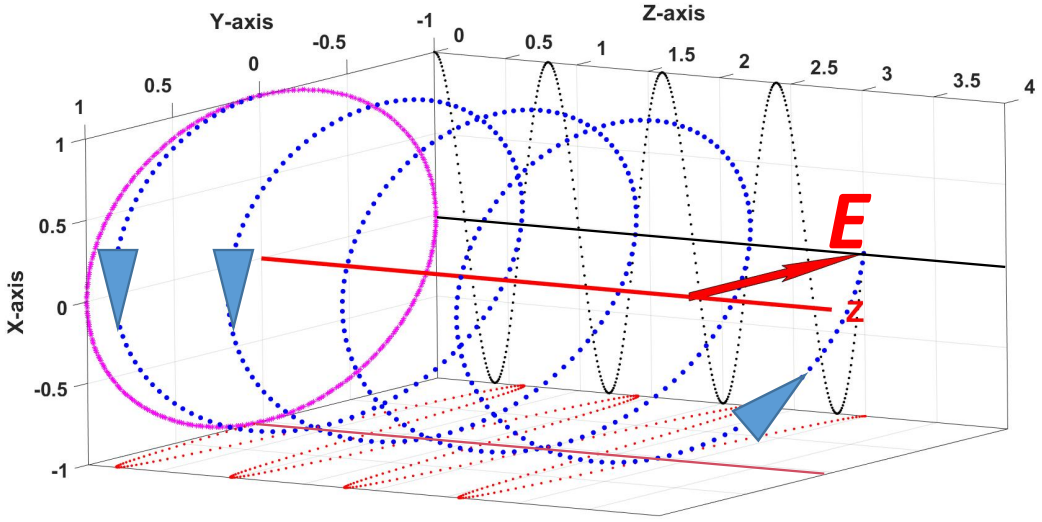
To boost electromagnetics' flavor, we accompanied many problems with short Matlab code acting as *a calculator and result illustrator*. We completely ignored the description of algorithms behind the codes because they are often quite complicated and required ample, very sophisticated as well as abstract mathematical specifics poorly helping to understand the physical picture. Such an approach lets us shift the focus from the often emasculated and practically fruitless problems primarily based on math transformations to more complicated but close to practical tasks. We hope to visualize typically invisible EM fields and stimulate in-depth analysis and the following discussion of the fundamental principles. Projecting images onto a big screen, the lectures may organize whole-class conversation making the audience significantly engaged. These scripts are for people who have never programmed in Matlab or CST before.
We cheer our readers to look through Matlab scripts and CST models to discover and sophisticate the algorithm embedded in them. We encourage you to use the student edition of MATLAB and CST STUDIO SUITE® to get enhanced problem understanding.

Attention. Regrettably, copy and paste into Matlab Command Window saves appropriate Matlab format just in Chrome Web Browser. You have to restart https://emfieldbook.com/ in https://www.google.com/ if your browser is different.

Problem 1. EM Wave Polarization.



Refresh the material of Section 5.1 in Chapter 5. Copy and paste Matlab script bellow into **Matlab Command Window,** enter the input parameters n =1, a, and ψ according to expression (5.2) in Chapter 5, and run it. The full-screen Figure 1 animates the space behavior of the E-vector, matching three consecutive wavefronts. The blue vector corresponds to the wavefront at $t = 0, z = 0$, the red one follows behind and reaches the cross-section $z = 0$ later with a delay of Δt, while the green one comes to the same cross-section $z = 0$ at $t = 2\Delta t$. The matching color thick dots trace each vector tip's movement along the spiral path forwards the z-axis. The magenta, red, and black dots in $\theta\varphi-, \theta Z -$ and $\varphi Z -$ plane demonstrate the behavior of the total E-vector (polarization ellipse) and its harmonic components correspondingly $E_\theta(z)$ and $E_\varphi(\mathrm{z})$. The larger diameter dots of the corresponding color depicts the components' magnitude at $z = 0$. Finally, the image in Figure 1 is rotated to be comparable to Figure 5.1.4 in Chapter 5[1]. Please check and understand all the cases mentioned in Section 5.1.1.

```
close all, clc; clear; a=input('Enter Magnitude of Current Ratio a = ')
psi=input('Enter Phase (|psi|<= pi/2) of Current Ratio [rad] psi = ')
```

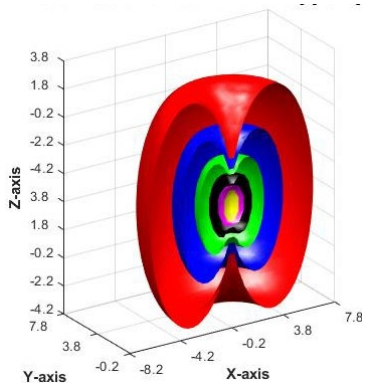
[1] Excellent animations can be found at Prof. Ian Copper's website http://www.physics.usyd.edu.au/teach_res/mp/doc/cemPol1.htm, and these two https://www.youtube.com/watch?v=Q0qrU4nprB0 and https://www.youtube.com/watch?v=8YkfEft4p-w.

```
n=input('Enter coefficient 0.9<= n <=1.1 giving differencies in propagation coefficients = ')
H=6*pi; omega_t=linspace(0,pi,5e0); gamma_z=linspace(0,H,1.5e2); x = ([-1, 1/8, 1/16, 1, 1/16, 1/8,-1]+1)/2;
y = [-1/20,-1/20,-1/10, 0, 1/10, 1/20, 1/20]/3;
for i=1:length(omega_t); for j=1:length(gamma_z); Exx=x*cos(omega_t(i)-gamma_z(j));
      Exy=y*sin(omega_t(i)-n*gamma_z(j)+psi); Eyx=y*sin(omega_t(i)-gamma_z(j));
      Eyy=x*cos(omega_t(i)-n*gamma_z(j)+psi); Ex(i,j,:)=Exx+Eyx; Ey(i,j,:)=a*(Exy+Eyy); end; end;
figure('units','normalized','outerposition',[0 0 1 1]); hold on; box on;
grid on; g1 = hgtransform; g2 = hgtransform; g3 = hgtransform;
text(0,0,1.25*H,'\bfZ','FontSize',45,'Color','k'); xlabel('\bf\Theta-axis'); ylabel('\bf\phi-axis');
ht(1)=text(1,-1,1,'\bfE_\theta vs. Z-coordinate'); set(ht(1),'FontSize',25,'Rotation',90)
ht(2)=text(-1,1,1,'\bfE_\phi vs. Z-coordinate'); set(ht(2),'FontSize',25,'Rotation',90,'Color','r')
v1=[0,0,0]; v2=[0,0,1.1*H]; v=[v2;v1]; hp=plot3(v(:,1),v(:,2),v(:,3),'k'); set(hp,'LineWidth',8);
r=-linspace(0,pi,10); th=linspace(0,2*pi,20); [R,T]=meshgrid(r,th);
X=R.*cos(T)/25; Y=R.*sin(T)/25; Z=R+H*1.25; s=surf(X,Y,Z); set(s,'FaceColor',[0 0 0]);
   for j=1:length(gamma_z); xlim([-1 1]); ylim([-1 1]); zlim([0 1.25]*H); view(144,33); axis square
       hpt1=patch('XData',Ex(1,j,:),'YData',Ey(1,j,:),'FaceColor','b','Parent',g1);
       hpt2=patch('XData',Ex(2,j,:),'YData',Ey(2,j,:),'FaceColor','r','Parent',g2);
       hpt3=patch('XData',Ex(3,j,:),'YData',Ey(3,j,:),'FaceColor','g','Parent',g3);
       g1.Matrix = makehgtform('translate',[0,0,gamma_z(j)]); g2.Matrix = makehgtform('translate',[0,0,gamma_z(j)]);
       g3.Matrix = makehgtform('translate',[0,0,gamma_z(j)]); h1=g1.Matrix; h2=g2.Matrix; h3=g3.Matrix;
       scatter3(hpt1.XData(4),hpt1.YData(4),h1(3,4),140,'b','filled');
       scatter3(hpt2.XData(4),hpt2.YData(4),h2(3,4),140,'r','filled');
       scatter3(hpt3.XData(4),hpt3.YData(4),h3(3,4),140,'g','filled');
       if j==1; M=100; else; M=20; end; scatter3(hpt1.XData(4),hpt1.YData(4),0,10,'m*');
       scatter3(-1,hpt1.YData(4),h1(3,4),M,'r','filled'); scatter3(hpt1.XData(4),-1,h1(3,4),M,'k','filled');
       scatter3(hpt2.XData(4),hpt2.YData(4),0,10,'m*'); scatter3(-1,hpt2.YData(4),h2(3,4),M,'r','filled');
       scatter3(hpt2.XData(4),-1,h2(3,4),M,'k','filled'); scatter3(hpt3.XData(4),hpt3.YData(4),0,10,'m*');
       scatter3(-1,hpt3.YData(4),h3(3,4),M,'r','filled'); scatter3(hpt3.XData(4),-1,h3(3,4),M,'k','filled');
       drawnow; if j~=length(gamma_z); delete(hpt1); delete(hpt2); delete(hpt3); end; end; delete(ht); camroll(-130);
```

1. What does the term polarization mean? How is this term connected to E- and H-vector orientation? May the polarization define as the direction of EM wave propagation?
2. Come back to Section 3.1.8 in Chapter 3 and explain the polarization effect through the photons migration.
3. What could you tell looking at Figure 1 about the motion of the E-vector tip?
4. What happens with polarization when two collocated linear polarized orthogonal electric dipoles emit EM waves? How to keep the total polarization linear? What does it mean slant polarization?
5. Can the polarization be rectangular? If the answer is yes, explain how to create it. If the answer is no, then why (*Hint.* Recall that E- and H-fields are related to each other by differential operations).
6. What are the differences between RHCP and LHCP? Run Matlab script to check your answer.
7. Prove analytically using the expression (5.4) and (5.6) that the superposition of LHCP and LHCP waves propagating in the same direction with equal speed can form the joint EM wave of linear or elliptically polarization.
8. What kind of polarization could form two linear polarized waves propagating in the same direction with equal speed and magnitude? Run Matlab script to check your answer.
9. What is the polarization of the propagating wave if $E_\theta = E_0 \cos(\omega t - kz)$ and $E_\varphi = \pm E_0 \sin(\omega t - kz)$? Run a Matlab script to check your answer.
10. The same question as #9 as $E_\theta = E_0 \cos(\omega t - kz)$ and $E_\varphi = \pm 2E_0 \sin(\omega t - kz)$? Run a Matlab script to check your answer. Do the projections in θZ-, φZ-, and $\theta\varphi$-plane reflect the harmonic nature of the propagating wave correctly?
11. The same question as #9 as $E_\theta = E_0 \cos(\omega t - kz)$ and $E_\varphi = \pm 2E_0 \cos(\omega t - kz)$? Run a Matlab script to check your answer. Do the projections in θZ-, φZ-, and $\theta\varphi$-plane reflect the harmonic nature of the propagating wave correctly?
12. The same question as #9 as $E_\theta = E_0 \cos(\omega t - kz)$ and $E_\varphi = E_0 \cos(\omega + kz)$? Run a Matlab script to check your answer. Do the projections in θZ-, φZ-, and $\theta\varphi$-plane reflect the harmonic nature of the propagating wave correctly?
13. Why do we need to match receiving and transmit antenna polarization? What kind of polarization is better to use for communication if the LP receiving antenna orientation is unknown or might vary?
14. Explain and provide examples of systems where LP is preferable (check page 204 in the book). Does the cost and complexity of the antenna influence the choice of polarization?

15. How and why can the CP polarization improve the performance of different types of radar and communication systems? Give an example like weather radar, satellite, and cellular communications, etc.
16. Give an example of systems that benefit from applications of the wave with twisted polarization.
17. Explain the terms co-, cross-polarization, and depolarization. What does it mean the axial ratio $AR = 0.5$? Might all these terms be applied to the waves with twisted polarization? Explain.
18. In Section 2.7.4 of Chapter 2, we learned that the propagation coefficient k of EM waves might depend on their structure (check later Section 6.8 in Chapter 6), particularly wave polarization (so-called Faraday Effect in ferrite, ionosphere, magneto-optic crystals, etc.). Matlab script allows simulating this phenomenon. Rerun the script entering, for example, $n = 0.9$, $a = 1$, and $\psi = \pi/2$ and watch the CP polarization variation as the wave propagates over the z-axis. Try to explain the results and think about how to use them practically for polarization transform. Do the projections in θZ-, φZ-, and $\theta\varphi$-plane continue to be the harmonic waves?

Problem 2. Near-field Zone vs. Far-field Zone. Refresh the material of Section 3.1.5 in Chapter 3, Section 4.3.1 in Chapter 4, Sections 5.2.5 and 5.2.9 in Chapter 5. Note the expression for E_r in (5.20) is not correctly printed and should be rewritten as



$$E_r = E_{r0}\left(\frac{1}{(kr)^2} - \frac{j}{(kr)^3}\right)\cos\theta e^{j(\omega t - kr)}$$

The field magnitudes in (5.20) are equal to $H_{\varphi 0} = 0.5 I_e^{exc} k\,\Delta l/\lambda$, $E_{\theta 0} = 120\pi H_{\varphi 0}$, $E_{\varphi 0} = 240\pi H_{\varphi 0}$ and can be found by applying (5.40) to (4.60). It was assumed that $I_e^{exc} = 1[A]$, $k = 2\pi$. The elementary electric radiator of $\Delta l/\lambda = 0.1$ is oriented along z-axis and enclosed in cube of 80λ x 80λ x 80λ divided uniformly into 211 x 211 x 211 cells of $\cong 0.4\lambda$ x 0.4λ x 0.4λ each. You might change any or all of these parameters editing Matlab script but be careful. The near-fields head very fast to infinity as $r \to 0$, making it

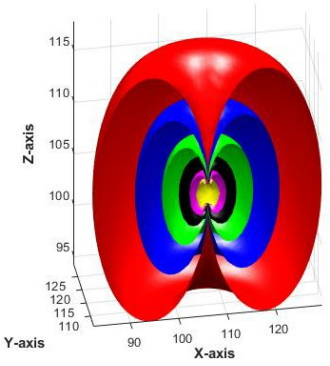


Figure 1

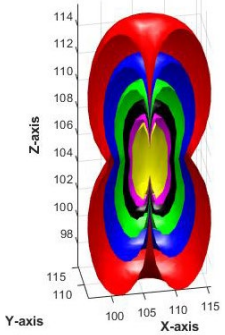


Figure 2

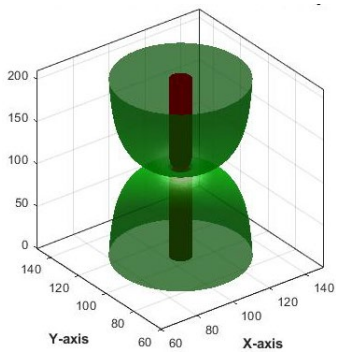


Figure 3

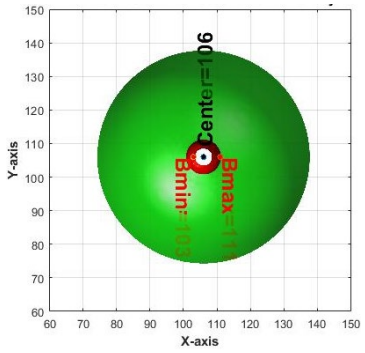


Figure 4

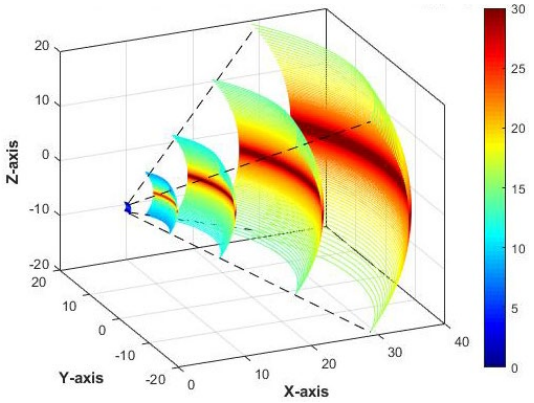


Figure 5

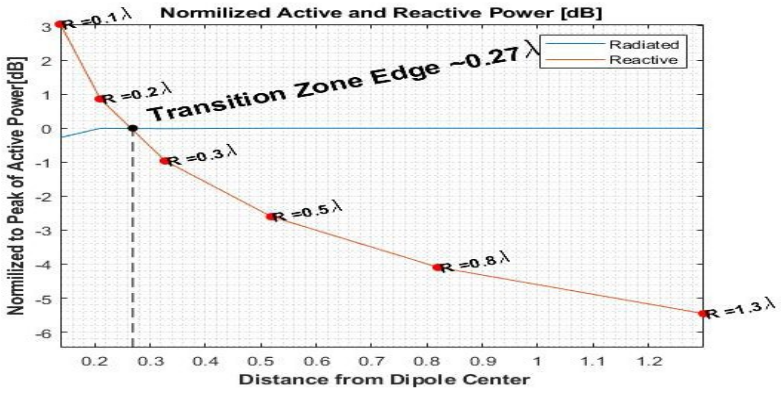


Figure 6

difficult to adjust the drawings' appearance, colormap properly, and scaling. Copy and paste Matlab script

below into **Matlab Command Window.** Six plots shown above should appear consequently. Figure 1 (top left on display) demonstrates the *cut in half* isosurfaces of the *radiated power density* normalized to the peak of total power density accumulated inside the cube. Over the entire isosurface, the magnitude of power density is constant and was chosen from 0.0001 (red surface relatively far from the radiator) to 0.01 (yellow surface) in 6 points of the logarithmic scale. Figure 2 (bottom left) is the same but for the normalized *reactive/accumulated power density* inside the cube. Figure 3 (top center) depicts the animated isosurfaces showing the radiated power density ratio to the reactive one inside the cube. The ratio magnitude is in logarithmic scale and includes 25 points from 0.1 (radiated = 0.1* reactive) to $10^{0.8} \sim 6.3$ (radiated = 6.3*reactive) to get smooth animation. The starting animation is colored in red, meaning that the reactive part exceeds the radiated one. The color switches to green as soon as this ratio surpasses one. The final image includes the red surface symbolizing that the radiated and reactive power densities are equal. The circles in Figure 4 (bottom center) represent the hourglass's top view in Figure 3. Figure 5 (top right) illustrates how the ratio between radiated and reactive power density changes as the EM energy propagates along the *x*-axis from the radiator virtual center at $x = 0$, $y = 0$, $z = 0$.

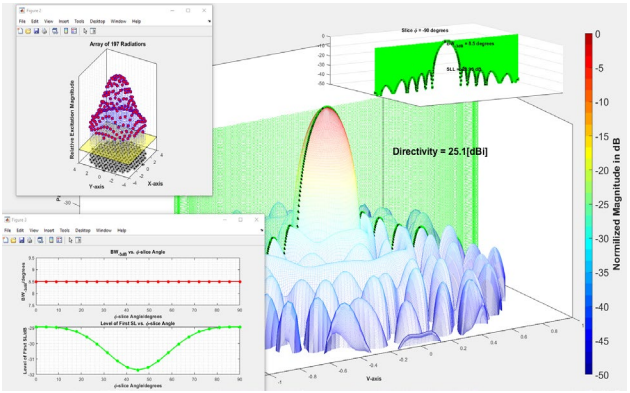
The contour lines on each surface and colorbar nearby give the graduated in dB reading. Keep in mind that the axes' scale in Figures 1 – 5 is specified into the cell index numbers from the cube's left bottom corner. To convert any of these indexes into real-world cell position normalized to wavelength, follow the expression N*80/211, where 80 is the cube side, and 211 is the quantity of each side split. For example, the dipole center is situated at the point marked as **Center** (106,106,106) or ~ $(40.2\lambda, 40.2\lambda, 40.2\lambda)/(2\pi)$ as it is pictured in Figure 4. The red points marked as Bmin and Bmax in Figure 4 are located on the bounding surface where the fields emitted by the dipole transfer smoothly into the near- and far-field zone. Therefore, the transition zone for elementary electric dipole extends between 3 and 5 cell lengths or from $3*80/211/(2\pi) = 0.18\lambda$ to $5*80/211/(2\pi) = 0.3\lambda$.[2] This slightly adjusted value is depicted in Figure 6 (bottom right) that describes the behavior of *radiated* power (blue line), i.e., traveling away and never returning, and *reactive* (red line), i.e., accumulated nearby, along with the *average* distance $R = \sqrt{ab}$ from the dipole center where *a* and *b* are the minor and major axis of ellipses in Figure 1.

1.

[2] Note that according to Figure 1 and 2 this limit depends on the elevation angle and exceeds that is given typically in the literature for the elementary electrical dipole (see, for example, problem 2.3-5 in W. L. Stutzman, G. A. Thiele, Antenna Theory and Design, 3rd edition, 2013) and our book on page 212.

```
close all; clc; clear; warning('off','all'); I=1; L_Lam=0.1; Lam=1; k=2*pi/Lam; BoxSize=40/Lam; N=2e2+11;
x=linspace(-1,1,N)*BoxSize; y=linspace(-1,1,N)*BoxSize; z=linspace(-1,1,N)*BoxSize;
[X,Y,Z]=meshgrid(x,y,z); [Az,El,R] = cart2sph(X,Y,Z);
Etheta=(60*pi*I*L_Lam*k)*((1i./R+1./R.^2-1i./R.^3).*cos(El).*exp(-1i*R));
Er=(120*pi*I*L_Lam*k)*((1./R.^2-1i./R.^3).*sin(El).*exp(-1i*R)); Ephi=0*Er;
E=sqrt(Etheta.*conj(Etheta)+Er.*conj(Er)); Emax=max(E(:)); E1=20*log10(E/Emax);
Hphi=(I*k*L_Lam/2)*((1i./R+1./R.^2).*cos(El).*exp(-1i*R));
H=sqrt(Hphi.*conj(Hphi)); Hmax=max(H(:)); H1=20*log10(H/Hmax);
Sr=Etheta.*conj(Hphi)/2; Stheta=-Er.*conj(Hphi)/2; SrR=real(Sr); SthetaR=real(Stheta); SR=sqrt(SrR.^2+SthetaR.^2);
SrI=imag(Sr); SthetaI=imag(Stheta); SI=sqrt(SrI.^2+SthetaI.^2); SS=sqrt(SI.^2+SR.^2); SSMax=max(SS(:));
Ratio=SR./SI; RatR=10*log10(real(Ratio)); DR = abs(SR/SSMax); DR(1:N/2,:,:)=NaN;
DI = abs(SI/SSMax); DI(1:N/2,:,:)=NaN;
f1=figure(1); movegui(f1,'northwest'); grid on; jc=0; cc={'r','b','g','k','m','y'}; hold all;
for id=logspace(-4, -2, 6)
    jc=jc+1; isoR(jc)=id; [fc,vr]=isosurface(DR,id); a=vr(fc(:,2),:)-vr(fc(:,1),:); b=vr(fc(:,3),:)-vr(fc(:,1),:);
    c=cross(a,b,2); area=sum(sqrt(sum(c.^2, 2)));
    pR=patch('Faces',fc,'Vertices',vr,'FaceColor',char(cc(jc)),'EdgeColor','none'); PowerR(jc)=id*area;
    a1=max(pR.Vertices(:))-106; b1=abs(min(pR.Vertices(:))-106); Rho(jc)=sqrt(a1*b1*80/211/pi)/2/pi; end
title('\bfIsosurfaces of Radiated Power Density [W/m^2]');
xlabel('\bfX-axis'); ylabel('\bfY-axis'); zlabel('\bfZ-axis'); axis tight;
view(-10,33); daspect([1,1,.4]); camlight left; camlight; lighting gouraud;
f2=figure(2); movegui(f2,'southwest'); grid on; jc=0;
for id=logspace(-4, -2, 6); jc=jc+1; isoI(jc)=id; [fc1,vr1]=isosurface(DI,id);
    pI = patch('Faces',fc1,'Vertices',vr1,'FaceColor',char(cc(jc)),'EdgeColor','none');
    a = vr1(fc1(:,2),:)-vr1(fc1(:,1),:); b = vr1(fc1(:,3),:)-vr1(fc1(:,1),:); c = cross(a,b,2); area=sum(sqrt(sum(c.^2,2)));
    PowerI(jc)=id*area; end
xlabel('\bfX-axis'); ylabel('\bfY-axis'); zlabel('\bfZ-axis');
title('\bfIsosurfaces of Reactive Power Density [W/m^2]'); axis tight;
view(-10,33); daspect([1,1,.4]); camlight left; camlight; lighting gouraud
f3=figure(3); box on; hold all; grid on; vals = logspace(-1, 0.8, 25);
[faces,verts,colors] = isosurface(Ratio,vals(1),R);
h=patch('Vertices',verts,'Faces',faces,'FaceVertexCData',colors,'FaceColor','interp','EdgeColor','none'); alpha(0.7);
axis([60  150 60 150 0 210]); xlabel('\bfX-axis'); ylabel('\bfY-axis'); camlight; lighting phong;
for id = vals; [fc2,vr2]=isosurface(Ratio,id); if id<1
    set(h, 'Faces', fc2, 'Vertices', vr2,'facecolor', 'r', 'edgecolor', 'none'); else
    set(h, 'Faces', fc2, 'Vertices', vr2,'facecolor', 'g', 'edgecolor', 'none'); end
  pause(0.2); view(3); axis square; end
xlabel('\bfX-axis'); ylabel('\bfY-axis'); title('\bfRatio of Radiated to Reactive Power Density')
[fc3,vr3]=isosurface(Ratio,1); pR1= patch('Faces',fc3,'Vertices',vr3,'FaceColor','r','EdgeColor','none');
Ax=get(gca,'Xtick'); Ay=get(gca,'Ytick'); Az=get(gca,'Ztick'); savefig(f3,'Poyn.fig');
ho=openfig('Poyn.fig'); movegui(figure(4),'south'); view(0,90); hsc = scatter3(106,106,117,'MarkerFaceColor','k');
hsc.SizeData = 20; hsc = scatter3(111,106,117,'MarkerFaceColor','r'); hsc.SizeData = 20;
hsc = scatter3(103,106,117,'MarkerFaceColor','r'); hsc.SizeData = 20;
text(114,106,117,'Bmax=111','Color','r','FontSize',18,'FontWeight','Bold','Rotation',-90);
text(106,109,117,'Center=106','Color','k','FontSize',18,'FontWeight','Bold','Rotation',90);
text(100,106,117,'Bmin=103','Color','r','FontSize',18,'FontWeight','Bold','Rotation',-90);
f5=figure(5); movegui(f5,'northeast'); grid on; box on; hold all; colormap(jet(80)); view(-24,26); colorbar; j1=0;
for jj=[.02 .1 0.2 0.35 0.5] ; j1=j1+1; [ysurf,zsurf] = meshgrid(y*jj,z*jj);
    rho=sqrt(max(abs(ysurf(:)))^2+max(abs(zsurf(:)))^2)*sqrt(2); xsurf=sqrt(abs(rho^2-ysurf.^2-zsurf.^2));
    hcs=contourslice(X,Y,Z,RatR,xsurf,ysurf,zsurf,150); cor1(j1,:)=hcs(1).Vertices(1,:); cor2(j1,:)=hcs(2).Vertices(1,:);
    cor3(j1,:)=hcs(3).Vertices(1,:); cor4(j1,:)=hcs(4).Vertices(1,:);     caxis([0 30]); drawnow; pause(0.2); end
title('\bfNormilized Ratio of Radiated to Reactive Power Density [dB]')
xlabel('\bfX-axis'); ylabel('\bfY-axis'); zlabel('\bfZ-axis');
plot3([cor1(1,1) cor1(j1,1)],[cor1(1,2) cor1(j1,2)],[cor1(1,3) cor1(j1,3)],'k--')
plot3([cor2(1,1) cor2(j1,1)],[cor2(1,2) cor2(j1,2)],[cor2(1,3) cor2(j1,3)],'k--')
plot3([cor3(1,1) cor3(j1,1)],-[cor3(1,2) cor3(j1,2)],[cor3(1,3) cor3(j1,3)],'k--')
plot3([cor4(1,1) cor4(j1,1)],-[cor4(1,2) cor4(j1,2)],[cor4(1,3) cor4(j1,3)],'k--'); shading interp;
f6=figure(6); movegui(f6,'southeast');
PRe=10*log10(PowerR/max(PowerR)); PIm=10*log10(PowerI/max(PowerR));
px=(Rho(4)+Rho(5))/2; plot(Rho,PRe,Rho,PIm); hold all;
plot(px+0*Rho,linspace(min(PIm)-1,0,length(Rho)),'--k'); grid minor; ylim([min(PIm)-1 4]);
hsc = scatter(px,0,'filled','k'); hsc.SizeData = 30;
text(px*1.1,0.4,'Transition Zone Edge ~0.27\lambda','Color','k','FontSize',15,'FontWeight','Bold','Rotation',15);
for jj=1:6; scatter(Rho(jj),PIm(jj),'filled','r'); hsc.SizeData = 30;
    ht=text(Rho(jj),PIm(jj),['R =',num2str(round(Rho(jj),1)),'\lambda']);
    set(ht,'Color','k','FontSize',10,'FontWeight','Bold','Rotation',15); end
title('\bfNormilized Active and Reactive Power [dB]'); axis tight; ylabel('\bfNormilized to Peak of Active Power[dB]');
xlabel('\bf Distance from Dipole Center'); legend('Radiated','Reactive')
```

Problem 3. Antenna Radiation Parameters. This workout's main objective is to illustrate through the animation 3D radiation pattern, multiple 2D pattern slices, half-power-beamwidth (HPBW or BW-3db) sidelobe level (SLL), antenna factor, directivity, effect of radiator number and aperture shape, impact of tapering, etc. The Matlab code you can find at https://1drv.ms/f/s!AjtsKS-uvNP1avsbUFY9UsxLVVE is universal and allows you to provide much broader research just slightly editing the code. The mathematical model illustrates the array factor of discrete and independent (no mutual coupling) radiators organized as a periodical array of rectangular or circular shape in XY-plane. The code behind the model might be merely altered if you wish. Please do not hesitate to propose new features, examples, or any constructive comments!



Refresh the Sections 5.2.6, 5.2.7, 5.2.9-5.2.14, 5.4.1, 5.4.2, 5.4.5, 5.6.1 in Chapter 5. Since all plots are represented in *uv*-coordinates, pay special attention to images and expressions in Section 5.2.6 and Figure 5.6.3 in Section 5.6.1, defining the pattern slices. Go to the mentioned website, click subsequently on the icon Document and AntennaParameters. Mark, copy and download the AntennaParametersUV.m file to a newly created directory named Antenna Parameters, for example. Then open the AntennaParametersUV.m file in Matlab and run the animation.

The figure above depicts the final screenshot you can watch during the simulation. The large central image is the normalized array pattern in dB. The perpendicular plane (green) rotates stepwise inside the azimuthal sector $-90° \le \varphi \le 0°$ with step 4.5°, cutting the pattern into vertical slices. The cross-line marked by dot-line is the seeking 2D antenna pattern $F(\sin\theta)$, where θ is the elevation angle while the azimuth angle φ is fixed. The sequential patterns are synchronously depicted as the picture-in-picture (top right corner) with a green curtain. Usually, for array sizes exceeding several wavelengths $\sin\theta \cong \theta$ [radian], this plot is similar to a classical pattern in Figure 5.2.8 on page 216. The array directivity, present φ-value, BW_{-3db}, and the first SLL (typically, the highest) are revealed. The left top corner figure demonstrates the array structure (in black on base) and each element's relative excitation magnitude (red sports above the yellow plane of zero volumes). Once the Matlab code has been launched, you have multiple options to satisfy your curiosities:

1. First of all, select between arrays with a rectangular and circular aperture.
2. To define the rectangular array, enter the *normalized to wavelength* separation *dx* and *dy* between the array elements (see Figure 5.6.1 on page 261) and the element number *Nx* and *Ny* in each row and column. Then, the computer asks you to choose between the uniform, simple cosine, and Dolph-Chebyshev amplitude distribution.
3. To define the circular aperture, enter the aperture radius and separation *d* between elements. Both values should be *normalized to wavelength.* The program uniformly spreads the discrete elements inside the circle. The next step is to decide what kind of excitation distribution you wish to simulate: uniform, simple cosine tapering, or more sophisticated Taylor profile with the specified SLL[3].
4. Watch the animation paying particular attention to the patterns' shape and comments appearing in the picture-in-picture window (top right).

Warning! If the defined array comprises more than 500 elements, you should be patient, letting the computer finish its task.

[3] For simplicity, Taylor distribution devised for antennas excited continuously was applied to a planar array of discrete elements by sampling. Such an approach limits |SLL| <= 30 dB. If Matlab Phased Array Antenna Toolbox is included in your package, replace several code lines with Amp = taylortaperc(pos,diam). See https://www.mathworks.com/help/phased/ref/taylortaperc.html for details.

Problem 4. Uniform Circular Array. Refresh the Sections 5.6.2 in Chapter 5. Since some plots are represented in *uv*-coordinates, i.e., according to Figure 1

$$u = \cos\theta\cos\varphi$$
$$v = \cos\theta\sin\varphi$$

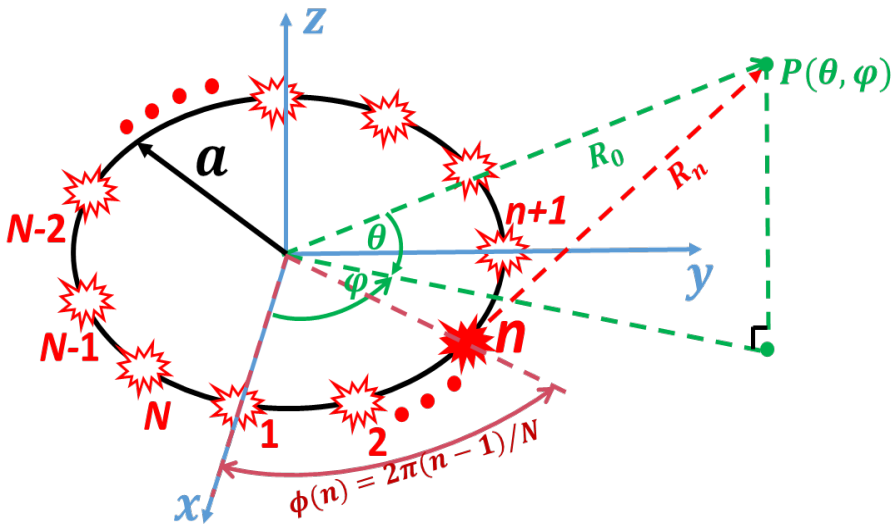


Figure 1. Uniform circular array in *XY*-plane

Pay special attention to images and expressions in Section 5.2.6 and Figure 5.6.3 in Section 5.6.1, defining the pattern slices. Figure 1 depicts the schematic of a circular array containing N radiators. This generic array has elements equally spaced along the periphery of a circle, and the radiators are excited with equal amplitude. Here $\boldsymbol{a}$ is the circular array radius, φ is the angle in the azimuthal plane, $\phi(n) = 2\pi(n-1)/N$, is the n^{th} element's azimuth position. Therefore, the radiator coordinates in *XY*-plane are (n = 1,2,…,N)

$$x(n) = a * \cos\phi(n), \quad y(n) = a * \sin\phi(n)$$

The transmit path R_n from this radiator to the point $P(\theta,\varphi)$,$-\pi/2 \le \theta \le \pi/2$ and $0 \le \varphi \le 2\pi$, is

$$R_n = \sqrt{(R_0\sin\theta)^2 + (R_0u - x(n))^2 + (R_0v - y(n))^2} = \sqrt{R_0^2 + a^2 - 2aR_0\cos\theta\cos\left(\varphi - \phi(n)\right)}$$

To direct the radiation pattern peak at the point $P(\theta_0,\varphi_0)$, the coherent field summation from all radiators at this point is required. Therefore, we need to offset the transmit paths' differences by phase-shifting the signals coming from a generator to each radiator ($k = 2\pi/\lambda$) as

$$\delta_n = k\sqrt{R_0^2 + a^2 - 2aR_0\cos\theta_0\cos\left(\varphi_0 - \phi(n)\right)}$$

or providing the equivalent time delay (check Section 5.5.4). Then the total field at the point $P(\theta,\varphi)$ is the sum of fields produced by all the radiators and can be written in the same form as for a planar array (see the expression (5.106) and (5.107) in Section 5.6.1)

$$E_\Sigma = e^{j\omega_s t} f(\theta,\varphi) \sum_{n=1}^{N} e^{-j(kR_n - \delta_n)}$$

Here $\omega_s = 2\pi f_s$, f_s is the baseband signal frequency of the propagating in free space EM-wave, $f(\theta,\varphi)$ is the radiator pattern assumed the same for all radiators, and $kR_n - \delta_n = 0$, $e^{-j(kR_n-\delta_n)} = 1$ as $\theta = \theta_0$, $\varphi = \varphi_0$. You can find the Matlab code at https://1drv.ms/f/s!AjtsKS-uvNP1avsbUFY9UsxLVVE, where $f(\theta,\varphi) = \cos\theta$. It roughly corresponds to a short vertical dipole with single director or reflector. Open the file directory Circular Array, mark, copy and download the CircularArray.m file to a newly created directory named Circular Array, for example. Then open the CircularArray.m file in Matlab and run the simulation.

Keep in mind the array radius must be entered in the unit normalized to wavelength and scan angles in degrees. Figure 2 below illustrates the plots you should get. The top right depicts the array geometry where all sizes (r = 2 and separation between adjacent radiators is 0.695) are normalized to wavelength. The next top one is the array pattern in UV-coordinates as the main beam is steered to 65°. Two bottom plots demonstrate the azimuth and elevation pattern. All the patterns illustrate a uniform circular array's primary disadvantage, a relatively high level of sidelobes. Check please through the simulation that the circular array provides a beam scan with almost no main beam distortions at any angle between 0° and 360°.

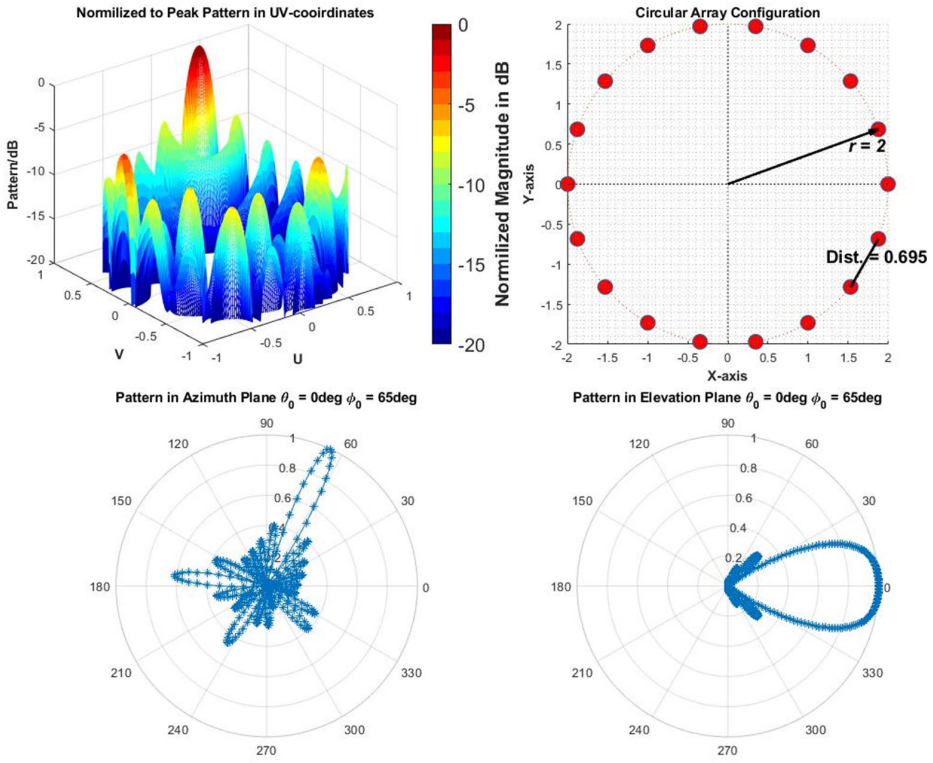
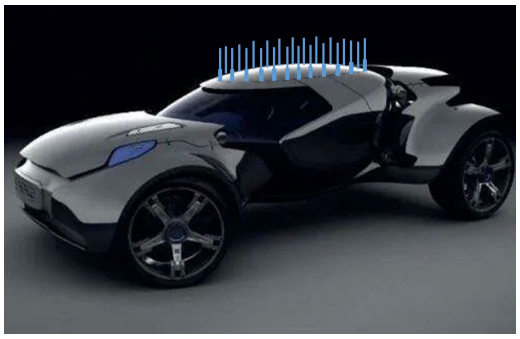


Figure 2. Simulation result plots

Problem 5. Uniform Circular Array as Frequency Diverse Array (FDA). Ultra-Fast 360° Bearing (Azimuth) Scan. Refresh Sections 5.5.6 in Chapter 5. The figure on the left demonstrates (not in scale) the circular array of vertical monopoles mounted on the car's roof. Such an antenna can serve as an integral part of anti-collision, communication, or traffic monitoring systems[4]. You can find more information by looking through the referenced book. A circular structure's significant advantage is its ability to scan a beam azimuthally through 360° with little change in either the beamwidth or the sidelobe level. The *frequency modulated scan* technique described in Section 5.5.6 converts such an array into an ultra-fast beam steering structure without the electric power-greedy and costly analog or digital phase shifters. The physical sizes, radar stealth, and mass of radar drastically reduce as the operating frequencies are shifted to the mm-wave range. The overall processing of the FDA data is even simpler and faster than the conventional phased-array radar processing[5].



You can find the Matlab code at https://1drv.ms/f/s!AjtsKS-uvNP1avsbUFY9UsxLVVE. Open the file directory UltraFast, mark, copy and download the UltraFastArray.m file to a newly created directory named UltraFast, for example. Then open the UltraFastArray.m file in Matlab and run the simulation. Just follow the instructions on the screen. The main beam rotation was pushed by transforming the azimuth angle $\varphi_0 = 2\pi ft$ into time dependable value, i.e., defining the phase shift between radiators as (check Problem 4)

$$\delta_n(t) = k\sqrt{R_0^2 + a^2 - 2aR_0\cos\theta_0\cos\left(2\pi ft - \phi(n)\right)}$$

[4] V. Rabinovich and N. Alexandrov, Antenna Arrays and Automotive Applications, Springer, 2013.

[5] Yunhan Dong, Frequency Diverse Array Radar Data Processing, https://radar2018.org/abstracts/pdf/abstract_49.pdf

It implies that the azimuth 360° coverage takes $1/f$ seconds, i.e., $10^{-6}s$ as f = 1MHz, which defines such bearing as ultra-fast. Note that even low-frequency surveillance radars typically work at frequencies above $f_s = 150\text{MHz}$. Thus, we can always assume $f \ll f_s$[6]. Notably, it is convenient to match single radar pulse and 360°-scan duration, i.e., provide a within-pulse scan revisiting the same target repeatedly. Finally, we obtain

$$|E_\Sigma| = \left| f(\theta, \varphi) \sum_{n=1}^{N} e^{-j(kR_n - \delta_n(t))} \right|$$

As before $R_n = \sqrt{R_0^2 + a^2 - 2aR_0 \cos\theta \cos\left(\varphi - \phi(n)\right)}$.

The described model is slightly simplified and does not include the time dependable signal phase $\omega_s(t - R_0/c)$. We ask the reader to update the Matlab code if you wish.

[6] For good animation visibility, the frequency *f* in Matlab simulation was chosen 1Hz, meaning the full 360o azimuth scan takes 1 second. You can adjust this parameter if you wish.