```
%****************************************************************
% Helical Antenna
% -------------------------------------------------------------
% Programmed by Bo Yang, 10/2002
% Reference: "Antenna Theory: Analysis and Design, Second Edition,
% Constantine A. Balanis", Section 10.3.1, Helical Antenna
% -------------------------------------------------------------
%
% This program calculates the radiation characteristics of a helical
% antenna, both in the Normal mode and the Axial(End-fire) mode. The
% radiation characteristics which are calculated are:
% -------------------------------------------------------------
% I. Normal Mode
% a. Pitch angle alpha (in degrees)
% b. Axial ratio (AR)
% c. HPBW (in degrees)
% c. Directivity (dimensionless and in dB)
% ***
% II. Axial (End-Fire) Mode
% a. Pitch angle alpha (in degrees)
% b. Input impedance (ohms)
% c. Axial ratio (AR)
% d. Relative phase velocity ratio p
% e. HPBW (in degrees) (both approximate and numerical)
% f. FNNW (in degrees) (both approximate and numerical)
% g. Directivity (dimensionless and in dB) (both approximate and numerical)
% -------------------------------------------------------------
% Normalized radiation pattern of both modes are plotted. To get
% correct output, just follow the instructions below.
%
%%%%%%%%%%%%%%%%
% Brief Manual %
%%%%%%%%%%%%%%%%
% -------------------------------------------------------------
% 1.Choice of output
% -----------------
% Input 1 for Screen output
% Input 2 for File output
%
% 2.Choice of radiation mode
% ------------------------
% Input 1 to select Normal mode
% Input 2 to select Axial (End-Fire) mode
%
% 3.Input
% -------
% a. Number of turns N
% b. Circumference of loops C (in lambda)
% c. Spacing between turns S (in lambda)
% ***
% For axial mode, also need to select end-fire mode:
% Input 1 to select Ordinary End-Fire
% Input 2 to select Hansen-Woodyard End-Fire
%
% 4.Output
% --------
% I. Normal Mode
% a. Pitch angle alpha (in degrees)
% b. Axial ratio (AR)
% c. HPBW (in degrees)
% c. Directivity (dimensionless and in dB)
% d. Plot the normalized radiation pattern (0 to -60 dB)
%
% II. Axial (End-Fire) Mode
% a. Pitch angle alpha (in degrees)
% b. Input impedance (ohms)
% c. Axial ratio (AR)
% d. Relative phase velocity ratio p
% e. HPBW (in degrees)
%    A. Approximate (use Equation 10-31)
%    B. Numerical (found from pattern)
% f. FNNW (in degrees)
%    A. Approximate (use Equation 10-32)
%    B. Numerical (found from pattern, set "zero" point to be less
%       than 1e-6)
% g. Directivity (dimensionless and in dB)
%    A. Approximate (use Equation 10-33)
%    B. Numerical (calculated from equation D0=4*pi*Umax/Prad)
% h. Plot the normalized radiation pattern (in dB: 0 to -60 dB)
% ------------------------------------------------------------
clear all;
close all;
fprintf('\n   *** NOTICE: This program uses "polar_dB.m" to plot the patterns!\n\n');

%---Choice of output-------------------------------------------
fprintf('Output device option: \n\tOption (1): Screen\n\tOption (2): File \n');
ERR = 1;
while(ERR ~= 0)
  DEVICE = str2num(input('\nOutput device = ','s'));
  if(DEVICE == 1)
    ERR = 0;
  elseif(DEVICE == 2)
    FILNAM = input('Input the desired output filename: ','s');
    ERR = 0;
  else
    fprintf('\nOutputting device number should be either 1 or 2\n');
  end
end

%---Choice of radiation mode--------------
ERR = 1;
while(ERR ~= 0)
   fprintf('\nSELECT RADIATION MODE:\n');
   fprintf('----------------------\n');
   fprintf('(1) NORMAL MODE\n');
   fprintf('(2) AXIAL MODE\n\n');
   SELECT_mode = str2num(input('SELECT = ','s'));
   if or((SELECT_mode == 1), (SELECT_mode == 2))
      ERR = 0;
   end
end

%---Start of main program---

switch SELECT_mode,
   case 1
%---Normal mode----------
   fprintf('\nInput:\n');
   fprintf('--------\n');

%---Input turn numbers------------------------------
   ERR = 1;
   while(ERR ~= 0)
      N = str2num(input('Number of turns N = ','s'));
      N = round(N);
      if (isempty(N))
         fprintf('\n   *** ERROR: Please enter an integer!\n\n');
      elseif (N <= 0)
         fprintf('\n   *** ERROR: The number has to be greater than 0!\n\n');
      elseif (N > 10)
         fprintf('\n   *** WARNING: Large turn number may cause multi-lobe!\n\n');
         ERR = 0;
      else
         ERR = 0;
      end
   end

%---Input circumference of loops-------------------------------------
   ERR = 1;
   while(ERR ~= 0)
      fprintf('\n   *** NOTICE: For Normal mode, C << lambda.');
      fprintf('\n              It is recommended that C/lambda <= 1/10.\n\n');
      C = str2num(input('Circumference of loops C (in lambda) = ','s'));
      if (isempty(C))
```

```
         fprintf('\n   *** ERROR: Please enter an number!\n\n');
      elseif (C <= 0)
         fprintf('\n   *** ERROR: The number has to be greater than 0!\n\n');
      else
         ERR = 0;
      end
   end

%---Input spacing between turns-------------------------------------
   ERR = 1;
   while(ERR ~= 0)
      fprintf('\n   *** NOTICE: For Normal mode, S << lambda.');
      fprintf('\n             It is recommended that S/lambda <= 1/20.\n\n');
      S = str2num(input('Spacing between turns S (in lambda) = ','s'));
      if (isempty(S))
         fprintf('\n   *** ERROR: Please enter a number!\n\n');
      elseif (S <= 0)
         fprintf('\n   *** ERROR: The number has to be greater than 0!\n\n');
      else
         ERR = 0;
      end
   end

%---Normal mode main program----------
%---Setup----------------------------
   step=0.1; % Step: accuracy control
   delta=1e-2; % To avoid sigularity in calculation
   angle=[0+delta:step:360+delta];
   theta=angle*pi/180;

%---Pitch angle-----------
   alpha=atan(S/C)*180/pi;

%---Axial Ratio---
   AR=2*S/C;

%---Pattern-------------------------
   L0=sqrt(C^2+S^2);
   E=sin(theta); % Element factor
   M=2*N; % Image due to ground plane
   kesi=2*pi*(S*cos(theta));
   AF=sin(M*kesi/2)./sin(kesi/2)/N; % Array factor
   U=(E.*AF).^2; % Pattern mutiplicatin
   [Umax,Index]=max(U);
   U=U/Umax;
   Umax=1;
   U_dB=10*log10(U);
```

```
%---HPBW(From pattern)--------
HP_dB=(1/sqrt(2));
for i=Index:length(U)
    if U(i)<HP_dB
        HPBW=angle(i)+angle(i-1)-2*angle(Index);
        break;
    end
end

%---Directivity-----------------------
    integrand=U.*sin(theta)*step*pi/180;
    Prad=2*pi*sum(integrand(1:round(length(integrand)/2)));
    D0=4*pi*Umax/Prad;
    D_dB=10*log10(D0);

%---Create output------------
if(DEVICE == 2)
   fid = fopen(FILNAM,'w');
else
   fid = DEVICE;
end

%---Echo input parameters and output computed parameters---------
fprintf(fid,'\nHELIX: NORMAL MODE:\n');
fprintf(fid,'\nInput parameters:\n-----------------');
fprintf(fid,'\nNumber of turns N = %3.0f',N);
fprintf(fid,'\nCircumference of loops C (in lambda) = %6.4f',C);
fprintf(fid,'\nSpacing between turns S (in lambda) = %6.4f',S);
fprintf(fid,'\nLength (one-turn) L0 (in lambda) = %6.4f',sqrt(C^2+S^2));
fprintf(fid,['\nLength (',num2str(N),'-turn) LN (in lambda) = %6.4f'],N*sqrt(C^2+S^2));

fprintf(fid,'\n\nOutput parameters:\n------------------');
fprintf(fid,'\nPitch angle alpha (in degrees) = %6.4f',alpha);
fprintf(fid,'\nAxial Ratio AR (dimensionless) = %6.4f',AR);
fprintf(fid,'\nHPBW (in degrees) = %6.4f',HPBW);
fprintf(fid,'\nDirectivity(approximate) (dimensionless) = 1.5');
fprintf(fid,'\nDirectivity(approximate) (in dB) = 1.7609');
fprintf(fid,'\nDirectivity(numerical from pattern) (dimensionless) = %6.4f',D0);
fprintf(fid,'\nDirectivity(numerical from pattern) (in dB) \t= %6.4f',D_dB);
fprintf(fid,'\n\n');
if(DEVICE == 2)
   fclose(fid);
end

%---Plot normalized radiation pattern(in dB:-60-0dB)---------------------
polar_dB(angle,U_dB,-60,0,4,'-');
title('Normalized Radiation Pattern of Normal Mode Helical Antenna(in dB)');
```

```
case 2
%---Axial(End-fire) mode
   fprintf('\nInput:\n');
   fprintf('--------\n');

%---Input turn numbers------------------------------
   ERR = 1;
   while(ERR ~= 0)
      N = str2num(input('Number of turns N = ','s'));
      N = round(N);
      if (isempty(N))
         fprintf('\n   *** ERROR: Please enter an integer!\n\n');
      elseif (N <= 0)
         fprintf('\n   *** ERROR: The number has to be greater than 0!\n\n');
      else
         ERR = 0;
      end
   end

%---Input circumference of loops-------------------------------------
   ERR = 1;
   while(ERR ~= 0)
      fprintf('\n   *** NOTICE: To achieve nearly circular polarizaion, it is');
      fprintf('\n               recommended that 3/4 <= (C/lambda) <= 4/3.\n\n');
      C = str2num(input('Circumference of loops C (in lambda) = ','s'));
      if (isempty(C))
         fprintf('\n   *** ERROR: Please enter an number!\n\n');
      elseif (C <= 0)
         fprintf('\n   *** ERROR: The number has to be greater than 0!\n\n');
      else
         ERR = 0;
      end
   end

%---Input spacing between turns-------------------------------------
   ERR = 1;
   while(ERR ~= 0)
      fprintf('\n   *** NOTICE: To achieve nearly circular polarizaion, it is');
      fprintf('\n               recommended that S/lambda is approximately 1/4.\n\n');
      S = str2num(input('Spacing between turns S (in lambda) = ','s'));
      if (isempty(S))
         fprintf('\n   *** ERROR: Please enter an number!\n\n');
      elseif (S <= 0)
         fprintf('\n   *** ERROR: The number has to be greater than 0!\n\n');
      else
         ERR = 0;
      end
   end
```

```
%---Axial(End-fire) mode main program---
%---Setup-------------------------------
    step=0.1; % Step: accuracy control
    delta=1e-2; % To avoid sigularity in calculation
    angle=[0+delta:step:360+delta];
    theta=angle*pi/180;


%---Choice of End-Fire mode--------------
ERR = 1;
while(ERR ~= 0)
    fprintf('\nSELECT END-FIRE MODE:\n');
    fprintf('----------------------\n');
    fprintf('(1) ORDINARY END-FIRE\n');
    fprintf('(2) HANSEN-WOODYARD END-FIRE\n');
    fprintf('(3) END-FIRE (p=1)\n\n');
    SELECT_end = str2num(input('SELECT = ','s'));
    if (SELECT_end== 1)|(SELECT_end == 2)|(SELECT_end == 3)
        ERR = 0;
    end
end

%---Pitch Angle-------
alpha=atan(S/C)*180/pi;

%---Input Impedence---
R=140*C;

%---HPBW(Approximate)----
HPBW_app=52/(C*sqrt(N*S));

%---FNBW(Approximate)----
FNBW_app=115/(C*sqrt(N*S));

%---Directivity(Approximate)---
D_app=15*N*C^2*S;
D_app_dB=10*log10(D_app);

%---Axial Ratio---
AR=(2*N+1)/(2*N);

%---Pattern-----
L0=sqrt(C^2+S^2);
if SELECT_end == 1
%---Ordinary end-fire----------
   p=L0/(S+1);
elseif SELECT_end == 2
```

```
%---Hansen-Woodyard end-fire----
  p=L0/(S+(AR));
else
%---End-fire (p=1)----
  p=1;
end
kesi=2*pi*(S*cos(theta)-L0/p);
U=(sin(pi/(2*N))*cos(theta).*sin(N/2*kesi)./sin(kesi/2)).^2;
Umax=max(U);
U=U/Umax;
Umax=1;
U_dB=10*log10(U);

%---Numerical results from pattern---
if (SELECT_end ~= 3)
%---HPBW(From pattern)--------
imax=1;
HP_dB=(1/sqrt(2));
for i=1:length(U)
   if U(i)<HP_dB
      HPBW=angle(i)+angle(i-1);
      break;
   end
end
%---FNBW(From pattern)---------
for i=1:length(U)
   if U(i)<1e-4 % set "zero" point to be less than 1e-4
      FNBW=angle(i)+angle(i-1);
      break;
   end
end
else
%---HPBW(From pattern)--------
HP_dB=(1/sqrt(2));
[temp,imax]=max(U);
end
for i=imax:-1:2
   if U(i)<HP_dB
      HPBW=angle(imax-i)+angle(imax-i+1);
      break;
   end
end
%---FNBW(From pattern)---------
for i=imax:-1:2
   if U(i)<1e-4 % set "zero" point to be less than 1e-4
      FNBW=angle(imax-i)+angle(imax-i+1);
      break;
   end
```

```
end
%---Directivity(From pattern)------
integrand=U.*sin(theta)*step*pi/180;
Prad=2*pi*sum(integrand(1:round(length(integrand)/2)));
D0=4*pi*Umax/Prad;
D_dB=10*log10(D0);

%---Create output------------
if(DEVICE == 2)
  fid = fopen(FILNAM,'wt');
else
  fid = DEVICE;
end

%---Plot normalized radiation pattern(in dB:-60-0dB)----------------------
polar_dB(angle,U_dB,-60,0,4,'-');
if SELECT_end == 1
   title('Normalized Radiation Pattern of Ordinary End-Fire Helical Antenna(in dB)');
elseif SELECT_end == 2
   title('Normalized Radiation Pattern of Hansen-Woodyard End-Fire Helical Antenna(in dB)');
else
   title('Normalized Radiation Pattern of End-Fire (p=1) Helical Antenna(in dB)');
end

%---Echo input parameters and output computed parameters---------
if SELECT_end == 1
   fprintf(fid,'\n HELIX: ORDINARY END-FIRE MODE:\n');
elseif SELECT_end == 2
   fprintf(fid,'\n HELIX: HANSEN-WOODYARD END-FIRE MODE:\n');
else
   fprintf(fid,'\n HELIX: END-FIRE MODE (p=1):\n');
end
fprintf(fid,'\nInput parameters:\n-----------------');
fprintf(fid,'\nNumber of turns N = %3.0f',N);
fprintf(fid,'\nCircumference of loops C (in lambda) = %6.4f',C);
fprintf(fid,'\nSpacing between turns S (in lambda) = %6.4f',S);
fprintf(fid,'\nLength (one-turn) L0 (in lambda) = %6.4f',sqrt(C^2+S^2));
fprintf(fid,['\nLength (',num2str(N),'-turn) LN (in lambda) = %6.4f'],N*sqrt(C^2+S^2));

if (SELECT_end == 1)|(SELECT_end == 2)|(imax == 1)
fprintf(fid,'\n\nOutput parameters:\n------------------');
fprintf(fid,'\nPitch angle alpha (in degrees) = %6.4f',alpha);
fprintf(fid,'\nInput impedance R (ohms)) = %6.4f',R);
fprintf(fid,'\nAxial Ratio AR (dimesionless) = %6.4f',AR);
fprintf(fid,'\nRelative phase velocity ratio p = %6.4f',p);
fprintf(fid,'\n\nHBPW (in degrees):');
fprintf(fid,'\n   A. Approximate(10-31) = %6.4f',HPBW_app);
fprintf(fid,'\n   B. Numerical(from pattern) = %6.4f',HPBW);
fprintf(fid,'\n\nFNBW(in degrees):');
fprintf(fid,'\n   A. Approximate(10-32) = %6.4f',FNBW_app);
fprintf(fid,'\n   B. Numerical(from pattern) = %6.4f',FNBW);
fprintf(fid,'\n\nDirectivity:');
fprintf(fid,'\n   A1. Approximate(10-33) (dimensionless) = %6.4f',D_app);
fprintf(fid,'\n   A2. Approximate(10-33) (in dB) = %6.4f',D_app_dB);
fprintf(fid,'\n   B1. Numerical(from pattern) (dimensionless) = %6.4f',D0);
fprintf(fid,'\n   B2. Numerical(from pattern) (in dB) = %6.4f',D_dB);
fprintf(fid,'\n\n');
else
    fprintf(fid,'\n\n-------------------------------------------------------------------------------\n');
    fprintf(fid,'BAD DESIGN! MAXIMUM IS NOT AT 0 DEGREES!\n');
    fprintf(fid,'PLEASE SEE THE PLOTTED RADIATION PATTERN FOR DETAILS.\n');
end
if(DEVICE == 2)
   fclose(fid);
end
end

%---End of main program---
```