{x(t)\} = X(\omega)$ *then*

$$F\{x(t)\cos\omega_0 t\} = \frac{1}{2}[X(\omega + \omega_0) + X(\omega - \omega_0)] \tag{13.17}$$

$$F\{x(t)\sin(\omega_0 t)\} = \frac{1}{2j}[X(\omega - \omega_0) - X(\omega + \omega_0)] \tag{13.18}$$

8. *Derivatives:*

$$F\left\{\frac{d^n}{dt^n}(x(t))\right\} = (j\omega)^n X(\omega) \tag{13.19}$$

9. *Time Convolution: if* $x(t)$ *and* $h(t)$ *have Fourier transforms* $X(\omega)$ *and* $H(\omega)$*, respectively, then*

$$F\left\{\int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau\right\} = X(\omega)H(\omega) \tag{13.20}$$

10. *Frequency Convolution:*

$$F\{x(t)h(t)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\tau)H(\omega - \tau)d\tau \tag{13.21}$$

11. *Autocorrelation:*

$$F\left\{\int_{-\infty}^{\infty} x(\tau)x^*(\tau - t)d\tau\right\} = X(\omega)X^*(\omega) = |X(\omega)|^2 \tag{13.22}$$

12. *Parseval's Theorem: The energy associated with the signal* $x(t)$ *is*

$$E = \int_{-\infty}^{\infty} |x(t)^2| dt = \int_{-\infty}^{\infty} |X(\omega)|^2 d\omega \tag{13.23}$$

13. *Moments: The* n*th moment is*

$$m_n = \int_0^{\infty} t^n x(t)dt = \frac{d^n}{d\omega^n} X(\omega)\Big|_{\omega = 0} \tag{13.24}$$

13.3. The Fourier Series

A set of functions $S = \{\varphi_n(t) \; ; \; n = 1, \ldots, N\}$ is said to be orthogonal over the interval (t_1, t_2) if and only if

$$\int_{t_1}^{t_2} \varphi_i^*(t)\varphi_j(t)dt = \int_{t_1}^{t_2} \varphi_i(t)\varphi_j^*(t)dt = \begin{cases} 0 & i \neq j \\ \lambda_i & i = j \end{cases} \tag{13.25}$$

where the asterisk indicates complex conjugate, and λ_i are constants. If $\lambda_i = 1$ for all i, then the set S is said to be an orthonormal set.

An electrical signal $x(t)$ can be expressed over the interval (t_1, t_2) as a weighted sum of a set of orthogonal functions as

$$x(t) \approx \sum_{n=1}^{N} X_n \varphi_n(t) \tag{13.26}$$

where X_n are, in general, complex constants, and the orthogonal functions $\varphi_n(t)$ are called basis functions. If the integral-square error over the interval (t_1, t_2) is equal to zero as N approaches infinity, i.e.,

$$\lim_{N \to \infty} \int_{t_1}^{t_2} \left| x(t) - \sum_{n=1}^{N} X_n \varphi_n(t) \right|^2 dt = 0 \tag{13.27}$$

then the set $S = \{\varphi_n(t)\}$ is said to be complete, and Eq. (13.26) becomes an equality. The constants X_n are computed as

$$X_n = \frac{\int_{t_1}^{t_2} x(t)\varphi_n^*(t)dt}{\int_{t_1}^{t_2} |\varphi_n(t)|^2 dt} \tag{13.28}$$

Let the signal $x(t)$ be periodic with period T, and let the complete orthogonal set S be

$$S = \left\{ e^{\frac{j2\pi nt}{T}} \; ; \; n = -\infty, \infty \right\} \tag{13.29}$$

Then the complex exponential Fourier series of $x(t)$ is

$$x(t) = \sum_{n=-\infty}^{\infty} X_n e^{\frac{j2\pi nt}{T}} \tag{13.30}$$

Using Eq. (13.28) yields

$$X_n = \frac{1}{T} \int_{-T/2}^{T/2} x(t) e^{\frac{-j2\pi nt}{T}} dt \tag{13.31}$$

The FT of Eq. (13.30) is given by

$$X(\omega) = 2\pi \sum_{n=-\infty}^{\infty} X_n \delta\left(\omega - \frac{2\pi n}{T}\right) \tag{13.32}$$

where $\delta(\cdot)$ is delta function. When the signal $x(t)$ is real we can compute its trigonometric Fourier series from Eq. (13.30) as

$$x(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos\left(\frac{2\pi nt}{T}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{2\pi nt}{T}\right) \tag{13.33}$$

$$a_0 = X_0$$
$$a_n = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\cos\left(\frac{2\pi nt}{T}\right)dt \qquad (13.34)$$
$$b_n = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\sin\left(\frac{2\pi nt}{T}\right)dt$$

The coefficients a_n are all zeros when the signal $x(t)$ is an odd function of time. Alternatively, when the signal is an even function of time, then all b_n are equal to zero.

Consider the periodic energy signal defined in Eq. (13.33). The total energy associated with this signal is then given by

$$E = \frac{1}{T}\int_{t_0}^{t_0+T} |x(t)|^2 dt = \frac{a_0^2}{4} + \sum_{n=1}^{\infty}\left(\frac{a_n^2}{2} + \frac{b_n^2}{2}\right) \qquad (13.35)$$

13.4. Convolution and Correlation Integrals

The convolution $\phi_{xh}(t)$ between the signals $x(t)$ and $h(t)$ is defined by

$$\phi_{xh}(t) = x(t) \bullet h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \qquad (13.36)$$

where τ is a dummy variable, and the operator $\bullet$ is used to symbolically describe the convolution integral. Convolution is commutative, associative, and distributive. More precisely,

$$x(t) \bullet h(t) = h(t) \bullet x(t)$$
$$x(t) \bullet h(t) \bullet g(t) = (x(t) \bullet h(t)) \bullet g(t) = x(t) \bullet (h(t) \bullet g(t)) \qquad (13.37)$$

For the convolution integral to be finite at least one of the two signals must be an energy signal. The convolution between two signals can be computed using the FT

$$\phi_{xh}(t) = F^{-1}\{X(\omega)H(\omega)\} \qquad (13.38)$$

Consider an LTI system with impulse response $h(t)$ and input signal $x(t)$. It follows that the output signal $y(t)$ is equal to the convolution between the input signal and the system impulse response,

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \quad \textbf{(13.39)}$$

The cross-correlation function between the signals $x(t)$ and $g(t)$ is defined as

$$R_{xg}(t) = \int_{-\infty}^{\infty} x^*(\tau)g(t+\tau)d\tau \quad \textbf{(13.40)}$$

Again, at least one of the two signals should be an energy signal for the correlation integral to be finite. The cross-correlation function measures the similarity between the two signals. The peak value of $R_{xg}(t)$ and its spread around this peak are an indication of how good this similarity is. The cross-correlation integral can be computed as

$$R_{xg}(t) = F^{-1}\{X^*(\omega)G(\omega)\} \quad \textbf{(13.41)}$$

When $x(t) = g(t)$ we get the autocorrelation integral,

$$R_x(t) = \int_{-\infty}^{\infty} x^*(\tau)x(t+\tau)d\tau \quad \textbf{(13.42)}$$

Note that the autocorrelation function is denoted by $R_x(t)$ rather than $R_{xx}(t)$. When the signals $x(t)$ and $g(t)$ are power signals, the correlation integral becomes infinite and, thus, time averaging must be included. More precisely,

$$\bar{R}_{xg}(t) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x^*(\tau)g(t+\tau)d\tau \quad \textbf{(13.43)}$$

13.5. Energy and Power Spectrum Densities

Consider an energy signal $x(t)$. From Parseval's theorem, the total energy associated with this signal is

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2 d\omega \quad \textbf{(13.44)}$$

When $x(t)$ is a voltage signal, the amount of energy dissipated by this signal when applied across a network of resistance R is

$$E = \frac{1}{R}\int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi R}\int_{-\infty}^{\infty} |X(\omega)|^2 d\omega \quad \textbf{(13.45)}$$

Alternatively, when $x(t)$ is a current signal we get

$$E = R\int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{R}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2 d\omega \quad \textbf{(13.46)}$$

The quantity $\int |X(\omega)|^2 d\omega$ represents the amount of energy spread per unit frequency across a 1Ω resistor; therefore, the Energy Spectrum Density (ESD) function for the energy signal $x(t)$ is defined as

$$ESD = |X(\omega)|^2 \quad \textbf{(13.47)}$$

The ESD at the output of an LTI system when $x(t)$ is at its input is

$$|Y(\omega)|^2 = |X(\omega)|^2 |H(\omega)|^2 \quad \textbf{(13.48)}$$

where $H(\omega)$ is the FT of the system impulse response, $h(t)$. It follows that the energy present at the output of the system is

$$E_y = \frac{1}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2 |H(\omega)|^2 d\omega \quad \textbf{(13.49)}$$

Example:

The voltage signal $x(t) = e^{-5t}$; $t \geq 0$ is applied to the input of a low pass LTI system. The system bandwidth is $5Hz$, and its input resistance is 5Ω. If $H(\omega) = 1$ over the interval $(-10\pi < \omega < 10\pi)$ and zero elsewhere, compute the energy at the output.

Solution:

From Eqs. (13.45) and (13.49) we get

$$E_y = \frac{1}{2\pi R}\int_{\omega = -10\pi}^{10\pi} |X(\omega)|^2 |H(\omega)|^2 d\omega$$

Using Fourier transform tables and substituting $R = 5$ yield

$$E_y = \frac{1}{5\pi}\int_{0}^{10\pi} \frac{1}{\omega^2 + 25} d\omega$$

Completing the integration yields

$$E_y = \frac{1}{25\pi}[\text{atanh}(2\pi) - \text{atanh}(0)] = 0.01799 \ \ Joules$$

Note that an infinite bandwidth would give $E_y = 0.02$, *only 11% larger.*

The total power associated with a power signal $g(t)$ is

$$P = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} |g(t)|^2 dt \tag{13.50}$$

Define the Power Spectrum Density (PSD) function for the signal $g(t)$ as $S_g(\omega)$, where

$$P = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} |g(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_g(\omega) d\omega \tag{13.51}$$

It can be shown that (see Problem 1.13)

$$S_g(\omega) = \lim_{T\to\infty} \frac{|G(\omega)|^2}{T} \tag{13.52}$$

Let the signals $x(t)$ and $g(t)$ be two periodic signals with period T. The complex exponential Fourier series expansions for those signals are, respectively, given by

$$x(t) = \sum_{n=-\infty}^{\infty} X_n e^{\frac{j2\pi nt}{T}} \tag{13.53}$$

$$g(t) = \sum_{m=-\infty}^{\infty} G_m e^{\frac{j2\pi mt}{T}} \tag{13.54}$$

The power cross-correlation function $\overline{R}_{gx}(t)$ was given in Eq. (13.43), and is repeated here as Eq. (13.55),

$$\overline{R}_{gx}(t) = \frac{1}{T} \int_{-T/2}^{T/2} g^*(\tau) x(t+\tau) d\tau \tag{13.55}$$

Note that because both signals are periodic the limit is no longer necessary. Substituting Eqs. (13.53) and (13.54) into Eq. (13.55), collecting terms, and using the definition of orthogonality, we get

$$\bar{R}_{gx}(t) = \sum_{n=-\infty}^{\infty} G_n^* X_n e^{\frac{j2n\pi t}{T}} \tag{13.56}$$

When $x(t) = g(t)$, Eq. (13.56) becomes the power autocorrelation function,

$$\bar{R}_x(t) = \sum_{n=-\infty}^{\infty} |X_n|^2 e^{\frac{j2n\pi t}{T}} = |X_0|^2 + 2\sum_{n=1}^{\infty} |X_n|^2 e^{\frac{j2n\pi t}{T}} \tag{13.57}$$

The power spectrum and cross-power spectrum density functions are then computed as the FT of Eqs. (13.57) and (13.56), respectively. More precisely,

$$\begin{aligned} \bar{S}_x(\omega) &= 2\pi \sum_{n=-\infty}^{\infty} |X_n|^2 \delta\left(\omega - \frac{2n\pi}{T}\right) \\ \bar{S}_{gx}(\omega) &= 2\pi \sum_{n=-\infty}^{\infty} G_n^* X_n \delta\left(\omega - \frac{2n\pi}{T}\right) \end{aligned} \tag{13.58}$$

The line (or discrete) power spectrum is defined as the plot of $|X_n|^2$ versus n, where the lines are $\Delta f = 1/T$ apart. The DC power is $|X_0|^2$, and the total power is $\sum_{n=-\infty}^{\infty} |X_n|^2$.

13.6. 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 "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 or equal to the value x. More precisely,

$$F_X(x) = Pr\{X \le x\} \tag{13.59}$$

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

$$F_X(x_2) - F_X(x_1) = Pr\{x_1 \le X \le x_2\} \tag{13.60}$$

The *cdf* has the following properties:

$$\begin{aligned} 0 \le F_X(x) \le 1 \\ F_X(-\infty) = 0 \\ F_X(\infty) = 1 \\ F_X(x_1) \le F_X(x_2) \Leftrightarrow x_1 \le x_2 \end{aligned} \tag{13.61}$$

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*

$$f_X(x) = \frac{d}{dx} F_X(x) \tag{13.62}$$

or, equivalently,

$$F_X(x) = Pr\{X \le x\} = \int_{-\infty}^{x} f_X(\lambda) d\lambda \tag{13.63}$$

The probability that a random variable X has values in the interval (x_1, x_2) is

$$F_X(x_2) - F_X(x_1) = Pr\{x_1 \le X \le x_2\} = \int_{x_1}^{x_2} f_X(x) dx \tag{13.64}$$

Define the nth moment for the random variable X as

$$E[X^n] = \overline{X^n} = \int_{-\infty}^{\infty} x^n f_X(x) dx \tag{13.65}$$

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

$$E[(X - \bar{X})^n] = \overline{(X - \bar{X})^n} = \int_{-\infty}^{\infty} (x - \bar{x})^n f_X(x) dx \tag{13.66}$$

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,

$$\sigma_X^2 = \overline{(X - \bar{X})^2} \tag{13.67}$$

Appendix 13B has some common *pdf*s and their means and variances.

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

$$F_{XY}(x, y) = Pr\{X \le x; Y \le y\} \tag{13.68}$$

$$f_{XY}(x, y) = \frac{\partial^2}{\partial x \partial y} F_{XY}(x, y) \tag{13.69}$$

The marginal *cdf*s are obtained as follows:

$$\begin{aligned} F_X(x) &= \int_{-\infty}^{\infty} \int_{-\infty}^{x} f_{UV}(u, v)\,du\,dv = F_{XY}(x, \infty) \\ F_Y(y) &= \int_{-\infty}^{\infty} \int_{-\infty}^{y} f_{UV}(u, v)\,dv\,du = F_{XY}(\infty, y) \end{aligned} \tag{13.70}$$

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

$$F_{XY}(x, y) = F_X(x) F_Y(y) \tag{13.71}$$

$$f_{XY}(x, y) = f_X(x) f_Y(y) \tag{13.72}$$

Let us now 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

$$\begin{aligned} U &= T_1(X, Y) \\ V &= T_2(X, Y) \end{aligned} \tag{13.73}$$

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

$$f_{UV}(u, v) = f_{XY}(x, y)|J| \tag{13.74}$$

$$|J| = \begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{vmatrix} \tag{13.75}$$

where the determinant of the matrix of derivatives $|J|$ is called the Jacobian.

The characteristic function for the random variable X is defined as

$$C_X(\omega) = E[e^{j\omega X}] = \int_{-\infty}^{\infty} f_X(x) e^{j\omega x} dx \tag{13.76}$$

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 equal to

$$Y = X_1 + X_2 + \ldots + X_N \tag{13.77}$$

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

$$C_Y(\omega) = C_{X_1}(\omega) C_{X_2}(\omega) \ldots C_{X_N}(\omega) \tag{13.78}$$

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

$$f_Y(y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} C_Y(\omega) e^{-j\omega y} d\omega \tag{13.79}$$

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

$$E[X^n] = (-j)^n \frac{d^n}{d\omega^n} C_X(\omega) \Big|_{\omega = 0} \tag{13.80}$$

13.7. Multivariate Gaussian Distribution

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, $\underline{X}$. More precisely,

$$\underline{X}^t = \left[X_1 \; X_2 \; \ldots \; X_m\right] \tag{13.81}$$

where the superscript indicates the transpose operation. The joint *pdf* for the vector $\underline{X}$ is

$$f_{\underline{x}}(\underline{x}) = f_{x_1, x_2, \ldots, x_m}(x_1, x_2, \ldots, x_m) \tag{13.82}$$

The mean vector is defined as

$$\mu_x = \left[E[X_1] \; E[X_2] \; \ldots \; E[X_m]\right]^t \tag{13.83}$$

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

$$C_x = E[\underline{X} \; \underline{X}^t] - \mu_x \; \mu_x^t \tag{13.84}$$

Note that if the elements of the vector $\underline{X}$ are independent, then the covariance matrix is a diagonal matrix.

By definition a random vector $\underline{X}$ is multivariate Gaussian if its *pdf* has the form

$$f_{\underline{x}}(\underline{x}) = [(2\pi)^{m/2}|C_x|^{1/2}]^{-1} \exp\left(-\frac{1}{2}(\underline{x} - \mu_x)^t C_x^{-1}(\underline{x} - \mu_x)\right) \tag{13.85}$$

where μ_x is the mean vector, C_x is the covariance matrix, C_x^{-1} is inverse of the covariance matrix and $|C_x|$ is its determinant, and $\underline{X}$ is of dimension m. If $\underline{A}$ is a $k \times m$ matrix of rank k, then the random vector $\underline{Y} = \underline{A}\underline{X}$ is a k-variate Gaussian vector with

$$\mu_y = \underline{A}\mu_x \tag{13.86}$$

$$C_y = \underline{A} C_x \underline{A}^t \tag{13.87}$$

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

$$C_{\underline{X}} = E[\exp\{j(\omega_1 X_1 + \omega_2 X_2 + \ldots + \omega_m X_m)\}] = \tag{13.88}$$

$$\exp\left\{j\mu_x^t \underline{\omega} - \frac{1}{2}\underline{\omega}^t C_x \underline{\omega}\right\}$$

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

$$E[X_1X_2X_3] = \frac{\partial^3}{\partial\omega_1\partial\omega_2\partial\omega_3}C_{\underline{X}}(\omega_1, \omega_2, \omega_3) \qquad at \quad \underline{\omega} = \underline{0} \qquad \textbf{(13.89)}$$

Example:

The vector $\underline{X}$ is a 4-variate Gaussian with

$$\mu_x = \begin{bmatrix} 2 & 1 & 1 & 0 \end{bmatrix}^t$$

$$C_x = \begin{bmatrix} 6 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 3 \end{bmatrix}$$

Define

$$\underline{X}_1 = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \qquad \underline{X}_2 = \begin{bmatrix} X_3 \\ X_4 \end{bmatrix}$$

Find the distribution of $\underline{X}_1$ and the distribution of

$$\underline{Y} = \begin{bmatrix} 2X_1 \\ X_1 + 2X_2 \\ X_3 + X_4 \end{bmatrix}$$

Solution:

$\underline{X}_1$ has a bivariate Gaussian distribution with

$$\mu_{x_1} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \qquad C_{x_1} = \begin{bmatrix} 6 & 3 \\ 3 & 4 \end{bmatrix}$$

The vector $\underline{Y}$ can be expressed as

$$\underline{Y} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \underline{A}\underline{X}$$

It follows that

$$\mu_y = \underline{A}\mu_x = \begin{bmatrix} 4 & 4 & 1 \end{bmatrix}^t$$

$$C_y = \underline{A}C_x\underline{A}^t = \begin{bmatrix} 24 & 24 & 6 \\ 24 & 34 & 13 \\ 6 & 13 & 13 \end{bmatrix}$$

13.8. 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 outcomes of the experiment as well as 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 to 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

$$E[X^n(t)] = \int_{-\infty}^{\infty} x^n f_X(x;t)dx \qquad \textbf{(13.90)}$$

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

$$f_X(x_1, x_2;t_1, t_2) = f_X(x_1, x_2;t_1 + \Delta t, t_2 + \Delta t) \qquad \textbf{(13.91)}$$

for all t_1, t_2 and Δt.

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

$$\Re_X(t_1, t_2) = E[X(t_1)X(t_2)] \qquad \textbf{(13.92)}$$

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

$$E[X(t)] = \bar{X}$$
$$\Re_X(\tau) = E[X(t)X(t+\tau)] \tag{13.93}$$

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:

$$\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t)dt = E[X(t)] = \bar{X} \tag{13.94}$$

$$\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x^*(t)x(t+\tau)dt = \Re_X(\tau) \tag{13.95}$$

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

$$C_{XY}(t, t+\tau) = E[\{X(t) - E[X(t)]\}\{Y(t+\tau) - E[Y(t+\tau)]\}] \tag{13.96}$$

which can be written as

$$C_{XY}(t, t+\tau) = \Re_{XY}(\tau) - \bar{X}\bar{Y} \tag{13.97}$$

13.9. Sampling Theorem

Most modern communication and radar systems are designed to process discrete samples of signals bearing information. In general, we would like to determine the necessary condition such that a signal can be fully reconstructed from its samples by filtering, or data processing in general. The answer to this question lies in the sampling theorem which may be stated as follows: let the signal $x(t)$ be real-valued and band-limited with bandwidth B; this signal can be fully reconstructed from its samples if the time interval between samples is no greater than $1/(2B)$.

Fig. 13.1 illustrates the sampling process concept. The sampling signal $p(t)$ is periodic with period T_s, which is called the sampling interval. The Fourier series expansion of $p(t)$ is

$$p(t) = \sum_{n=-\infty}^{\infty} P_n e^{\frac{j2\pi nt}{T_s}} \tag{13.98}$$

The sampled signal $x_s(t)$ is then given by

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(t) P_n e^{\frac{j2\pi nt}{T_s}} \tag{13.99}$$

Taking the FT of Eq. (13.99) yields

$$X_s(\omega) = \sum_{n=-\infty}^{\infty} P_n \, X\left(\omega - \frac{2\pi n}{T_s}\right) = P_0 X(\omega) + \sum_{\substack{n=-\infty \\ n \neq 0}}^{\infty} P_n \, X\left(\omega - \frac{2\pi n}{T_s}\right) \tag{13.100}$$

where $X(\omega)$ is the FT of $x(t)$. Therefore, we conclude that the spectral density, $X_s(\omega)$, consists of replicas of $X(\omega)$ spaced $(2\pi / T_s)$ apart and scaled by the Fourier series coefficients P_n. A Low Pass Filter (LPF) of bandwidth B can then be used to recover the original signal $x(t)$.

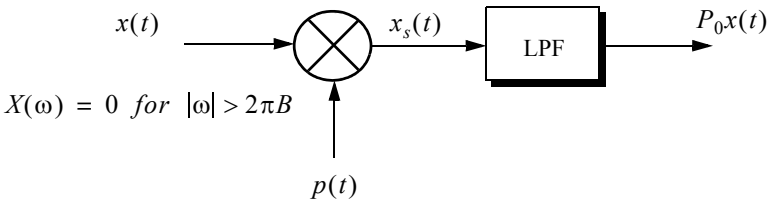

Figure 13.1. Concept of sampling.

When the sampling rate is increased (i.e., T_s decreases), the replicas of $X(\omega)$ move farther apart from each other. Alternatively, when the sampling rate is decreased (i.e., T_s increases), the replicas get closer to one another. The value of T_s such that the replicas are tangent to one another defines the minimum required sampling rate so that $x(t)$ can be recovered from its samples by using an LPF. It follows that

$$\frac{2\pi}{T_s} = 2\pi(2B) \Leftrightarrow T_s = \frac{1}{2B} \tag{13.101}$$

The sampling rate defined by Eq. (13.101) is known as the Nyquist sampling rate. When $T_s > (1/2B)$, the replicas of $X(\omega)$ overlap and, thus, $x(t)$ cannot be recovered cleanly from its samples. This is known as aliasing. In practice, ideal LPF cannot be implemented; hence, practical systems tend to over-sample in order to avoid aliasing.

Example:

Assume that the sampling signal $p(t)$ is given by

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s)$$

Compute an expression for $X_s(\omega)$.

Solution:

The signal $p(t)$ is called the Comb function. Its exponential Fourier series is

$$p(t) = \sum_{n=-\infty}^{\infty} \frac{1}{T_s} e^{\frac{2\pi n t}{T_s}}$$

It follows that

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(t) \frac{1}{T_s} e^{\frac{2\pi n t}{T_s}}$$

Taking the Fourier transform of this equation yields

$$X_s(\omega) = \frac{2\pi}{T_s} \sum_{n=-\infty}^{\infty} X\left(\omega - \frac{2\pi n}{T_s}\right).$$

Before proceeding to the next section, we will establish the following notation: samples of the signal $x(t)$ are denoted by $x(n)$ and referred to as a discrete time domain sequence, or simply a sequence. If the signal $x(t)$ is periodic, we will denote its sample by the periodic sequence $\tilde{x}(n)$.

13.10. The Z-Transform

The Z-transform is a transformation that maps samples of a discrete time domain sequence into a new domain known as the z-domain. It is defined as

$$Z\{x(n)\} = X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n} \qquad \textbf{(13.102)}$$

where $z = re^{j\omega}$, and for most cases, $r = 1$. It follows that Eq. (13.102) can be rewritten as

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-jn\omega} \tag{13.103}$$

In the z-domain, the region over which $X(z)$ is finite is called the Region of Convergence (ROC). Appendix 13C has a list of most common Z-transform pairs. The Z-transform properties are (the proofs are left as an exercise):

1. *Linearity:*

$$Z\{ax_1(n) + bx_2(n)\} = aX_1(z) + bX_2(z) \tag{13.104}$$

2. *Right-Shifting Property:*

$$Z\{x(n-k)\} = z^{-k}X(z) \tag{13.105}$$

3. *Left-Shifting Property:*

$$Z\{x(n+k)\} = z^{k}X(z) - \sum_{n=0}^{k-1} x(n)z^{k-n} \tag{13.106}$$

4. *Time Scaling:*

$$Z\{a^{n}x(n)\} = X(a^{-1}z) = \sum_{n=0}^{\infty} (a^{-1}z)^{-n}x(n) \tag{13.107}$$

5. *Periodic Sequences:*

$$Z\{x(n)\} = \frac{z^{N}}{z^{N}-1}Z\{x(n)\} \tag{13.108}$$

where N is the period.

6. *Multiplication by n :*

$$Z\{nx(n)\} = -z\frac{d}{dz}X(z) \tag{13.109}$$

7. *Division by n + a ; a is a real number:*

$$Z\left\{\frac{x(n)}{n+a}\right\} = \sum_{n=0}^{\infty} x(n) z^{a}\left(-\int_{0}^{z} u^{-k-a-1} du\right) \tag{13.110}$$

8. *Initial Value:*

$$x(n_0) = z^{n_0} X(z)\Big|_{z \to \infty} \tag{13.111}$$

9. *Final Value:*

$$\lim_{n \to \infty} x(n) = \lim_{z \to 1} (1 - z^{-1}) X(z) \tag{13.112}$$

10. *Convolution:*

$$Z\left\{\sum_{k=0}^{\infty} h(n-k)x(k)\right\} = H(z)X(z) \tag{13.113}$$

11. *Bilateral Convolution:*

$$Z\left\{\sum_{k=-\infty}^{\infty} h(n-k)x(k)\right\} = H(z)X(z) \tag{13.114}$$

Example:

Prove Eq. (13.109).

Solution:

Starting with the definition of the Z-transform,

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$

Taking the derivative, with respect to z, of the above equation yields

$$\frac{d}{dz} X(z) = \sum_{n=-\infty}^{\infty} x(n)(-n) z^{-n-1}$$

$$= (-z^{-1}) \sum_{n=-\infty}^{\infty} nx(n)z^{-n}$$

It follows that

$$Z\{nx(n)\} = (-z)\frac{d}{dz}X(z)$$

In general, a discrete LTI system has a transfer function $H(z)$ which describes how the system operates on its input sequence $x(n)$ in order to produce the output sequence $y(n)$. The output sequence $y(n)$ is computed from the discrete convolution between the sequences $x(n)$ and $h(n)$,

$$y(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m) \tag{13.115}$$

However, since practical systems require that the sequence $x(n)$ be of finite length, we can rewrite Eq. (13.115) as

$$y(n) = \sum_{m=0}^{N} x(m)h(n-m) \tag{13.116}$$

where N denotes the input sequence length. Taking the Z-transform of Eq. (13.116) yields

$$Y(z) = X(z)H(z) \tag{13.117}$$

and the discrete system transfer function is

$$H(z) = \frac{Y(z)}{X(z)} \tag{13.118}$$

Finally, the transfer function $H(z)$ can be written as

$$H(z)\Big|_{z=e^{j\omega}} = \left|H(e^{j\omega})\right| e^{\angle H(e^{j\omega})} \tag{13.119}$$

where $\left|H(e^{j\omega})\right|$ is the amplitude response, and $\angle H(e^{j\omega})$ is the phase response.

13.11. The Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is a mathematical operation that transforms a discrete sequence, usually from the time domain into the frequency domain, in order to explicitly determine the spectral information for the sequence. The time domain sequence can be real or complex. The DFT has finite length N, and is periodic with period equal to N.

The discrete Fourier transform for the finite sequence $x(n)$ is defined by

$$\tilde{X}(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi nk}{N}} \quad ; \; k = 0, \ldots, N-1 \qquad \textbf{(13.120)}$$

The inverse DFT is given by

$$\tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{\frac{j2\pi nk}{N}} \quad ; \; n = 0, \ldots, N-1 \qquad \textbf{(13.121)}$$

The Fast Fourier Transform (FFT) is not a new kind of transform different from the DFT. Instead, it is an algorithm used to compute the DFT more efficiently. There are numerous FFT algorithms that can be found in the literature. In this book we will interchangeably use the DFT and the FFT to mean the same thing. Furthermore, we will assume radix-2 FFT algorithm, where the FFT size is equal to $N = 2^m$ for some integer m.

13.12. Discrete Power Spectrum

Practical discrete systems utilize DFTs of finite length as a means of numerical approximation for the Fourier transform. It follows that input signals must be truncated to a finite duration (denoted by T) before they are sampled. This is necessary so that a finite length sequence is generated prior to signal processing. Unfortunately, this truncation process may cause some serious problems.

To demonstrate this difficulty, consider the time domain signal $x(t) = \sin 2\pi f_0 t$. The spectrum of $x(t)$ consists of two spectral lines at $\pm f_0$. Now, when $x(t)$ is truncated to length T seconds and sampled at a rate $T_s = T/N$, where N is the number of desired samples, we produce the sequence $\{x(n) \; ; \; n = 0, 1, \ldots, N-1\}$. The spectrum of $x(n)$ would still be composed of the same spectral lines if T is an integer multiple of T_s and if the DFT frequency resolution Δf is an integer multiple of f_0. Unfortunately, those two conditions are rarely met and, as a consequence, the spectrum of $x(n)$

spreads over several lines (normally the spread may extend up to three lines). This is known as spectral leakage. Since f_0 is normally unknown, this discontinuity caused by an arbitrary choice of T cannot be avoided. Windowing techniques can be used to mitigate the effect of this discontinuity by applying smaller weights to samples close to the edges.

A truncated sequence $x(n)$ can be viewed as one period of some periodic sequence $\tilde{x}(n)$ with period N. The discrete Fourier series expansion of $x(n)$ is

$$x(n) = \sum_{k=0}^{N-1} X_k e^{\frac{j2\pi nk}{N}} \tag{13.122}$$

It can be shown that the coefficients X_k are given by

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}} = \frac{1}{N}X(k) \tag{13.123}$$

where $X(k)$ is the DFT of $x(n)$. Therefore, the Discrete Power Spectrum (DPS) for the band limited sequence $x(n)$ is the plot of $|X_k|^2$ versus k, where the lines are Δf apart,

$$\begin{aligned} P_0 &= \frac{1}{N^2}|X(0)|^2 \\ P_k &= \frac{1}{N^2}\{|X(k)|^2 + |X(N-k)|^2\} \quad ; \; k = 1, 2, \ldots, \frac{N}{2} - 1 \\ P_{N/2} &= \frac{1}{N^2}|X(N/2)|^2 \end{aligned} \tag{13.124}$$

Before proceeding to the next section, we will show how to select the FFT parameters. For this purpose, consider a band limited signal $x(t)$ with bandwidth B. If the signal is not band limited, a LPF can be used to eliminate frequencies greater than B. In order to satisfy the sampling theorem, one must choose a sampling frequency $f_s = 1/T_s$, such that

$$f_s \geq 2B \tag{13.125}$$

The truncated sequence duration T and the total number of samples N are related by

$$T = NT_s \tag{13.126}$$

or equivalently,

$$f_s = \frac{N}{T} \tag{13.127}$$

It follows that

$$f_s = \frac{N}{T} \geq 2B \tag{13.128}$$

and the frequency resolution is

$$\Delta f = \frac{1}{NT_s} = \frac{f_s}{N} = \frac{1}{T} \geq \frac{2B}{N} \tag{13.129}$$

13.13. Windowing Techniques

Truncation of the sequence $x(n)$ can be accomplished by computing the product,

$$x_w(n) = x(n)w(n) \tag{13.130}$$

where

$$w(n) = \left\{ \begin{array}{ll} f(n) & ;\ n = 0, 1, \ldots, N-1 \\ 0 & otherwise \end{array} \right\} \tag{13.131}$$

where $f(n) \leq 1$. The finite sequence $w(n)$ is called a windowing sequence, or simply a window. The windowing process should not impact the phase response of the truncated sequence. Consequently, the sequence $w(n)$ must retain linear phase. This can be accomplished by making the window symmetrical with respect to its central point.

If $f(n) = 1$ for all n we have what is known as the rectangular window. It leads to the Gibbs phenomenon which manifests itself as an overshoot and a ripple before and after a discontinuity. Fig. 13.2 shows the amplitude spectrum of a rectangular window. Note that the first side lobe is at $-13.46dB$ below the main lobe. Windows that place smaller weights on the samples near the edges will have lesser overshoot at the discontinuity points (lower side lobes); hence, they are more desirable than a rectangular window. However, sidelobes reduction is offset by a widening of the main lobe. Therefore, the proper choice of a windowing sequence is continuous trade-off between side lobe reduction and

main lobe widening. Table 13.1 gives a summary of some windows with the corresponding impact on main beam widening and peak reduction.

TABLE 13.1. Common windows.

Window	Null-to-null Beamwidth. Rectangular window is the reference.	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*

The multiplication process defined in Eq. (13.131) is equivalent to cyclic convolution in the frequency domain. It follows that $X_w(k)$ is a smeared (distorted) version of $X(k)$. To minimize this distortion, we would seek windows that have a narrow main lobe and small side lobes. Additionally, using a window other than a rectangular window reduces the power by a factor P_w, where

$$P_w = \frac{1}{N}\sum_{n=0}^{N-1} w^2(n) = \sum_{k=0}^{N-1} |W(k)|^2 \qquad \textbf{(13.132)}$$

It follows that the DPS for the sequence $x_w(n)$ is now given by

$$P_0^w = \frac{1}{P_w N^2}|X(0)|^2$$

$$P_k^w = \frac{1}{P_w N^2}\{|X(k)|^2 + |X(N-k)|^2\} \qquad ; \; k = 1, 2, \ldots, \frac{N}{2} - 1 \qquad \textbf{(13.133)}$$

$$P_{N/2}^w = \frac{1}{P_w N^2}|X(N/2)|^2$$

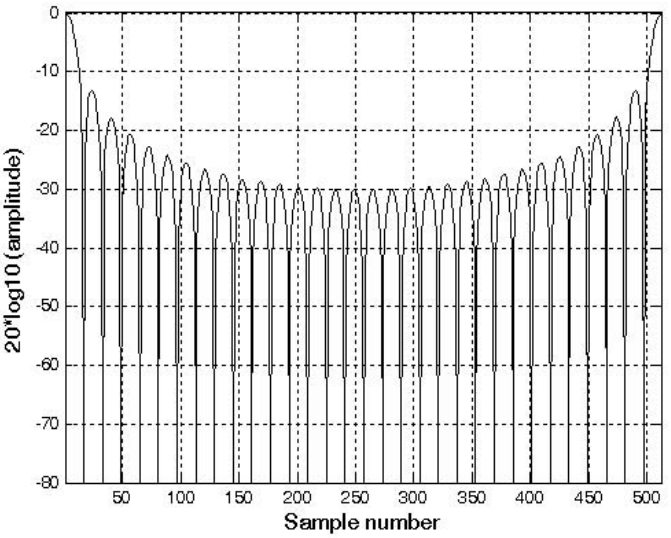

Figure 13.2. Normalized amplitude spectrum for rectangular window.

where P_w is defined in Eq. (13.132). Table 13.2 lists some common windows. Figs. 13.3 through 13.5 show the frequency domain characteristics for these windows. These figures can be reproduced using MATLAB program *"figs13.m"*.

TABLE 13.2. Some common windows. $n = 0, N-1$.

Window	Expression	First side lobe	Main lobe width
rectangular	$w(n) = 1$	$-13.46dB$	1
Hamming	$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$	$-41dB$	2
Hanning	$w(n) = 0.5\left[1 - \cos\left(\frac{2\pi n}{N-1}\right)\right]$	$-32dB$	2
Kaiser	$w(n) = \frac{I_0[\beta\sqrt{1-(2n/N)^2}]}{I_0(\beta)}$ I_0 is the zero-order modified Bessel function of the first kind	$-46dB$ *for* $\beta = 2\pi$	$\sqrt{5}$ *for* $\beta = 2\pi$

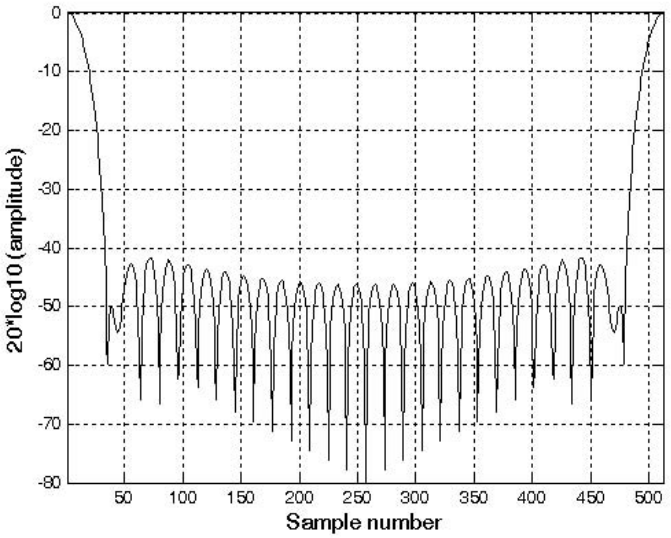

Figure 13.3. Normalized amplitude spectrum for Hamming window.

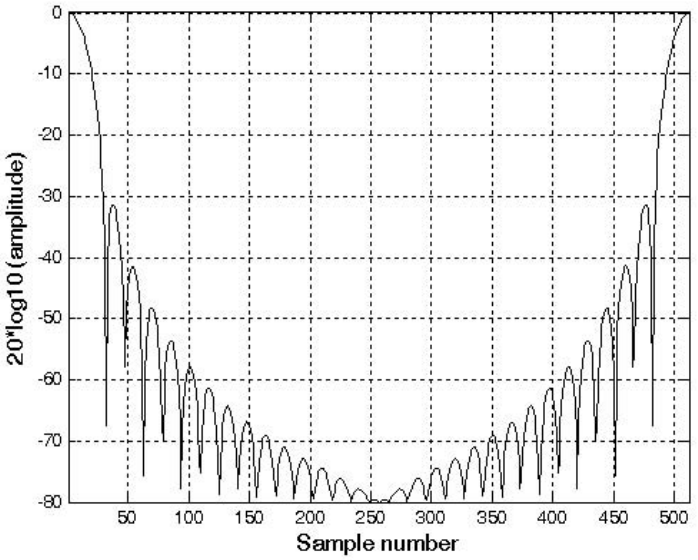

Figure 13.4. Normalized amplitude spectrum for Hanning window.

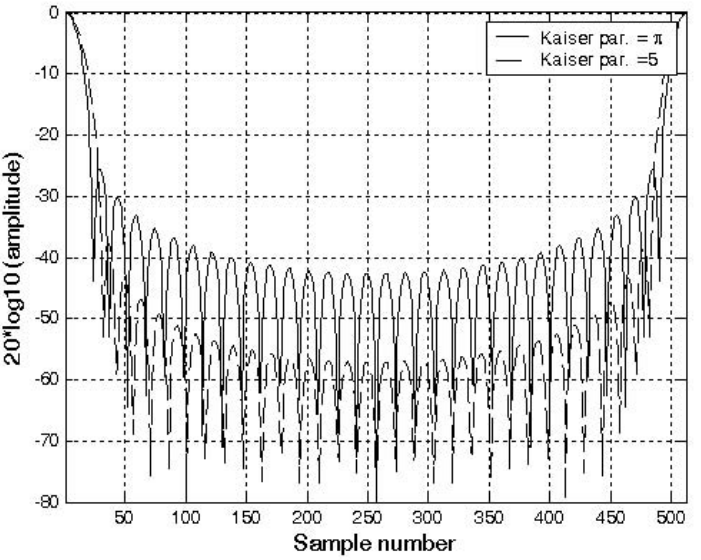

Figure 13.5. Normalized amplitude spectrum for Kaiser window.

13.14. MATLAB Programs

Listing 13.1. MATLAB Program "figs13.m"

```
%Use this program to reproduce figures in Section 13.13.
clear all
close all
eps = 0.0001;
N = 32;
win_rect (1:N) = 1;
win_ham = hamming(N);
win_han = hanning(N);
win_kaiser = kaiser(N, pi);
win_kaiser2 = kaiser(N, 5);
Yrect = abs(fft(win_rect, 512));
Yrectn = Yrect ./ max(Yrect);
Yham = abs(fft(win_ham, 512));
Yhamn = Yham ./ max(Yham);
Yhan = abs(fft(win_han, 512));
Yhann = Yhan ./ max(Yhan);
```

```
YK = abs(fft(win_kaiser, 512));
YKn = YK ./ max(YK);
YK2 = abs(fft(win_kaiser2, 512));
YKn2 = YK2 ./ max(YK2);
figure (1)
plot(20*log10(Yrectn+eps),'k')
xlabel('Sample number')
ylabel('20*log10(amplitude)')
axis tight
grid
figure(2)
plot(20*log10(Yhamn + eps),'k')
xlabel('Sample number')
ylabel('20*log10(amplitude)')
grid
axis tight
figure (3)
plot(20*log10(Yhann+eps),'k')
xlabel('Sample number')
ylabel('20*log10(amplitude)')
grid
axis tight
figure(4)
plot(20*log10(YKn+eps),'k')
grid
hold on
plot(20*log10(YKn2+eps),'k--')
xlabel('Sample number')
ylabel('20*log10(amplitude)')
legend('Kaiser par. = \pi','Kaiser par. =5')
axis tight
hold off
```

Appendix 13A *Fourier Transform Table*

$x(t)$	$X(\omega)$
$ARect(t/\tau)$; rectangular pulse	$A\tau Sinc(\omega\tau/2)$
$A\Delta(t/\tau)$; triangular pulse	$A\frac{\tau}{2}Sinc^2(\tau\omega/4)$
$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{t^2}{2\sigma^2}\right)$; Gaussian pulse	$\exp\left(-\frac{\sigma^2\omega^2}{2}\right)$
$e^{-at}u(t)$	$1/(a+j\omega)$
$e^{-a\|t\|}$	$\frac{2a}{a^2+\omega^2}$
$e^{-at}\sin\omega_0 t \;\; u(t)$	$\frac{\omega_0}{\omega_0^2+(a+j\omega)^2}$
$e^{-at}\cos\omega_0 t \;\; u(t)$	$\frac{a+j\omega}{\omega_0^2+(a+j\omega)^2}$
$\delta(t)$	1
1	$2\pi\delta(\omega)$
$u(t)$	$\pi\delta(\omega)+\frac{1}{j\omega}$
$\mathrm{sgn}(t)$	$\frac{2}{j\omega}$

$x(t)$	$X(\omega)$
$\cos\omega_0 t$	$\pi[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)]$
$\sin\omega_0 t$	$j\pi[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)]$
$u(t)\cos\omega_0 t$	$\frac{\pi}{2}[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)]+\frac{j\omega}{\omega_0^2-\omega^2}$
$u(t)\sin\omega_0 t$	$\frac{\pi}{2j}[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)]+\frac{\omega_0}{\omega_0^2-\omega^2}$
$\lvert t\rvert$	$\frac{-2}{\omega^2}$

Appendix 13B

Some Common Probability Densities

Chi-Square with N degrees of freedom

$$f_X(x) = \frac{x^{(N/2)-1}}{2^{N/2}\Gamma(N/2)} \exp\left\{\frac{-x}{2}\right\} \;;\; x > 0$$

$$\bar{X} = N \;;\; \sigma_X^2 = 2N$$

$$gamma\ function = \Gamma(z) = \int_0^\infty \lambda^{z-1} e^{-\lambda} d\lambda \;;\; Re\{z\} > 0$$

Exponential

$$(f_X(x) = a \exp\{-ax\}) \;;\; x > 0$$

$$\bar{X} = \frac{1}{a} \;;\; \sigma_X^2 = \frac{1}{a^2}$$

Gaussian

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\left(\frac{x - x_m}{\sigma}\right)^2\right\}; \bar{X} = x_m \;;\; \sigma_X^2 = \sigma^2$$

Laplace

$$f_X(x) = \frac{\sigma}{2} \exp\{-\sigma|x - x_m|\}$$

$$\bar{X} = x_m \;\; ; \;\; \sigma_X^2 = \frac{2}{\sigma^2}$$

Log-Normal

$$f_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(- \frac{(\ln x - \ln x_m)^2}{2\sigma^2} \right) \;\; ; \;\; x > 0$$

$$\bar{X} = \exp\left\{ \ln x_m + \frac{\sigma^2}{2} \right\} \;\; ; \;\; \sigma_X^2 = [\exp\{2\ln x_m + \sigma^2\}][\exp\{\sigma^2\} - 1]$$

Rayleigh

$$f_X(x) = \frac{x}{\sigma^2} \exp\left\{ \frac{-x^2}{2\sigma^2} \right\} \;\; ; \;\; x \geq 0$$

$$\bar{X} = \sqrt{\frac{\pi}{2}}\sigma \;\; ; \;\; \sigma_X^2 = \frac{\sigma^2}{2}(4 - \pi)$$

Uniform

$$f_X(x) = \frac{1}{b - a} \;\; ; \;\; a < b \quad ; \quad \bar{X} = \frac{a + b}{2} \;\; ; \;\; \sigma_X^2 = \frac{(b - a)^2}{12}$$

Weibull

$$f_X(x) = \frac{b x^{b-1}}{\bar{\sigma}_0} \exp\left(-\frac{(x)^b}{\bar{\sigma}_0} \right) \;\; ; \;\; (x, b, \bar{\sigma}_0) \geq 0$$

$$\bar{X} = \frac{\Gamma(1 + b^{-1})}{1 / (\sqrt[b]{\bar{\sigma}_0})} \;\; ; \;\; \sigma_X^2 = \frac{\Gamma(1 + 2b^{-1}) - [\Gamma(1 + b^{-1})]^2}{1 / [\sqrt[b]{(\bar{\sigma}_0)^2}]}$$

Appendix 13C *Z - Transform Table*

$x(n);\ n \geq 0$	$X(z)$	**ROC;** $\lvert z \rvert > R$
$\delta(n)$	1	0
1	$\dfrac{z}{z-1}$	1
n	$\dfrac{z}{(z-1)^2}$	1
n^2	$\dfrac{z(z+1)}{(z-1)^3}$	1
a^n	$\dfrac{z}{z-a}$	$\lvert a \rvert$
na^n	$\dfrac{az}{(z-a)^2}$	$\lvert a \rvert$
$\dfrac{a^n}{n!}$	$e^{a/z}$	0
$(n+1)a^n$	$\dfrac{z^2}{(z-a)^2}$	$\lvert a \rvert$
$\sin n\omega T$	$\dfrac{z\sin\omega T}{z^2 - 2z\cos\omega T + 1}$	1
$\cos n\omega T$	$\dfrac{z(z-\cos\omega T)}{z^2 - 2z\cos\omega T + 1}$	1

$x(n);\ n \geq 0$	$X(z)$	**ROC;** $\lvert z \rvert > R$
$a^n \sin n\omega T$	$\dfrac{az \sin \omega T}{z^2 - 2az \cos \omega T + a^2}$	$\dfrac{1}{\lvert a \rvert}$
$a^n \cos n\omega T$	$\dfrac{z(z - a^2 \cos \omega T)}{z^2 - 2az \cos \omega T + a^2}$	$\dfrac{1}{\lvert a \rvert}$
$\dfrac{n(n-1)}{2!}$	$\dfrac{z}{(z-1)^3}$	1
$\dfrac{n(n-1)(n-2)}{3!}$	$\dfrac{z}{(z-1)^4}$	1
$\dfrac{(n+1)(n+2)a^n}{2!}$	$\dfrac{z^3}{(z-a)^3}$	$\lvert a \rvert$
$\dfrac{(n+1)(n+2)\ldots(n+m)a^n}{m!}$	$\dfrac{z^{m+1}}{(z-a)^{m+1}}$	$\lvert a \rvert$

Chapter 14

MATLAB Program and Function Name List

This chapter provides a summary of all MATLAB program and function names used throughout this book. All these programs and functions can be downloaded from the CRC Press Web site (*www.crcpress.com*). For this purpose, follow this procedure: 1) from your Web browser type *"http://www.crcpress.com"*, 2) click on *"Electronic Products"*, 3) click on *"Download & Updates"*, and finally 4) follow instructions of how to download a certain set of code off that Web page. Furthermore, this MATLAB code can also be downloaded from The MathWorks Web site by following these steps: 1) from the Web browser type: *"http://mathworks.com/matlabcentral/fileexchange/"*, 2) place the curser on *"Companion Software for Books"* and click on *"Communications"*.

Chapter 1: Introduction to Radar Basics

Name	Purpose
radar_eq	*Implements radar equation*
fig1_12	*Reproduces Fig. 1.12*
fig1_13	*Reproduces Fig. 1.13*
ref_snr	*Calculates the radar reference range or SNR*
power_aperture	*Implements the power aperture radar equation*
fig1_16	*Reproduces Fig. 1.16*
casestudy1_1	*Program for mini design case study 1.1*
fig1_19	*Reproduces Fig. 1.19*
fig1_21	*Reproduces Fig. 1.21*
pulse_integration	*Performs coherent or non-coherent pulse integration*

Name	Purpose
myradarvisit1_1	*Program for "MyRadar" design case study - visit 1*
fig1_27	*Reproduces Fig. 1.27*
fig1_28	*Reproduces Fig. 1.128*

Chapter 2: Radar Detection

Name	Purpose (all functions have associated GUI)
fig2_2	*Reproduces Fig. 2.2*
que_func	*Implements Marcum's Q-function*
fig2_3	*Reproduces Fig. 2.3*
prob_snr1	*Calculates single pulse probability of detection*
fig2_6a	*Reproduces Fig. 2.6a*
improv_fac	*Calculates the improvement factor*
fig2_6b	*Reproduces Fig. 2.6b*
incomplete_gamma	*Calculates the incomplete Gamma function*
factor	*Calculates the factorial of an integer*
fig2_7	*Reproduces Fig. 2.7*
threshold	*Calculates the detection threshold value*
fig2_8	*Reproduces Fig. 2.8*
pd_swerling5	*Calculates the Swerling 0 or 5 Prob. of detection*
fig2_9	*Reproduces Fig. 2.9*
pd_swrling1	*Calculates the Swerling 1 Prob. of detection*
fig2_10	*Reproduces Fig. 2.10*
pd_swrling2	*Calculates the Swerling 2 Prob. of detection*
fig2_11ab	*Reproduces Fig.s 2.11 a and b*
pd_swrling3	*Calculates the Swerling 3 Prob. of detection*
fig2_12	*Reproduces Fig. 2.12*
pd_swrling4	*Calculates the Swerling 4 Prob. of detection*
fig2_13	*Reproduces Fig. 2.13*
fig2_14	*Reproduces Fig. 2.14*

Name	Purpose (all functions have associated GUI)
fluct_loss	*Calculates the SNR loss due to RCS fluctuation*
fig2_15	*Reproduces Fig. 2.15*
myradar_visit2_1	*Program for "MyRadar" design case study visit 2.1*
myradar_visit2_2	*Program for "MyRadar" design case study visit 2.2*
fig2_21	*Reproduces Fig. 2.21*

Chapter 3: Radar Waveforms

Name	Purpose
fig3_7	*Reproduces Fig. 3.7*
fig3_8	*Reproduces Fig. 3.8*
hrr_profile	*Computes and plots HRR profile*
fig3_17	*Reproduces Fig. 3.17*

Chapter 4: The Radar Ambiguity Function

Name	Purpose
single_pulse_ambg	*Calculate and plot ambiguity function for a single pulse*
fig4_2	*Reproduces Fig. 4.2*
fig4_4	*Reproduces Fig. 4.4*
lfm_ambig	*Calculates and plot LFM ambiguity function*
fig4_5	*Reproduces Fig. 4.5*
fig4_6	*Reproduces Fig. 4.6*
train_ambg	*Calculates and plots ambiguity function for a train of coherent pulses*
fig4_8	*Reproduces Fig. 4.8*
barker_ambg	*Calculates and plots ambiguity function corresponding to a Barker code*

Name	Purpose
prn_ambig	*Calculates and plots ambiguity function corresponding to a PRN code*
myradar_visit4	*Program for "MyRadar" design case study visit 4*

Chapter 5: Pulse Compression

Name	Purpose
fig5_3	*Reproduces Fig. 5.3*
matched_filter	*Performs pulse compression using a matched filter*
power_integer_2	*Calculates the power integer of 2 for a given positive integer*
stretch	*Performs pulse compression using stretch processing*
fig5_14	*Reproduces Fig. 5.14*

Chapter 6: Surface and Volume Clutter

Name	Purpose
clutter_rcs	*Calculates and plots clutter RCS versus range*
myradar_visit6	*Program for "MyRadar" design case study visit 6*

Chapter 7: Moving Target Indicator (MTI) - Clutter Mitigation

Name	Purpose
single_canceler	*Performs single delay line MTI operation*
double_canceler	*Performs double delay line MTI operation*
fig7_9	*Reproduces Fig. 7.9*
fig7_10	*Reproduces Fig. 7.10*
fig7_11	*Reproduces Fig. 7.11*
myradar_visit7	*Program for "MyRadar" design case study visit 7*

Chapter 8: Phased Arrays

Name	Purpose
fig8_5	*Reproduces Fig. 8.5*
fig8_7	*Reproduces Fig. 8.7*
linear_array	*Calculates the linear array gain pattern*
circular_array	*Calculates the array pattern for a circular array*
rect_array	*Calculates the rectangular array gain pattern*
circ_array	*Calculates the circular array gain pattern*
rec_to_circ	*Calculates the boundary for rectangular array with circular boundary*
fig8_52	*Reproduces Fig. 8.52*

Chapter 9: Target Tracking

Name	Purpose
mono_pulse	*Calculate the sum and difference antenna patterns*
ghk_tracker	*implements the GHK filter*
fir9_21	*Reproduces Fig. 9.21*
kalman_filter	*Implements a 3-state Kalman filter*
fig9_28	*Reproduces Fig. 9.28*
maketraj	*Calculates and generates a trajectory*
addnoise	*Corrupts a trajectory*
kalfilt	*Implements a 6-state Kalman filter*

Chapter 10: Electronic Countermeasures (ECM)

Name	Purpose
ssj_req	*Implements SSJ radar equation*
sir	*Calculates and plots the S/(J+N) ratio*
bun_thru	*Calculates the burnthrough range*
soj_req	*Implements the SOJ radar equation*

Name	Purpose
range_red_factor	*Calculates the range reduction factor*
fig10_8	*Reproduces Fig. 10.8*

Chapter 11: Radar Cross Section (RCS)

Name	Purpose (all functions have associated GUI)
rcs_aspect	*compute and plot RCS dependency on aspect angle*
rcs_frequency	*compute and plot RCS dependency on frequency*
example11_1	*Used in solving Example on page*
rcs_sphere	*compute and plot RCS of a sphere*
rcs_ellipsoid	*compute and plot RCS of an ellipsoid*
rcs_circ_plate	*compute and plot RCS of a circular flat plate*
rcs_frustum	*compute and plot RCS of a truncated cone*
rcs_cylinder	*compute and plot RCS of a cylinder*
rcs_rect_plate	*compute and plot RCS of a rectangular flat plate*
rcs_isosceles	*compute and plot RCS of a triangular flat plate*
CappedWedgeTM	*Used to calculate the TM E-field for a capped wedge*
rcs_cylinder_complex	*reproduce Fig. 2.22*
swerlin_models	*reproduce Fig. 2.24*

Chapter 12: High Resolution Tactical Synthetic Aperture Radar (TSAR)

Name	Purpose
fig12_12_13	*Reproduces Figs. 12.12 and 12.13*

Chapter 13: Signal Processing

Name	Purpose
figs13	*Reproduces Fig. 13.2 through Fig. 13.5.*

Bibliography

Abramowitz, M. and Stegun, I. A., Editors, *Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables*, Dover Publications, 1970.

Balanis, C. A., *Antenna Theory, Analysis and Design*, Harper & Row, New York, 1982.

Barkat, M., *Signal Detection and Estimation*, Artech House, Norwood, MA, 1991.

Barton, D. K., *Modern Radar System Analysis*, Artech House, Norwood, MA, 1988.

Benedict, T. and Bordner, G., Synthesis of an Optimal Set of Radar Track-While-Scan Smoothing Equations, *IRE Transaction on Automatic Control, Ac-7*, July 1962, pp. 27-32.

Berkowitz, R. S., *Modern Radar - Analysis, Evaluation, and System Design*, John Wiley & Sons, Inc, New York, 1965.

Beyer, W. H., *CRC Standard Mathematical Tables*, 26th edition, CRC Press, Boca Raton, FL, 1981.

Billetter, D. R., *Multifunction Array Radar*, Artech House, Norwood, MA, 1989.

Blackman, S. S., *Multiple-Target Tracking with Radar Application*, Artech House, Norwood, MA, 1986.

Blake, L. V., *A Guide to Basic Pulse-Radar Maximum Range Calculation Part-I Equations, Definitions, and Aids to Calculation*, Naval Res. Lab. Report 5868, 1969.

Blake, L. V., *Radar-Range Performance Analysis*, Lexington Books, Lexington, MA, 1980.

Boothe, R. R., *A Digital Computer Program for Determining the Performance of an Acquisition Radar Through Application of Radar Detection Probability Theory*, U.S. Army Missile Command: Report No. RD-TR-64-2. Redstone Arsenal, Alabama, 1964.

Brookner, E., Editor, *Aspects of Modern Radar*, Artech House, Norwood, MA, 1988.

Brookner, E., Editor, *Practical Phased Array Antenna System*, Artech House, Norwood, MA, 1991.

Brookner, E., *Radar Technology*, Lexington Books, Lexington, MA, 1996.

Burdic, W. S., *Radar Signal Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1968.

Brookner, E., *Tracking and Kalman Filtering Made Easy*, John Wiley & Sons, New York, 1998.

Cadzow, J. A., *Discrete-Time Systems, an Introduction with Interdisciplinary Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

Carlson, A. B., *Communication Systems, An Introduction to Signals and Noise in Electrical Communication*, 3rd edition, McGraw-Hill, New York, 1986.

Carpentier, M. H., *Principles of Modern Radar Systems*, Artech House, Norwood, MA, 1988.

Compton, R. T., *Adaptive Antennas*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

Costas, J. P., A Study of a Class of Detection Waveforms Having Nearly Ideal Range-Doppler Ambiguity Properties, *Proc. IEEE 72*, 1984, pp. 996-1009.

Curry, G. R., *Radar System Performance Modeling*, Artech House, Norwood, 2001.

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

Dillard, R. A. and Dillard, G. M., *Detectability of Spread-Spectrum Signals*, Artech House, Norwood, MA, 1989.

Edde, B., *Radar - Principles, Technology, Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

Elsherbeni, Atef, Inman, M. J., and Riley, C., "Antenna Design and Radiation Pattern Visualization," The 19th Annual Review of Progress in Applied Computational Electromagnetics, ACES'03, Monterey, California, March 2003.

Fehlner, L. F., *Marcum's and Swerling's Data on Target Detection by a Pulsed Radar*, Johns Hopkins University, Applied Physics Lab. Rpt. # TG451, July 2, 1962, and Rpt. # TG451A, Septemeber 1964.

Fielding, J. E. and Reynolds, G. D., *VCCALC: Vertical Coverage Calculation Software and Users Manual*, Artech House, Norwood, MA, 1988.

Gabriel, W. F., Spectral Analysis and Adaptive Array Superresolution Techniques, *Proc. IEEE*, Vol. 68, June 1980, pp. 654-666.

Gelb, A., Editor, *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974.

Grewal, M. S. and Andrews, A. P., *Kalman Filtering - Theory and Practice Using MATLAB*, 2nd edition, Wiley & Sons Inc., New York, 2001.

Hamming, R. W., *Digital Filters*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1983.

Hanselman, D. and Littlefield, B., *Mastering Matlab 5, A Complete Tutorial and Reference,* Malab Curriculum Series, Prentice-Hall, Englewood Cliffs, NJ, 1998.

Hirsch, H. L. and Grove, D. C., *Practical Simulation of Radar Antennas and Radomes*, Artech House, Norwood, MA, 1987.

Hovanessian, S. A., *Radar System Design and Analysis*, Artech House, Norwood, MA, 1984.

James, D. A., *Radar Homing Guidance for Tactical Missiles*, John Wiley & Sons, New York, 1986.

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

Kay, S. M., *Fundamentals of Statistical Signal Processing - Estimation Theory*, Volume I, Prentice Hall Signal Processing Series, New Jersey, 1993.

Kay, S. M., *Fundamentals of Statistical Signal Processing - Detection Theory,* Volume II, Prentice Hall Signal Processing Series, New Jersey, 1993.

Klauder, J. R., Price, A. C., Darlington, S., and Albershiem, W. J., The Theory and Design of Chirp Radars, *The Bell System Technical Journal*, Vol. 39, No. 4, 1960.

Knott, E. F., Shaeffer, J. F., and Tuley, M. T., *Radar Cross Section*, 2nd edition, Artech House, Norwood, MA, 1993.

Lativa, J., Low-Angle Tracking Using Multifrequency Sampled Aperture Radar, *IEEE - AES Trans.*, Vol. 27, No. 5, September 1991, pp.797-805.

Levanon, N., *Radar Principles*, John Wiley & Sons, New York, 1988.

Lewis, B. L., Kretschmer, Jr., F. F., and Shelton, W. W., *Aspects of Radar Signal Processing*, Artech House, Norwood, MA, 1986.

Long, M. W., *Radar Reflectivity of Land and Sea*, Artech House, Norwood, MA, 1983.

Lothes, R. N., Szymanski, M. B., and Wiley, R. G., *Radar Vulnerability to Jamming*, Artech House, Norwood, MA, 1990.

Mahafza, B. R. and Polge, R. J., Multiple Target Detection Through DFT Processing in a Sequential Mode Operation of Real Two-Dimensional Arrays, *Proc. of the IEEE Southeast Conf. '90*, New Orleans, LA, April 1990, pp. 168-170.

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.

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.

Mahafza, B. R., *Introduction to Radar Analysis*, CRC Press, Boca Raton, FL, 1998.

Mahafza, B. R., *Radar Systems Analysis and Design Using MATLAB*, CRC Press, Boca Raton, FL, 2000.

Marchand, P., *Graphics and GUIs with Matlab*, 2nd edition, CRC Press, Boca Raton, FL, 1999.

Marcum, J. I., A Statistical Theory of Target Detection by Pulsed Radar, Mathematical Appendix, *IRE Trans.*, Vol. IT-6, April 1960, pp. 59-267.

Meeks, M. L., *Radar Propagation at Low Altitudes*, Artech House, Norwood, MA, 1982.

Melsa, J. L. and Cohn, D. L., *Decision and Estimation Theory*, McGraw-Hill, New York, 1978.

Mensa, D. L., *High Resolution Radar Imaging*, Artech House, Norwood, MA, 1984.

Meyer, D. P. and Mayer, H. A., *Radar Target Detection: Handbook of Theory and Practice*, Academic Press, New York, 1973.

Monzingo, R. A. and Miller, T. W., *Introduction to Adaptive Arrays,* John Wiley & Sons, New York, 1980.

Morchin, W., *Radar Engineer's Sourcebook*, Artech House, Norwood, MA, 1993.

Morris, G. V., *Airborne Pulsed Doppler Radar*, Artech House, Norwood, MA, 1988.

Nathanson, F. E., *Radar Design Principles*, 2nd edition, McGraw-Hill, New York, 1991.

Navarro, Jr., A. M., *General Properties of Alpha Beta, and Alpha Beta Gamma Tracking Filters*, Physics Laboratory of the National Defense Research Organization TNO, Report PHL 1977-92, January 1977.

North, D. O., An Analysis of the Factors which Determine Signal/Noise Discrimination in Pulsed Carrier Systems, *Proc. IEEE 51*, No. 7, July 1963, pp. 1015-1027.

Oppenheim, A. V. and Schafer, R. W., *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

Oppenheim, A. V., Willsky, A. S., and Young, I. T., *Signals and Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

Orfanidis, S. J., *Optimum Signal Processing, an Introduction*, 2nd edition, McGraw-Hill, New York, 1988.

Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, second edition, McGraw-Hill, New York, 1984.

Parl, S. A., New Method of Calculating the Generalized Q Function, *IEEE Trans. Information Theory,* Vol. IT-26, No. 1, January 1980, pp. 121-124.

Peebles, Jr., P. Z., *Probability, Random Variables, and Random Signal Principles*, McGraw-Hill, New York, 1987.

Peebles, Jr., P. Z., *Radar Principles*, John Wiley & Sons, New York, 1998.

Pettit, R. H., *ECM and ECCM Techniques for Digital Communication Systems*, Lifetime Learning Publications, New York, 1982.

Polge, R. J., Mahafza, B. R., and Kim, J. G., *Extension and Updating of the Computer Simulation of Range Relative Doppler Processing for MM Wave Seekers*, Interim Technical Report, Vol. I, prepared for the U.S. Army Missile Command, Redstone Arsenal, Alabama, January 1989.

Polge, R. J., Mahafza, B. R., and Kim, J. G., Multiple Target Detection Through DFT Processing in a Sequential Mode Operation of Real or Synthetic Arrays, *IEEE 21st Southeastern Symposium on System Theory*, Tallahassee, FL, 1989, pp. 264-267.

Poularikas, A. and Seely, S., *Signals and Systems*, PWS Publishers, Boston, MA, 1984.

Rihaczek, A. W., *Principles of High Resolution Radars*, McGraw-Hill, New York, 1969.

Ross, R. A., Radar Cross Section of Rectangular Flat Plate as a Function of Aspect Angle, *IEEE Trans*. AP-14:320, 1966.

Ruck, G. T., Barrick, D. E., Stuart, W. D., and Krichbaum, C. K., *Radar Cross Section Handbook*, Volume 1, Plenum Press, New York, 1970.

Ruck, G. T., Barrick, D. E., Stuart, W. D., and Krichbaum, C. K., *Radar Cross Section Handbook*, Volume 2, Plenum Press, New York, 1970.

Rulf, B. and Robertshaw, G. A., *Understanding Antennas for Radar, Communications, and Avionics*, Van Nostrand Reinhold, 1987.

Scanlan, M.J., Editor, *Modern Radar Techniques*, Macmillan, New York, 1987.

Scheer, J. A. and Kurtz, J. L., Editors, *Coherent Radar Performance Estimation*, Artech House, Norwood, MA, 1993.

Shanmugan, K. S. and Breipohl, A. M., *Random Signals: Detection, Estimation and Data Analysis*, John Wiley & Sons, New York, 1988.

Sherman, S. M., *Monopulse Principles and Techniques*, Artech House, Norwood, MA.

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.

Skillman, W. A., *DETPROB: Probability of Detection Calculation Software and User's Manual*, Artech House, Norwood, MA, 1991.

Skolnik, M. I., *Introduction to Radar Systems*, McGraw-Hill, New York, 1982.

Skolnik, M. I., Editor, *Radar Handbook*, 2nd edition, McGraw-Hill, New York, 1990.

Stearns, S. D. and David, R. A., *Signal Processing Algorithms*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

Stimson, G. W., *Introduction to Airborne Radar*, Hughes Aircaft Company, El Segundo, CA, 1983.

Stratton, J. A., *Electromagnetic Theory*, McGraw-Hill, New York, 1941.

Stremler, F. G., *Introduction to Communication Systems*, 3rd edition, Addison-Wesley, New York, 1990.

Stutzman, G. E., *Estimating Directivity and Gain of Antennas*, IEEE Antennas and Propagation Magazine 40, August 1998, pp 7-11.

Swerling, P., *Probability of Detection for Fluctuating Targets*, IRE Transaction on Information Theory, Vol IT-6, April 1960, pp. 269-308.

Van Trees, H. L., *Detection, Estimation, and Modeling Theory*, Part I. Wiley & Sons, Inc., New York, 2001.

Van Trees, H. L., *Detection, Estimation, and Modeling Theory*, Part III.Wiley & Sons, Inc., New York, 2001.

Van Trees, H. L., *Optimum Array Processing*, Part IV of Detection, Estimation, and Modeling Theory, Wiley & Sons, Inc., New York, 2002.

Tzannes, N. S., *Communication and Radar Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

Urkowtiz, H., *Signal Theory and Random Processes*, Artech House, Norwood, MA, 1983.

Urkowitz, H., *Decision and Detection Theory*, Unpublished Lecture Notes, Lockheed Martin Co., Moorestown, NJ.

Vaughn, C. R., Birds and Insects as Radar Targets: A Review, *Proc. IEEE*, Vol. 73, No. 2, February 1985, pp. 205-227.

Wehner, D. R., *High Resolution Radar*, Artech House, Norwood, MA, 1987.

White, J. E., Mueller, D. D., and Bate, R. R., *Fundamentals of Astrodynamics*, Dover Publications, 1971.

Ziemer, R. E. and Tranter, W. H., *Principles of Communications, Systems, Modulation, and Noise,* 2nd edition, Houghton Mifflin, Boston, MA, 1985.

Zierler, N., *Several Binary-Sequence Generators*, MIT Technical Report No. 95, Sept. 1955.