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

$$P_r = \sigma P_{Di}$$  \hspace{1cm} (11.1)
\[ \sigma \text{ denotes the target cross section. Define } P_{Dr} \text{ as the power density of the scattered waves at the receiving antenna. It follows that} \]

\[ P_{Dr} = \frac{P_r}{4\pi R^2} \]  

Equating Eqs. (11.1) and (11.2) yields

\[ \sigma = 4\pi R^2 \left( \frac{P_{Dr}}{P_{Di}} \right) \]  

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

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

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 \( \sigma_t \), where \( \sigma_t > \sigma \). Assuming a spherical coordinate system defined by \( (\rho, \theta, \phi) \), then at range \( \rho \) the target scattered cross section is a function of \( (\theta, \phi) \). Let the angles \( (\theta_r, \phi_r) \) define the direction of propagation of the incident waves. Also, let the angles \( (\theta_s, \phi_s) \) define the direction of propagation of the scattered waves. The special case,
when \( \theta_s = \theta_i \) and \( \phi_s = \phi_i \), defines the monostatic RCS. The RCS measured by the radar at angles \( \theta_s \neq \theta_i \) and \( \phi_s \neq \phi_i \) is called the bistatic RCS.

The total target scattered RCS is given by

\[
\sigma_t = \frac{1}{4\pi} \int_{\phi_s=0}^{2\pi} \int_{\theta_s=0}^{\pi} \sigma(\theta_s, \phi_s) \sin \theta_s \ d\theta_s \ d\phi_s
\]

(11.5)

The amount of backscattered waves from a target is proportional to the ratio of the target extent (size) to the wavelength, \( \lambda \), 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 \(10^\circ\), the electrical spacing between the two scatterers is

\[
elec\text{-spacing} = \frac{2 \times (1.0 \times \cos(10^\circ))}{\lambda}\tag{11.6}
\]

\(\lambda\) is the radar operating wavelength.
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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>scat_spacing</td>
<td>scatterer spacing</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>freq</td>
<td>radar frequency</td>
<td>Hz</td>
<td>input</td>
</tr>
<tr>
<td>rcs</td>
<td>array of RCS versus aspect angle</td>
<td>dBsm</td>
<td>output</td>
</tr>
</tbody>
</table>
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 11.3. Illustration of RCS dependency on aspect angle.

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>scat_spacing</td>
<td>scatterer spacing</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>freql</td>
<td>start of frequency band</td>
<td>Hz</td>
<td>input</td>
</tr>
<tr>
<td>frequ</td>
<td>end of frequency band</td>
<td>Hz</td>
<td>input</td>
</tr>
<tr>
<td>rcs</td>
<td>array of RCS versus aspect angle</td>
<td>dBsm</td>
<td>output</td>
</tr>
</tbody>
</table>

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 \( f_0 = 9 \text{GHz} \) is used to detect (observe) those two scatterers using a PRF \( f_r = 300 \text{Hz} \) for a period of 3 seconds. Finally, assume a NB bandwidth \( B_{NB} = 1 \text{MHz} \) and a WB bandwidth \( B_{WB} = 2 \text{GHz} \). It follows that the radar’s NB and WB range resolutions are respectively equal to \( \Delta R_{NB} = 150 \text{m} \) and \( \Delta R_{WB} = 7.5 \text{cm} \).

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

\[ E_x = E_1 \sin(\omega t - kz) \]  \hspace{1cm} (11.7)

\[ E_y = E_2 \sin(\omega t - kz + \delta) \]  \hspace{1cm} (11.8)

where \( k = 2\pi / \lambda \), \( \omega \) is the wave frequency, the angle \( \delta \) 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,

\[ \vec{E} = \hat{a}_x E_1 \sin(\omega t - kz) + \hat{a}_y E_2 \sin(\omega t - kz + \delta) \]  \hspace{1cm} (11.9)

Figure 11.9. Electric field components along the x and y directions. The positive z direction is out of the page.
where \( \hat{\mathbf{a}}_x \) and \( \hat{\mathbf{a}}_y \) 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

\[
\frac{E_x^2}{E_1^2} - \frac{2E_xE_y\cos\delta}{E_1E_2} + \frac{E_y^2}{E_2^2} = \left(\sin\delta\right)^2
\]  

(11.10)

Note that Eq. (11.10) has no dependency on \( \omega t \).

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

\[
AR = \frac{OA}{OB} \quad (1 \leq AR \leq \infty)
\]  

(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 \( \xi = 45^\circ \). When \( E_1 = E_2 \) and \( \delta = 90^\circ \), the wave is said to be Left Circularly Polarized (LCP), while if \( \delta = -90^\circ \) 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.](image-url)
In general, an arbitrarily polarized electric field may be written as the sum of two circularly polarized fields. More precisely,

$$\vec{E} = \vec{E}_R + \vec{E}_L$$  \hspace{1cm} (11.12)

where $\vec{E}_R$ and $\vec{E}_L$ are the RCP and LCP fields, respectively. Similarly, the RCP and LCP waves can be written as

$$\vec{E}_R = \vec{E}_V + j\vec{E}_H$$  \hspace{1cm} (11.13)

$$\vec{E}_L = \vec{E}_V - j\vec{E}_H$$  \hspace{1cm} (11.14)

where $\vec{E}_V$ and $\vec{E}_H$ are the fields with vertical and horizontal polarizations, respectively. Combining Eqs. (11.13) and (11.14) yields

$$E_R = \frac{E_H - jE_V}{\sqrt{2}}$$  \hspace{1cm} (11.15)

$$E_L = \frac{E_H + jE_V}{\sqrt{2}}$$  \hspace{1cm} (11.16)

Using matrix notation Eqs. (11.15) and (11.16) can be rewritten as

$$\begin{bmatrix} E_R \\ E_L \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \begin{bmatrix} E_H \\ E_V \end{bmatrix} = \begin{bmatrix} T \end{bmatrix} \begin{bmatrix} E_H \\ E_V \end{bmatrix}$$  \hspace{1cm} (11.17)

$$\begin{bmatrix} E_H \\ E_V \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ j & -j \end{bmatrix} \begin{bmatrix} E_R \\ E_L \end{bmatrix} = \begin{bmatrix} T \end{bmatrix}^{-1} \begin{bmatrix} E_H \\ E_V \end{bmatrix}$$  \hspace{1cm} (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:

case 1:  \( \hat{E}(t, z) = \hat{a}_x \cos(\omega_0 t + \frac{2\pi z}{\lambda}) + \hat{a}_y \sqrt{3} \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) \)

case 2:  \( \hat{E}(t, z) = \hat{a}_x \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) + \hat{a}_y \sin\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) \)

case 3:  \( \hat{E}(t, z) = \hat{a}_x \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) + \hat{a}_y \cos\left(\omega_0 t + \frac{2\pi z}{\lambda} + \frac{\pi}{3}\right) \)

case 4:  \( \hat{E}(t, z) = \hat{a}_x \cos\left(\omega_0 t + \frac{2\pi z}{\lambda}\right) + \hat{a}_y \sqrt{3} \cos\left(\omega_0 t + \frac{2\pi z}{\lambda} + \frac{\pi}{3}\right) \)

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 11.12. Circularly polarized electric field.

Figure 11.13. 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

$$
\begin{bmatrix}
  E_1^s \\
  E_2^s
\end{bmatrix}
= [S]
\begin{bmatrix}
  E_1^i \\
  E_2^i
\end{bmatrix}
= 
\begin{bmatrix}
  s_{11} & s_{12} \\
  s_{21} & s_{22}
\end{bmatrix}
\begin{bmatrix}
  E_1^i \\
  E_2^i
\end{bmatrix}
$$

(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:

$$
\begin{bmatrix}
  \sigma_{11} & \sigma_{12} \\
  \sigma_{21} & \sigma_{22}
\end{bmatrix}
= 4\pi R^2
\begin{bmatrix}
  |s_{11}|^2 & |s_{12}|^2 \\
  |s_{21}|^2 & |s_{22}|^2
\end{bmatrix}
$$

(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

\[
\begin{bmatrix}
E_H^t \\
E_V^t
\end{bmatrix} = \begin{bmatrix}
s_{HH} & s_{HV} \\
s_{VH} & s_{VV}
\end{bmatrix}\begin{bmatrix}
E_H^i \\
E_V^i
\end{bmatrix}
\tag{11.21}
\]

\[
\begin{bmatrix}
E_R^t \\
E_L^t
\end{bmatrix} = \begin{bmatrix}
s_{RR} & s_{RL} \\
s_{LR} & s_{LL}
\end{bmatrix}\begin{bmatrix}
E_R^i \\
E_L^i
\end{bmatrix}
\tag{11.22}
\]

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

\[
s_{RR} = -\frac{s_{VV} + s_{HH} - j(s_{HV} + s_{VH})}{2}
\]

\[
s_{RL} = \frac{s_{VV} + s_{HH} + j(s_{HV} - s_{VH})}{2}
\]

\[
s_{LR} = \frac{s_{VV} + s_{HH} - j(s_{HV} - s_{VH})}{2}
\]

\[
s_{LL} = -\frac{s_{VV} + s_{HH} + j(s_{HV} + s_{VH})}{2}
\]

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
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 \((\theta, \varphi)\), 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.

\[
\begin{align*}
 s_{HH} &= -\frac{s_{RR} + s_{RL} + s_{LR} - s_{LL}}{2} \\
 s_{VV} &= j\frac{s_{RR} - s_{LR} + s_{RL} - s_{LL}}{2} \\
 s_{HV} &= -j\frac{s_{RR} + s_{LR} - s_{RL} - s_{LL}}{2} \\
 s_{VV} &= \frac{s_{RR} + s_{LL} + js_{RL} + s_{LR}}{2}
\end{align*}
\]  

(11.26)

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figure}
\caption{Direction of antenna receiving backscattered waves.}
\end{figure}
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

\[
\frac{\sigma}{\pi r^2} = \left(\frac{j}{kr}\right) \sum_{n=1}^{\infty} (-1)^n (2n + 1) \left[ \frac{kr J_{n-1}(kr) - n J_n(kr)}{kr H_{n-1}^{(1)}(kr) - n H_n^{(1)}(kr)} \right] \]

(11.27)

where \( r \) is the radius of the sphere, \( k = 2\pi/\lambda \), \( \lambda \) 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

\[
Y_n(kr) = J_n(kr) + jY_n(kr) \]

(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,

\[
\sigma = \pi r^2 \quad r \gg \lambda \]

(11.29)

Second is the Rayleigh region (small sphere). In this case,

\[
\sigma \approx 9\pi r^2 (kr)^4 \quad r \ll \lambda \]

(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:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1 \quad (11.31)$$

One widely accepted approximation for the ellipsoid backscattered RCS is given by

$$\sigma = \frac{\pi a^2 b^2 c^2}{\left(a^2 \sin^2 \theta \cos^2 \phi + b^2 \sin^2 \theta \sin^2 \phi + c^2 \cos^2 \theta \right)^2} \quad (11.32)$$

When $a = b$, the ellipsoid becomes roll symmetric. Thus, the RCS is independent of $\phi$, and Eq. (11.32) is reduced to

![Figure 11.17. Ellipsoid.](image-url)
\[
\sigma = \frac{\pi b^4 c^2}{(a^2 (\sin \theta)^2 + c^2 (\cos \theta)^2)^2}
\]  

(11.33)

and for the case when \(a = b = c\),

\[
\sigma = \pi c^2
\]  

(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 \(\theta\) for \(\phi = 45^\circ\). 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 (\(\theta = 90^\circ\)) the RCS corresponds to that of a sphere of radius \(c\), and is often referred to as the broadside specular RCS value.

**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:

\[
[rcs] = rcs_ellipsoid (a, b, c, phi)
\]

Figure 11.18a. Ellipsoid backscattered RCS versus aspect angle.
where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>ellipsoid a-radius</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>$b$</td>
<td>ellipsoid b-radius</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>$c$</td>
<td>ellipsoid c-radius</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>$\phi$</td>
<td>ellipsoid roll angle</td>
<td>degrees</td>
<td>input</td>
</tr>
<tr>
<td>$rcs$</td>
<td>array of RCS versus aspect angle</td>
<td>dBsm</td>
<td>output</td>
</tr>
</tbody>
</table>

Fig. 11.18b shows the GUI workspace associated with function. To execute this GUI type “rcs_ellipsoid_gui” from the MATLAB Command window.

![GUI workspace](image)

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 $\phi$. The RCS is only aspect angle dependent. For normal incidence (i.e., zero aspect angle) the backscattered RCS for a circular flat plate is
For non-normal incidence, two approximations for the circular flat plate backscattered RCS for any linearly polarized incident wave are

\[
\sigma = \frac{4\pi^3 r^4}{\lambda^2} \quad \theta = 0^\circ \quad (11.35)
\]

\[
\sigma = \frac{\lambda r}{8\pi \sin \theta (\tan(\theta))^2} \quad (11.36)
\]

\[
\sigma = \pi k^2 r^4 \left(\frac{2J_1(2kr\sin\theta)}{2kr\sin\theta}\right)^2 (\cos\theta)^2 \quad (11.37)
\]

where \( k = 2\pi/\lambda \), and \( J_1(\beta) \) is the first order spherical Bessel function evaluated at \( \beta \). 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”.

**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:

\[
[\text{rcs}] = \text{rcs_circ_plate}(r, \text{freq})
\]

where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
<td>radius of circular plate</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>( \text{freq} )</td>
<td>frequency</td>
<td>Hz</td>
<td>input</td>
</tr>
<tr>
<td>( \text{rcs} )</td>
<td>array of RCS versus aspect angle</td>
<td>dBsm</td>
<td>output</td>
</tr>
</tbody>
</table>

© 2004 by Chapman & Hall/CRC CRC Press LLC
11.5.4. Truncated Cone (Frustum)

Figs. 11.21 and 11.22 show the geometry associated with a frustum. The half cone angle $\alpha$ is given by

$$\tan \alpha = \frac{(r_2 - r_1)}{H} = \frac{r_2}{L}$$  \hfill (11.38)

Define the aspect angle at normal incidence with respect to the frustum’s surface (broadside) as $\theta_n$. Thus, when a frustum is illuminated by a radar located at the same side as the cone’s small end, the angle $\theta_n$ is

$$\theta_n = 90^\circ - \alpha$$  \hfill (11.39)

Alternatively, normal incidence occurs at

$$\theta_n = 90^\circ + \alpha$$  \hfill (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.
where \( \lambda \) is the wavelength, and \( z_1, z_2 \) are defined in Fig. 11.21. Using trigonometric identities, Eq. (11.41) can be reduced to

\[
\sigma_{\theta_n} = \frac{8\pi(z_2^{3/2} - z_1^{3/2})^2}{9\lambda \sin\theta_n} \tan\alpha (\sin\theta_n - \cos\theta_n \tan\alpha)^2
\]

For non-normal incidence, the backscattered RCS due to a linearly polarized incident wave is

\[
\sigma = \frac{\lambda z \tan\alpha (\sin\theta - \cos\theta \tan\alpha)^2}{8\pi \sin\theta \sin\theta \tan\alpha + \cos\theta}
\]

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

\[
\sigma = \frac{\lambda z \tan\alpha (\tan(\theta - \alpha))}{8\pi \sin\theta}
\]

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

\[
\sigma = \frac{\lambda z \tan\alpha (\tan(\theta + \alpha))}{8\pi \sin\theta}
\]

For example, consider a frustum defined by \( H = 20.945 \text{cm} \), \( r_1 = 2.057 \text{cm} \), \( r_2 = 5.753 \text{cm} \). It follows that the half cone angle is \( 10^\circ \). 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 occurs at \( 100^\circ \), while for the second case it occurs at \( 80^\circ \). These plots can be reproduced using MATLAB function “\texttt{rcs_frustum_gui.m}” given in Listing 11.8 in Section 11.9.

**MATLAB Function “\texttt{rcs_frustum.m}”**

The function “\texttt{rcs_frustum.m}” computes and plots the backscattered RCS of a truncated conic section. The syntax is as follows:

\[
[r\text{cs}] = \texttt{rcs_frustum} (r1, r2, \text{freq}, \text{indicator})
\]
Figure 11.23a. Backscattered RCS for a frustum.

Figure 11.23b. Backscattered RCS for a frustum.
where

\[
\sigma_{\theta_n} = \frac{2\pi H^2 r_2^2 r_1^2}{\lambda \left[ r_1^2 (\cos \varphi)^2 + r_2^2 (\sin \varphi)^2 \right]^{1.5}} \tag{11.46}
\]

\[
\sigma = \frac{\lambda r_2^2 r_1^2 \sin \theta}{8\pi (\cos \theta)^2 \left[ r_1^2 (\cos \varphi)^2 + r_2^2 (\sin \varphi)^2 \right]^{1.5}} \tag{11.47}
\]

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. 11.24 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

\[
\sigma_{\theta_n} = \frac{2\pi H^2 r_2^2 r_1^2}{\lambda \left[ r_1^2 (\cos \varphi)^2 + r_2^2 (\sin \varphi)^2 \right]^{1.5}} \tag{11.46}
\]

\[
\sigma = \frac{\lambda r_2^2 r_1^2 \sin \theta}{8\pi (\cos \theta)^2 \left[ r_1^2 (\cos \varphi)^2 + r_2^2 (\sin \varphi)^2 \right]^{1.5}} \tag{11.47}
\]

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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_1 )</td>
<td>small end radius</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>( r_2 )</td>
<td>large end radius</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>freq</td>
<td>frequency</td>
<td>Hz</td>
<td>input</td>
</tr>
<tr>
<td>indicator</td>
<td>indicator = 1 when viewing from large end indicator = 0 when viewing from small end</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td>rcs</td>
<td>array of RCS versus aspect angle</td>
<td>dBsm</td>
<td>output</td>
</tr>
</tbody>
</table>

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

\[
\sigma_{\theta_n} = \frac{2\pi H^2 r_2^2 r_1^2}{\lambda \left[ r_1^2 (\cos \varphi)^2 + r_2^2 (\sin \varphi)^2 \right]^{1.5}} \tag{11.46}
\]

\[
\sigma = \frac{\lambda r_2^2 r_1^2 \sin \theta}{8\pi (\cos \theta)^2 \left[ r_1^2 (\cos \varphi)^2 + r_2^2 (\sin \varphi)^2 \right]^{1.5}} \tag{11.47}
\]

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, \( r = 0.125 \text{m} \) and \( H = 1 \text{m} \).
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)
```

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>radius r1</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>r2</td>
<td>radius r2</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>h</td>
<td>length of cylinder</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>freq</td>
<td>frequency</td>
<td>Hz</td>
<td>input</td>
</tr>
<tr>
<td>phi</td>
<td>roll viewing angle</td>
<td>degrees</td>
<td>input</td>
</tr>
<tr>
<td>CylinderType</td>
<td>'Circular,' i.e., $r_1 = r_2$</td>
<td>none</td>
<td>input</td>
</tr>
<tr>
<td></td>
<td>'Elliptic,' i.e., $r_1 \neq r_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rcs</td>
<td>array of RCS versus aspect angle</td>
<td>dBsm</td>
<td>output</td>
</tr>
</tbody>
</table>

Figure 11.25b. Backscattered RCS for an elliptical cylinder, $r_1 = 0.125\,m$, $r_2 = 0.05\,m$, and $H = 1\,m$.

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
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

\[ \sigma_V = \frac{b^2}{\pi} \left| \sigma_{1V} - \sigma_{2V} \left[ \frac{1}{\cos \theta} + \frac{\sigma_{2V}}{4(\sigma_{3V} + \sigma_{4V})} \right] \sigma_{5V} \right|^2 \]  
\[ \sigma_H = \frac{b^2}{\pi} \left| \sigma_{1H} - \sigma_{2H} \left[ \frac{1}{\cos \theta} - \frac{\sigma_{2H}}{4(\sigma_{3H} + \sigma_{4H})} \right] \sigma_{5H} \right|^2 \]

where \( k = \frac{2\pi}{\lambda} \) and

\[ \sigma_{1V} = \cos(kasin\theta) - j\frac{\sin(kasin\theta)}{\sin\theta} = (\sigma_{1H})^* \]

\[ \sigma_{2V} = \frac{e^{jk(a-\pi/4)}}{\sqrt{2\pi}(ka)^3/2} \]

\[ \sigma_{3V} = \frac{(1 + \sin\theta)e^{-jkasin\theta}}{(1 - \sin\theta)^2} \]

\[ \sigma_{4V} = \frac{(1 - \sin\theta)e^{jkasin\theta}}{(1 + \sin\theta)^2} \]

Figure 11.26. Rectangular flat plate.
Eqs. (11.50) and (11.51) are valid and quite accurate for aspect angles $0^\circ \leq \theta \leq 80^\circ$. For aspect angles near $90^\circ$, Ross obtained by extensive fitting of measured data an empirical expression for the RCS. It is given by

$$\sigma_{5V} = 1 - \frac{e^{j(2ka - \pi/2)}}{8\pi(ka)^3} \quad (11.56)$$

$$\sigma_{2H} = \frac{4e^{j(ka + \pi/4)}}{\sqrt{2\pi}(ka)^{1/2}} \quad (11.57)$$

$$\sigma_{3H} = \frac{e^{-jka\sin\theta}}{1 - \sin\theta} \quad (11.58)$$

$$\sigma_{4H} = \frac{e^{jka\sin\theta}}{1 + \sin\theta} \quad (11.59)$$

$$\sigma_{5H} = 1 - \frac{e^{j(2ka + (\pi/2))}}{2\pi(ka)} \quad (11.60)$$

Eqs. (11.50) and (11.51) are valid and quite accurate for aspect angles $0^\circ \leq \theta \leq 80^\circ$. For aspect angles near $90^\circ$, Ross obtained by extensive fitting of measured data an empirical expression for the RCS. It is given by

$$
\sigma_H \to 0
$$

$$\sigma_V = \frac{ab^2}{\lambda} \left\{ \left[ 1 + \frac{\pi}{2(2a/\lambda)^2} \right] + \left[ 1 - \frac{\pi}{2(2a/\lambda)^2} \right] \cos \left( 2ka - \frac{3\pi}{5} \right) \right\} \quad (11.61)
$$

The backscattered RCS for a perfectly conducting thin rectangular plate for incident waves at any $\theta, \varphi$ can be approximated by

$$\sigma = \frac{4\pi a^2 b^2}{\lambda^2} \left( \frac{\sin (ak \sin \theta \cos \varphi)}{ak \sin \theta \cos \varphi} \right)^2 \left( \frac{\sin (bk \sin \theta \sin \varphi)}{bk \sin \theta \sin \varphi} \right)^2 \left( \cos \theta \right)^2 \quad (11.62)
$$

Eq. (11.62) is independent of the polarization, and is only valid for aspect angles $\theta \leq 20^\circ$. 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:

$$[\text{rcs}] = \text{rcs\_rect\_plate} (a, b, \text{freq})$$

Figure 11.27a. Backscattered RCS for a rectangular flat plate.

Figure 11.27b. Backscattered RCS for a rectangular flat plate.
where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>short side of plate</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>(b)</td>
<td>long side of plate</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>freq</td>
<td>frequency</td>
<td>Hz</td>
<td>input</td>
</tr>
<tr>
<td>rcs</td>
<td>array of RCS versus aspect angle</td>
<td>dBsm</td>
<td>output</td>
</tr>
</tbody>
</table>

Fig. 11.27c shows the GUI workspace associated with this function.

\[
\sigma = \frac{4\pi A^2}{\lambda^2} (\cos \theta)^2 \sigma_0 \tag{11.63}
\]

\[
\sigma_0 = \frac{[(\sin \alpha)^2 - (\sin(\beta/2))^2]^2 + \sigma_{01}}{\alpha^2 - (\beta/2)^2} \tag{11.64}
\]

\[
\sigma_{01} = 0.25(\sin \phi)^2 [(2a/b) \cos \phi \sin \beta - \sin \phi \sin 2\alpha]^2 \tag{11.65}
\]

where \(\alpha = k \sin \theta \cos \phi\), \(\beta = k b \sin \theta \sin \phi\), and \(A = ab/2\). For waves incident in the plane \(\phi = 0\), the RCS reduces to

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 \(\theta \leq 30^\circ\) by
and for incidence in the plane $\varphi = \pi/2$

$$\sigma = \frac{4\pi A^2}{\lambda^2} \left(\frac{\sin \theta}{\alpha^4}\right)^2 \left[ \frac{(\sin \alpha)^4}{\alpha^4} + \frac{(\sin 2\alpha - 2\alpha)^2}{4\alpha^4} \right]$$  \hspace{1cm} (11.66)

and for incidence in the plane $\varphi = \pi/2$

$$\sigma = \frac{4\pi A^2}{\lambda^2} \left(\frac{\sin \theta}{\alpha^4}\right)^2 \left[ \frac{(\sin (\beta/2))^4}{(\beta/2)^4} \right]$$  \hspace{1cm} (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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>height of plate</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>$b$</td>
<td>base of plate</td>
<td>meters</td>
<td>input</td>
</tr>
<tr>
<td>$freq$</td>
<td>frequency</td>
<td>Hz</td>
<td>input</td>
</tr>
<tr>
<td>$phi$</td>
<td>roll angle</td>
<td>degrees</td>
<td>input</td>
</tr>
<tr>
<td>$rcs$</td>
<td>array of RCS versus aspect angle</td>
<td>dBsm</td>
<td>output</td>
</tr>
</tbody>
</table>
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 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 $(\rho_0, \phi_0)$ results in TM$^z$ incident field with the electric field expression given by

$$E_z^i = -I_e \frac{\omega \mu_0}{4} H_0^{(2)}(k|\rho - \rho_0|)$$  \hspace{1cm} (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:
where $J_v$ is the Bessel function of order $v$ and argument $x$ and $H_v^{(2)}$ is the Hankel function of the second kind of order $v$ and argument $x$. From Maxwell's equations, the magnetic field component $H_\phi$ is related to the electric field component $E_z$ for a TM$^\alpha$ wave by

$$H_\phi = \frac{1}{j\omega\mu} \frac{\partial E_z}{\partial \rho} \quad (11.71)$$
Thus, the magnetic field component $H_\phi$ in the various regions may be written as

\begin{align}
H_\phi^I &= \frac{k_1}{j\omega\mu_0} \sum_{n=0}^{\infty} a_n J_v'(k_i \rho) \sin \nu (\phi - \alpha) \sin \nu (\phi_0 - \alpha) \\
H_\phi^{II} &= \frac{k}{j\omega\mu_0} \sum_{n=0}^{\infty} \left( b_n J_v'(k \rho) + c_n H_v^{(2)'}(k \rho) \right) \sin \nu (\phi - \alpha) \sin \nu (\phi_0 - \alpha) \\
H_\phi^{III} &= \frac{k}{j\omega\mu_0} \sum_{n=0}^{\infty} d_n H_v^{(2)'}(k \rho) \sin \nu (\phi - \alpha) \sin \nu (\phi_0 - \alpha)
\end{align}

(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,

\begin{align}
E_z &= 0 \quad \text{at} \quad \phi = \alpha, 2\pi - \beta \\
E_z^I &= E_z^{II} \\
H_\phi^I &= H_\phi^{II} \quad \text{at} \quad \rho = a \\
E_z^{II} &= E_z^{III} \\
H_\phi^{II} - H_\phi^{III} &= -J_e \quad \text{at} \quad \rho = \rho_0
\end{align}

(11.74) (11.75)

The current density $J_e$ may be given in Fourier series expansion as

\begin{align}
J_e = \frac{I_e}{\rho_0} \delta (\phi - \phi_0) = \frac{2}{2\pi - \alpha - \beta} \frac{I_e}{\rho_0} \sum_{n=0}^{\infty} \sin \nu (\phi - \alpha) \sin \nu (\phi_0 - \alpha)
\end{align}

(11.76)

The boundary condition on the PEC surface is automatically satisfied by the $\phi$ dependence of the electric field Eq. (11.72). From the boundary conditions in Eq. (11.73)

\begin{align}
\sum_{n=0}^{\infty} a_n J_v (k_i a) \sin \nu (\phi - \alpha) \sin \nu (\phi_0 - \alpha) = \\
\sum_{n=0}^{\infty} \left( b_n J_v (k a) + c_n H_v^{(2)} (k a) \right) \sin \nu (\phi - \alpha) \sin \nu (\phi_0 - \alpha)
\end{align}

(11.77)
From the boundary conditions in Eq. (11.75), we have

$$\frac{k_i}{j\omega\mu_0} \sum_{n=0}^{\infty} a_n J_v'(k_i a) \sin (\varphi - \alpha) \sin (\varphi_0 - \alpha) =$$

$$\frac{k}{j\omega\mu_0} \sum_{n=0}^{\infty} \left( b_n J_v'(k a) + c_n H_v^{(2)\nu}'(k a) \right) \sin (\varphi - \alpha) \sin (\varphi_0 - \alpha)$$

From the boundary conditions in Eq. (11.75), we have

$$\sum_{n=0}^{\infty} \left( b_n J_v(k \rho_0) + c_n H_v^{(2)\nu}(k \rho_0) \right) \sin (\varphi - \alpha) \sin (\varphi_0 - \alpha) =$$

$$\sum_{n=0}^{\infty} d_n H_v^{(2)\nu}(k \rho_0) \sin (\varphi - \alpha) \sin (\varphi_0 - \alpha)$$

$$\frac{k}{j\omega\mu_0} \sum_{n=0}^{\infty} \left( b_n J_v'(k \rho_0) + c_n H_v^{(2)\nu}'(k \rho_0) \right) \sin (\varphi - \alpha) \sin (\varphi_0 - \alpha) =$$

$$\frac{k}{j\omega\mu_0} \sum_{n=0}^{\infty} d_n H_v^{(2)\nu}(k \rho_0) \sin (\varphi - \alpha) \sin (\varphi_0 - \alpha) - \frac{2}{2\pi - \alpha - \beta} \rho_0 \sum_{n=0}^{\infty} \sin (\varphi - \alpha) \sin (\varphi_0 - \alpha)$$

Since Eqs. (11.77) and (11.80) hold for all \( \varphi \), the series on the left and right hand sides should be equal term by term. More precisely,

$$a_n J_v(k \rho_0) + c_n H_v^{(2)\nu}(k \rho_0) =$$

$$d_n H_v^{(2)\nu}(k \rho_0)$$

From Eqs. (11.81) and (11.83), we have

$$a_n = \frac{1}{J_v(k_i a)} \left[ b_n J_v(k a) + c_n H_v^{(2)\nu}(k a) \right]$$

$$d_n = c_n + b_n \frac{J_v(k \rho_0)}{H_v^{(2)\nu}(k \rho_0)}$$
Multiplying Eq. (11.83) by \( H^{(2)}_{\nu}(\nu) \) and Eq. (11.84) by \( H^{(2)}_{\nu} \), and by subtraction and using the Wronskian of the Bessel and Hankel functions, we get

\[
b_n = -\frac{\pi \omega \mu I_e}{2\pi - \alpha - \beta} H^{(2)}_{\nu}(k \rho_0) \tag{11.87}
\]

Substituting \( b_n \) in Eqs. (11.81) and (11.82) and solving for \( c_n \) yield

\[
c_n = \frac{\pi \omega \mu I_e}{2\pi - \alpha - \beta} \left[ H^{(2)}_{\nu}(k \rho_0) \frac{k J'_{\nu}(ka) J_{\nu}(ka) - k J_{\nu}(ka) J'_{\nu}(ka)}{k H^{(2)'}_{\nu}(ka) J_{\nu}(ka) - k H^{(2)}_{\nu}(ka) J'_{\nu}(ka)} \right] \tag{11.88}
\]

From Eqs. (11.86) through (11.88), \( d_n \) may be given by

\[
d_n = \frac{\pi \omega \mu I_e}{2\pi - \alpha - \beta} \left[ H^{(2)}_{\nu}(k \rho_0) \frac{k J'_{\nu}(ka) J_{\nu}(ka) - k J_{\nu}(ka) J'_{\nu}(ka)}{k H^{(2)'}_{\nu}(ka) J_{\nu}(ka) - k H^{(2)}_{\nu}(ka) J'_{\nu}(ka)} - J_{\nu}(k \rho_0) \right] \tag{11.89}
\]

which can be written as

\[
d_n = \frac{\pi \omega \mu I_e}{2\pi - \alpha - \beta} \left\{ \frac{k J_{\nu}(ka) \left[ J'_{\nu}(ka) H^{(2)'}_{\nu}(k \rho_0) - H^{(2)'}_{\nu}(ka) J_{\nu}(k \rho_0) \right] + K}{k H^{(2)'}_{\nu}(ka) J_{\nu}(ka) - k H^{(2)}_{\nu}(ka) J'_{\nu}(ka)} \right\} \tag{11.90}
\]

Substituting for the Hankel function in terms of Bessel and Neumann functions, Eq. (11.90) reduces to

\[
d_n = -j \frac{\pi \omega \mu I_e}{2\pi - \alpha - \beta} \left\{ \frac{k J_{\nu}(ka) \left[ J'_{\nu}(ka) Y_{\nu}(k \rho_0) - Y'_{\nu}(ka) J_{\nu}(k \rho_0) \right] + K}{k H^{(2)'}_{\nu}(ka) J_{\nu}(ka) - k H^{(2)}_{\nu}(ka) J'_{\nu}(ka)} \right\} \tag{11.91}
\]

With these closed form expressions for the expansion coefficients \( a_n \), \( b_n \), \( c_n \) and \( d_n \), the field components \( E_z \) and \( H_\rho \) can be determined from Eq. (11.69) and Eq. (11.72), respectively. Alternatively, the magnetic field component \( H_\rho \) can be computed from

\[
H_\rho = -\frac{1}{j \omega \mu \rho} \left\{ \frac{1}{\partial E_z}{\partial \phi} \right\} \tag{11.92}
\]
Thus, the $H_\rho$ expressions for the three regions defined in Fig. 11.30 become

$$H_{\rho}^{I} = -\frac{1}{j\omega\mu\rho} \sum_{n=0}^{\infty} a_n J_v(k_1\rho) \cos v(\varphi - \alpha) \sin v\varphi_0 - \alpha)$$

$$H_{\rho}^{II} = -\frac{1}{j\omega\mu\rho} \sum_{n=0}^{\infty} \left(b_n J_v(k\rho) + c_n H_v^{(2)}(k\rho)\right) \cos v(\varphi - \alpha) \sin v\varphi_0 - \alpha)$$

$$H_{\rho}^{III} = -\frac{1}{j\omega\mu\rho} \sum_{n=0}^{\infty} d_n H_v^{(2)}(k\rho) \cos v(\varphi - \alpha) \sin v\varphi_0 - \alpha)$$

### 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

$$E_{z}^{III} = E_{z}^{i} + E_{z}^{s} = \sqrt{\frac{2j}{\pi k\rho}} e^{-j\varphi_0} \sum_{n=0}^{\infty} d_n J_v \cos v(\varphi - \alpha) \sin v\varphi_0 - \alpha)$$

$$E_{z}^{i} = -I_e \frac{\omega\mu_0}{4} \sqrt{\frac{2j}{\pi k\rho}} e H_{\rho} = -\frac{1}{j\omega\mu\rho} \frac{1}{\partial E_{z}}$$

Note that $d_n$ can be written as

$$d_n = -\frac{\omega\mu_0 I_e}{4} \partial_n^{\phi}$$

where

$$d_n^{\phi} = j \frac{4\pi}{2\pi - \alpha - \beta} \begin{bmatrix} kJ_v(k_1a) \left[ Y_v(k_1a) J_v(k_1\rho_0) - J_v(k_1a) Y_v(k_1\rho_0) \right] + K \\ kJ_v'(k_1a) \left[ Y_v'(k_1a) J_v(k_1\rho_0) - J_v'(k_1a) Y_v(k_1\rho_0) \right] \\ kH_v^{(2)}(k_1a) J_v(k_1a) - k_1H_v^{(2)}(k_1a) J_v'(k_1a) \end{bmatrix}$$

Substituting Eq. (11.95) into Eq. (11.94), the scattered field $f(\phi)$ is
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

\[
E_z^x = \frac{-\omega \mu_0 I_e}{4} \frac{2j}{\pi k \rho} e^{-jk\rho} \tag{11.97}
\]

\[
\left( \sum_{n=0}^{\infty} d_n^\nu \sin \nu(\varphi - \alpha) \sin (\varphi_0 - \alpha) - e^{jk\rho_0 \cos(\varphi - \varphi_0)} \right)
\]

11.6.3. Special Cases

**Case I:** \( \alpha = \beta \) (reference at bisector); The definition of \( \nu \) reduces to

\[
\nu = \frac{n\pi}{2(\pi - \beta)} \tag{11.100}
\]

and the same expression will hold for the coefficients (with \( \alpha = \beta \)).

**Case II:** \( \alpha = 0 \) (reference at face); the definition of \( \nu \) takes on the form

\[
\nu = \frac{n\pi}{2\pi - \beta} \tag{11.101}
\]

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
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,

\[
\begin{align*}
b_n &= -\frac{\pi \omega \mu J_e}{2\pi - \alpha - \beta} H_v^{(2)}(k\rho_0) \\
c_n &= \frac{\pi \omega \mu J_e}{2\pi - \alpha - \beta} H_v^{(2)}(k\rho_0) \frac{J_v(ka)}{H^*_v(ka)} \\
d_n &= \frac{j \pi \omega \mu J_e}{2\pi - \alpha - \beta} \frac{J_v(ka)J_v(k\rho_0) - J_v(ka)Y_v(k\rho_0)}{H^*_v(ka)} \\
a_n &= \frac{1}{J_v(ka)} \left[ b_nJ_v(ka) + c_nH^*_v(ka) \right] = 0
\end{align*}
\]

Note that \( H^*_v \) is obtained from Eq.(11.102).

\[
(11.102)
\]

**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 \( \nu \) reduce to \( n/2 \). Once, the electric field component \( E_z \) in the different regions is computed, the corresponding magnetic field component \( H_\varphi \) can be computed using Eq. (11.71) and the magnetic field component \( H_\rho \) may be computed as

\[
H_\rho = -\frac{1}{j \omega \mu \rho} \frac{1}{\partial E_z}{\partial \phi}
\]

\[
(11.104)
\]
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 $H_\rho$ 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 $H_\phi$ 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 $H_p$ 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 $H_q$ 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 $H_\rho$ 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 $H_\phi$ 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 $H_\rho$ 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 $H_\phi$ 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 $H_y$ 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 $H_\phi$ 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 $H_p$ 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 $H_\phi$ 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.
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.

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.
As an example, consider a circular cylinder with two perfectly conducting circular flat plates on both ends. Assume linear polarization and let $H = 1\, m$ and $r = 0.125\, m$. The backscattered RCS for this object versus aspect angle is shown in Fig. 11.56. Note that at aspect angles close to $0^\circ$ and $180^\circ$ 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_cylinder_complex.m” given in Listing 11.19 in Section 11.9.

![Figure 11.56. Backscattered RCS for a cylinder with flat plates.](image)

### 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 2

The Chi-square distribution applies to a wide range of targets; its pdf is given by

\[ f(\sigma) = \frac{m}{\Gamma(m)\sigma_{av}} \left( \frac{m\sigma}{\sigma_{av}} \right)^{m-1} e^{-m\sigma/\sigma_{av}} \quad \sigma \geq 0 \]  

(11.105)

where \( \Gamma(m) \) is the gamma function with argument \( m \), and \( \sigma_{av} \) 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

\[ f(\sigma) = \frac{1}{\sigma_{av}} \exp\left( - \frac{\sigma}{\sigma_{av}} \right) \quad \sigma \geq 0 \]  

(11.106)

where \( \sigma_{av} \) 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

\[ f(\sigma) = \frac{4\sigma}{\sigma_{av}^2} \exp\left(-\frac{2\sigma}{\sigma_{av}}\right) \quad \sigma \geq 0 \quad (11.107) \]

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 pdfs 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.](image)

**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);
aspect4 = 10*log10(major_axis/minor_axis)
alpha4 = (180/pi) * atan2(ay6(1),ax6(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);
aspect4 = 10*log10(major_axis/minor_axis)
alpha4 = (180/pi) * atan2(ay6(1),ax6(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);
aspect4 = 10*log10(major_axis/minor_axis)
alpha4 = (180/pi) * atan2(ay6(1),ax6(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);
aspect4 = 10*log10(major_axis/minor_axis)
alpha4 = (180/pi) * atan2(ay6(1),ax6(1))
end

% Case 4
ax6 = cos(wct);
ay6 = sqrt(3) .* cos(wct+(pi/3));
M6 = moviein(N);
figure(6)
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   = 0.000001;
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   =1000000+1000000*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
...
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 * c^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
        rcs = pi * c^2;
    end
end
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
%H 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)
\[ rcs(q,:) = (\pi \cdot a^2 \cdot b^2 \cdot c^2) ./ (a^2 \cdot \cos\phi_s \cdot (\sin\theta_r)^2 + b^2 \cdot \sin\phi_s \cdot (\sin\theta_r)^2 + c^2 \cdot (\cos\theta_r)^2)^2 ; \]

\[
\text{elseif}\ (a == b \& a ~= c) \\
rcs(q,:) = (\pi \cdot b^4 \cdot c^2) ./ ( b^2 \cdot (\sin\theta_r)^2 + c^2 \cdot (\cos\theta_r)^2 )^2 ; \\
\text{elseif}\ (a == b \& a == c) \\
rcs(q,:) = \pi \cdot c^2 ;
\]

end

rcs\_db = 10.0 \cdot \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 a 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 \cdot \pi^3 \cdot r^4 / lambda^2) + eps;

rcs\_mu(index) = rcs\_po(1);

else

x = (4. \cdot \pi \cdot r / lambda) \cdot \sin(aspect);

val1 = 4. \cdot \pi^3 \cdot r^4 / lambda^2;

val2 = 2. \cdot \text{besselj}(1,x) / x;

rcs\_po(index) = val1 \cdot (val2 \cdot \cos(aspect))^2 + eps;

% Compute RCS using Eq. (2.36)

val1m = lambda \cdot r;

end

% Compute RCS using Eq. (2.36)

val1m = lambda \cdot 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);
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
% 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
% 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

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

% 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 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

% 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']);
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.(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(-2.*ka+(pi/4))/2.*pi*ka);
% Compute vertical polarization RCS
rcs_v = (b^2/pi).*(abs(sigma1v-sigma2v.*((1./cos(theta))...+0.25.*sigma2v.*((sigma3v+sigma4v)).*(sigma5v).^2))).^2+eps;
% compute horizontal polarization RCS
rcs_h = (b^2/pi).*(abs(sigma1h-sigma2h.*((1./cos(theta))...-0.25.*sigma2h.*((sigma3h+sigma4h)).*(sigma5h).^2))).^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(angle)).^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”**

```matlab
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
```

© 2004 by Chapman & Hall/CRC CRC Press LLC
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
le = .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*s斯特m*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*s斯特m*Aterm/Bterm;
case 'None'
  EzLS(m) = Term0*ssterm*besselj(v,krhop);
end
end
EzFLs(ip) = abs(sum(EzLS));
else
  EzFLs(ip)=0;
end
EzFLs = EzFLs/max(EzFLs);

figure(1);
plot(xphi,EzFLs,'linewidth',1.5);
xlabel('Observation angle \phi^\circ');
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
  end
end
© 2004 by Chapman & Hall/CRC CRC Press LLC
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 summation
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, 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('E_z [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 ===
\[ Nc = \text{round}(1 + 2k \times \text{max}(\text{max}(\rho))) \]  
\% number of terms for series summation

\[ \text{Term} = \frac{4\pi}{2\pi - \alpha - \beta}; \]

for \( m = 1:Nc \)
\v = m*vi;

switch CapType
\begin{align*}
\text{case 'Diel'} \\
\quad b(m) &= \text{Term} \times \text{img}^v; \\
\quad c(m) &= -b(m) \times (k*\text{dbesselj}(v,ka)*\text{besselj}(v,k1a)...) \\
\qquad \quad -k1*\text{besselj}(v,ka)*\text{dbesselj}(v,k1a)) ... \\
\qquad \quad / (k*\text{dbesselh}(v,2,ka)*\text{besselh}(v,k1a)...) \\
\qquad \quad -k1*\text{besselh}(v,2,ka)*\text{dbesselj}(v,k1a)); \\
\quad a(m) &= (b(m) \times \text{besselj}(v,ka)+c(m) \times \text{besselh}(v,2,ka))/\text{besselj}(v,k1a); \\
\end{align*}

\text{case 'Cond'}
\begin{align*}
\quad b(m) &= -\text{Term} \times \text{img}^v; \\
\quad c(m) &= -b(m) \times \text{besselj}(v,ka)/\text{besselh}(v,2,ka); \\
\quad a(m) &= 0; \\
\end{align*}

\text{case 'None'}
\begin{align*}
\quad b(m) &= -\text{Term} \times \text{img}^v; \\
\quad c(m) &= 0; \\
\quad a(m) &= b(m); \\
\end{align*}

end
end
termhphi = sqrt(-1)*omega*mu0;
termhrho = -termhphi;

for \( i = 1:nx \)
\begin{align*}
\quad &\text{for } j = 1:ny \\
\quad &\quad \text{for } m = 1:Nc \\
\quad &\quad \quad \v = m*vi; \% Equation \\
\quad &\quad \quad [Ezt,Hphit,Hrhot] = \text{DielCappedWedgeTMFields_PW}(v,m,\rho(i,j),\phi(i,j), ... \\
\quad &\quad \quad \text{phip,ar,k1,}\alpha,\beta,a,b,c); \\
\quad &\quad \quad Eztt(m) = Ezt; \\
\quad &\quad \quad Hphitt(m) = Hphit; \\
\quad &\quad \quad Hrhott(m) = Hrhot; \\
\quad &\quad \end{align*}
\end{align*}

\begin{align*}
\quad &\text{EzPW}(i,j) = \text{sum}(Eztt); \\
\quad &\text{HphiPW}(i,j) = \text{sum}(Hphitt)/\text{termhphi}; \\
\quad &\text{HrhopW}(i,j) = \text{sum}(Hrhott)/\text{termhrho}; \\
\quad &\text{end}
\end{align*}
\end{align*}

\text{end}

\text{figure}(6);
\text{surf}(x,y,\text{abs}(EzPW));
\text{axis ('equal');}
Listing 11.13. MATLAB Function "DielCappedWedgeTMFields_Ls.m"

function [Ezt,Hphit,Hrhot] = DielCappedWedgeTMFields_Ls(v,m,rhoij,phiij,rhop,phip,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;
  % Initialization
  % Calculation of near field components
end

% Subsequent code...

© 2004 by Chapman & Hall/CRC CRC Press LLC
jvkrho = besselj(v,krho);
hvkrho = besselh(v,2,krho);
jvk1rho = besselj(v,k1rho);
djvkrho = dbesselj(v,krho);
djk1rho = 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 location
    Ezt = d(m)*hvkrho*ssterm;
    Hphit = k*d(m)*dhvkrho*ssterm;
    Hrhot = v*d(m)*hvkrho*scterm/rhoij;
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);
\( hvkrho = \text{besselh}(v,2,krho); \)
\( jvk1rho = \text{besselj}(v,k1rho); \)
\( djvkrho = \text{dbesselj}(v,krho); \)
\( djvk1rho = \text{dbesselj}(v,k1rho); \)
\( dhvkrho = \text{dbesselh}(v,2,krho); \)
\( ssterm = \sin(v*(\phiip-alpha))*\sin(v*(\phiij-alpha)); \)
\( scterm = \sin(v*(\phiip-alpha))*\cos(v*(\phiij-alpha)); \)

if \( \rhoij \leq a \)
   \% field point location is inside the cap region
   \[ Ezt = a(m) \ast jvk1rho \ast ssterm; \]
   \[ Hphit = k1 \ast a(m) \ast djvk1rho \ast ssterm; \]
   \[ Hrhot = v \ast a(m) \ast jvk1rho \ast scterm/rhoij; \]
else
   \% field point location is between the cap and the line source location
   \[ Ezt = (b(m) \ast jvkrho + c(m) \ast hvkrho) \ast ssterm; \]
   \[ Hphit = k \ast (b(m) \ast djvkrho + c(m) \ast dhvkrho) \ast ssterm; \]
   \[ Hrhot = v \ast (b(m) \ast jvkrho + c(m) \ast hvkrho) \ast scterm/rhoij; \]
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;
    h hh=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
end

% draw radial circles with dB ticks
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;
\[cst = \cos(th); \quad snt = \sin(th);\]
\[cs = [-cst; \; cst];\]
\[sn = [-snt; \; snt];\]
\[\text{plot}(rmax*cs,rmax*sn,ls,'color',tc,'linewidth',1,...\]
\[\quad 'handlevisibility','off')\]
\[
\text{% annotate spokes in degrees}\nrt = 1.1*rmax;\n\text{for } i = 1:\text{length}(th)\text{\quad text}(rt*cst(i),rt*snt(i),int2str(i*30),...\]
\[
\quad 'horizontalalignment','center',...\]
\[
\quad 'handlevisibility','off');\]
\[
\text{if } i == \text{length}(th)\text{\quad loc = int2str(0);\text{\quad else}}\text{\quad loc = int2str(180+i*30);\text{\quad end}}\text{\quad text}(-rt*cst(i),-rt*snt(i),loc,'horizontalalignment','center',...\]
\[
\quad 'handlevisibility','off')\]
\[
\text{% set view to 2-D}\text{\quad view(2);\text{\quad axis}(rmax*[-1 1 -1.15 1.15]);\text{\quad end}}\text{\quad % Reset defaults.}\text{\quad set(cax, 'DefaultTextFontAngle', fAngle , ...\}
\[
\quad 'DefaultTextFontName', fName , ...\]
\[
\quad 'DefaultTextFontSize', fSize, ...\]
\[
\quad 'DefaultTextFontWeight', fWeight, ...\]
\[
\quad 'DefaultTextUnits',fUnits );\text{\quad % Tranfrom data to dB scale}\text{\quad rmin = 0; rmax=1;\text{\quad rinc = (rmax-rmin)/rticks;\text{\quad rhodb=zeros(1,length(rho));}}\text{\quad for } i=1:\text{length}(rho)\text{\quad if } rho(i)==0\text{\quad rhodb(i)=0;\text{\quad else}}\text{\quad rhodb(i)=rmax+2*log10(rho(i))*rinc;\text{\quad end}}\text{\quad if rhodb(i)<=0\text{\quad rhodb(i)=0;\text{\quad end}}\text{\quad end}}\]
% 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
\[ \sigma = 0:0.001:6; \]
\[ \sigma_{\text{bar}} = 1.5; \]
\[ \text{swer}_3_4 = (4 ./ \sigma_{\text{bar}}^2) .* \sigma .* ... \]
\[ \quad \text{exp}(-2.* (\sigma ./ \sigma_{\text{bar}})); \]
\[ \%t.*\text{exp}(-t.^2)./2. \]
\[ \text{swer}_1_2 = (1./\sigma_{\text{bar}}) .* \text{exp}(-\sigma ./ \sigma_{\text{bar}}); \]
\[ \text{plot}(\sigma, \text{swer}_1_2,'k', \sigma, \text{swer}_3_4,'k'); \]
\[ \text{grid}; \]
\[ \text{gtext ('Swerling I,II');} \]
\[ \text{gtext ('Swerling III,IV');} \]
\[ \text{xlabel ('sigma');} \]
\[ \text{ylabel ('Probability density');} \]
\[ \text{title ('\sigma_{\text{bar}} = 1.5');} \]