Table of Contents

CSUATR_v2 Program Data Interpretation Patch Array

This program takes will display a dataset wtth the following assumptions: All data points have the same S-Parameter Data points were obtained in a grid-pattern Data points were obtained in a zig-zag fashion (down, right, up, right..) The x-axis had the greater number of movements Each element of array

```
close all

% Allows user to choose if they want to seperate plots or a subplot
seperatePlotEnable = 1;
subplotEnable = 1;

% 26 - 7 GHz
% 51 - 8 GHz
% 76 - 9 GHz
% 101 - 10 GHz
% 126 - 11 GHz
% 151 - 12 GHz
% 176 - 13 GHz
freqIndex = 26; % Frequency index for 8 GHz
dataFile = strings(1,5);

% Data path and file names of each scan
dataPath = "C:\Users\ajhgo\Desktop\ATR\PatchReadingMatlab\"; % Complete path
 to your data
dataFile(1) = "PatchArray_Reading1_Ele1.txt"; % Name of the data file
dataFile(2) = "PatchArray_Reading1_Ele2.txt"; % Name of the data file
dataFile(3) = "PatchArray_Reading1_Ele3.txt"; % Name of the data file
dataFile(4) = "PatchArray_Reading1_Ele4.txt"; % Name of the data file
```

```
dataFile(5) = "PatchArray_Reading1_Ele5.txt"; % Name of the data file

% Creates array of data structs
for i = 1:5
    dataStructs(i) = ReadCSUATRData(dataPath + dataFile(i));
end

% Create empty arrays for real and imag
dataRe_ind = zeros(length(dataStructs(1).real(:,freqIndex)),5);
dataIm_ind = zeros(length(dataStructs(1).real(:,freqIndex)),5);
dataRe_tot = zeros(length(dataStructs(1).real(:,freqIndex)),1);
dataIm_tot = zeros(length(dataStructs(1).real(:,freqIndex)),1);

% Retrieve real and imag parts of each scan
for i = 1:5
    % Superposing the real and imag parts on top of eachother
    dataRe_tot = dataRe_tot +  dataStructs(i).real(:,freqIndex);
    dataIm_tot = dataIm_tot +  dataStructs(i).imaginary(:,freqIndex);

    % Indepenent readings from each scan
    dataRe_ind(:,i) = dataStructs(i).real(:,freqIndex);
    dataIm_ind(:,i) = dataStructs(i).imaginary(:,freqIndex);

%     dataAbs(:,i) = sqrt(dataRe(:,i).^2 + dataIm(:,i).^2);
%     dataAbs_sum = dataAbs_sum + dataAbs(:,i);
end

% Calculating the magnitude
dataMag_tot = sqrt(dataRe_tot.^2 + dataIm_tot.^2);
dataMag_int = sqrt(dataRe_ind.^2 + dataIm_ind.^2);

% Azimuth axis from -90 deg to 90 deg
azi = linspace(-pi/2,pi/2,length(dataRe_tot));

dataMag_tot_dB = 10.*log10(dataMag_tot);
% dataMag_int_dB = 10.*log10(dataMag_int);
% % dataMag_tot_dB = dataMag_tot_dB - max(dataMag_tot_dB);
% dataMag_tot_dB(dataMag_tot_dB<-40) = -40;
% h = polarplot(azi,dataMag_tot_dB+40,'Linewidth',1,'color',[.21 .81 .94]);
% haxes = get(h,'Parent');
% haxes.RTickLabel = {'-40','-30','-20','-10','0'};

% Polar_dB(azi,dataMag_tot_dB,[-70 -10],6)
% Polar_dB(azi,dataMag_int_dB(:,4),[-70 -20],6)

if (subplotEnable == 1)
    sgtitle('Readings at 8GHz')
    % Plot element 1
    subplot(3,2,1)
    polarplot(azi,smooth(dataMag_int(:,1)));
    title('Radiation Pattern from Element 1')

    % Plot element 2
    subplot(3,2,2)
```

```
    polarplot(azi,smooth(dataMag_int(:,2)));
    title('Radiation Pattern from Element 2')

    % Plot element 3
    subplot(3,2,3)
    polarplot(azi,smooth(dataMag_int(:,3)));
    title('Radiation Pattern from Element 3')

    % Plot element 4
    subplot(3,2,4)
    polarplot(azi,smooth(dataMag_int(:,4)));
    title('Radiation Pattern from Element 4')

    % Plot element 5
    subplot(3,2,5)
    polarplot(azi,smooth(dataMag_int(:,5)));
    title('Radiation Pattern from Element 5')

    % Plot total radiation pattern
    subplot(3,2,6)
    polarplot(azi,smooth(dataMag_tot));
    title('Total Radiation Pattern')
end

if (seperatePlotEnable == 1)
    % Plot element 1
    figure()
    polarplot(azi,smooth(dataMag_int(:,1)));
    title('Radiation Pattern from Element 1')

    % Plot element 2
    figure()
    polarplot(azi,smooth(dataMag_int(:,2)));
    title('Radiation Pattern from Element 2')

    % Plot element 3
    figure()
    polarplot(azi,smooth(dataMag_int(:,3)));
    title('Radiation Pattern from Element 3')

    % Plot element 4
    figure()
    polarplot(azi,smooth(dataMag_int(:,4)));
    title('Radiation Pattern from Element 4')

    % Plot element 5
    figure()
    polarplot(azi,smooth(dataMag_int(:,5)));
    title('Radiation Pattern from Element 5')

    % Plot total radiation pattern
    figure()
    polarplot(azi,smooth(dataMag_tot));
    title('Total Radiation Pattern')
```

```
end

% dataAbs_sum = zeros(length(dataStructs(1).real(:,freqIndex)),1);

% dataTest1 = dataRe(:,i);
% dataAbs_sum_dB = transpose(dataAbs_sum);
% dataAbs_dB = transpose(dataAbs(:,i));
% azi = linspace(-pi/2,pi/2,length(dataAbs_sum));
%
% polarplot(azi,smooth(dataAbs_sum_dB));
% hold on
% polarplot(azi,smooth(dataAbs_dB));
```

Retrieve Data

```
function data = ReadCSUATRData(filename)
fprintf('Reading file: %s\n', filename)
data.id = [];

data.sParameter = []; %data.mode = [];
data.sourcePower = []; %data.power = [];
data.ifBandwidth = [];
data.averagePoints = []; %data.avgpoints = [];

data.horizontal = [];
data.vertical = [];
data.depth = [];
data.azimuth = [];
data.elevation = [];
data.polarization = [];

data.frequency = [[]];
data.real = [[]];
data.imaginary = [[]];

% read file
fileID = fopen(filename,'r'); % opening file
count = 0;
tline = fgetl(fileID); % get first data point
while ischar(tline)
    %fprintf("line: %d, value: %s\n", count, tline)
    if (rem(count, 4) == 0)
        values = textscan(tline, "%d %s %f %f %d %f %f %f %f %f %f");

        data.id = [data.id, values{1}];

        data.sParameter = [data.sParameter, values{2}];
        data.sourcePower = [data.sourcePower, values{3}];
        data.ifBandwidth = [data.ifBandwidth, values{4}];
        data.averagePoints = [data.averagePoints, values{5}];

        data.horizontal = [data.horizontal, values{6}];
        data.vertical = [data.vertical, values{7}];
```

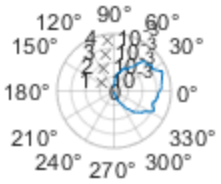
```
        data.depth = [data.depth, values{8}];
        data.azimuth = [data.azimuth, values{9}];
        data.elevation = [data.elevation, values{10}];
        data.polarization = [data.polarization, values{11}];

    elseif (rem(count, 4) == 1)
        values = transpose(cell2mat(textscan(tline, "%f")));
        data.frequency = [data.frequency; values];
    elseif (rem(count, 4) == 2)
        values = transpose(cell2mat(textscan(tline, "%f")));
        data.real = [data.real; values];
    elseif (rem(count, 4) == 3)
        values = transpose(cell2mat(textscan(tline, "%f")));
        data.imaginary = [data.imaginary; values];
    end
    tline = fgetl(fileID); % get next data point
    count = count + 1;
end
fclose(fileID); % closing file
end
```

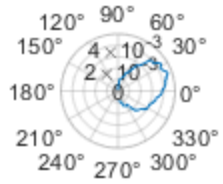
```
Reading file: C:\Users\ajhgo\Desktop\ATR\PatchReadingMatlab
\PatchArray_Reading1_Ele1.txt
Reading file: C:\Users\ajhgo\Desktop\ATR\PatchReadingMatlab
\PatchArray_Reading1_Ele2.txt
Reading file: C:\Users\ajhgo\Desktop\ATR\PatchReadingMatlab
\PatchArray_Reading1_Ele3.txt
Reading file: C:\Users\ajhgo\Desktop\ATR\PatchReadingMatlab
\PatchArray_Reading1_Ele4.txt
Reading file: C:\Users\ajhgo\Desktop\ATR\PatchReadingMatlab
\PatchArray_Reading1_Ele5.txt
```

Readings at 8GHz

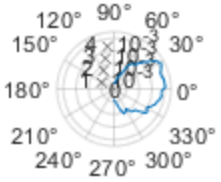
Radiation Pattern from Element 1



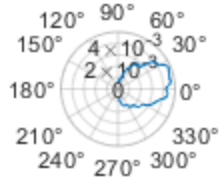
Radiation Pattern from Element 2



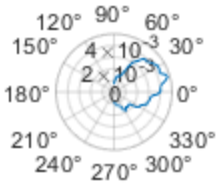
Radiation Pattern from Element 3



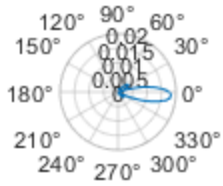
Radiation Pattern from Element 4



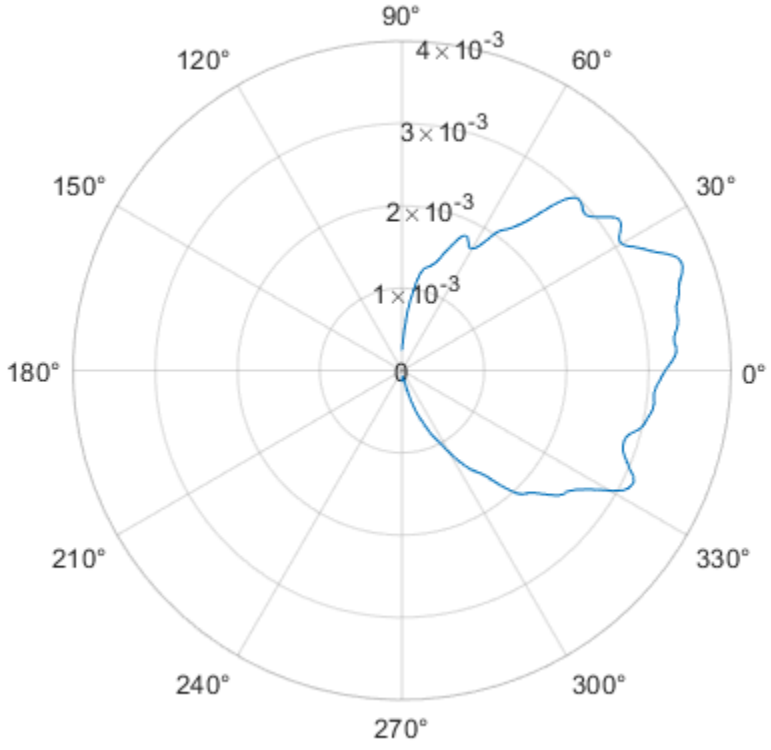
Radiation Pattern from Element 5

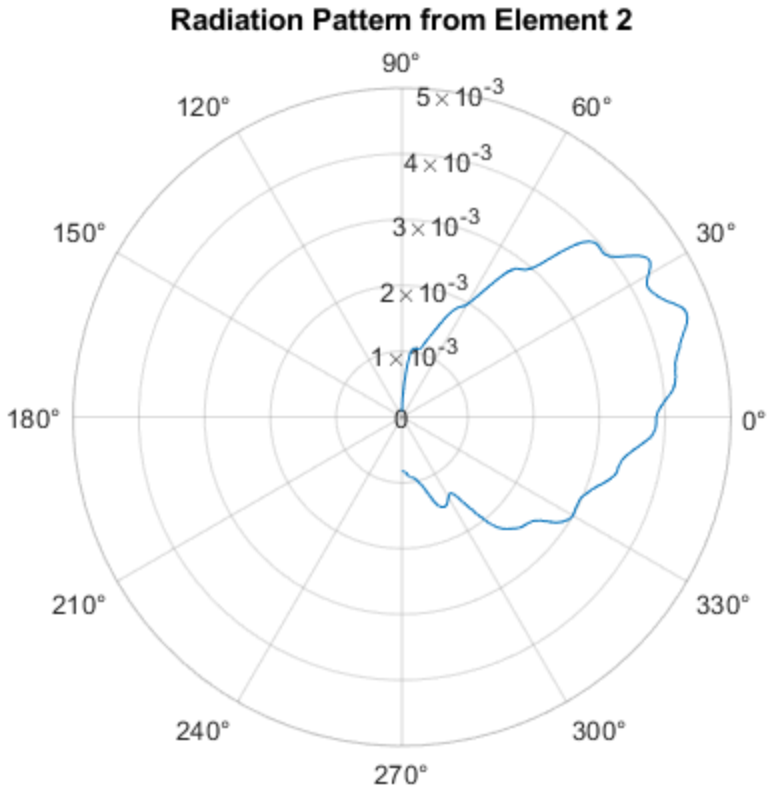


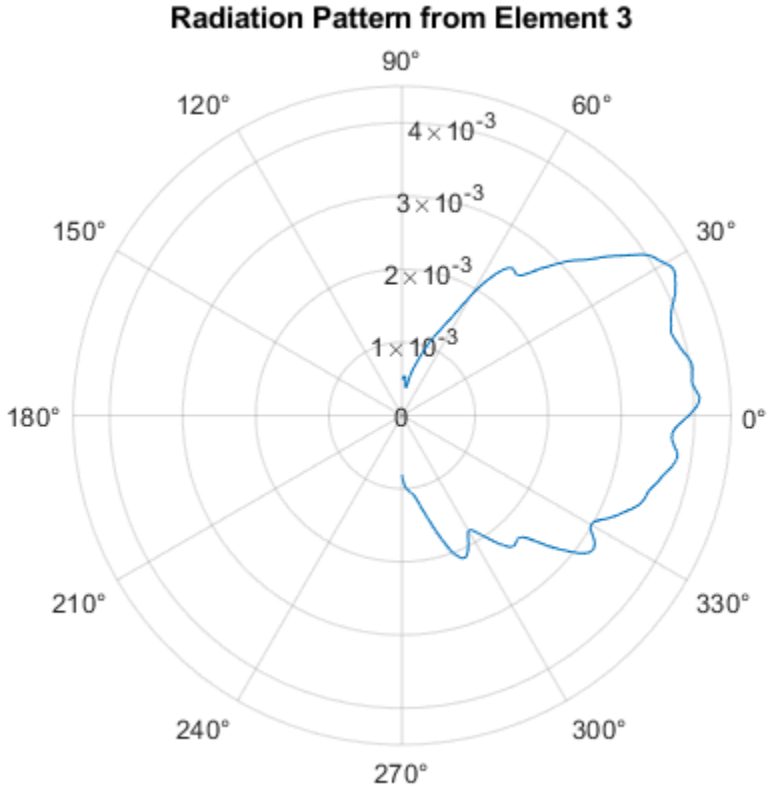
Total Radiation Pattern

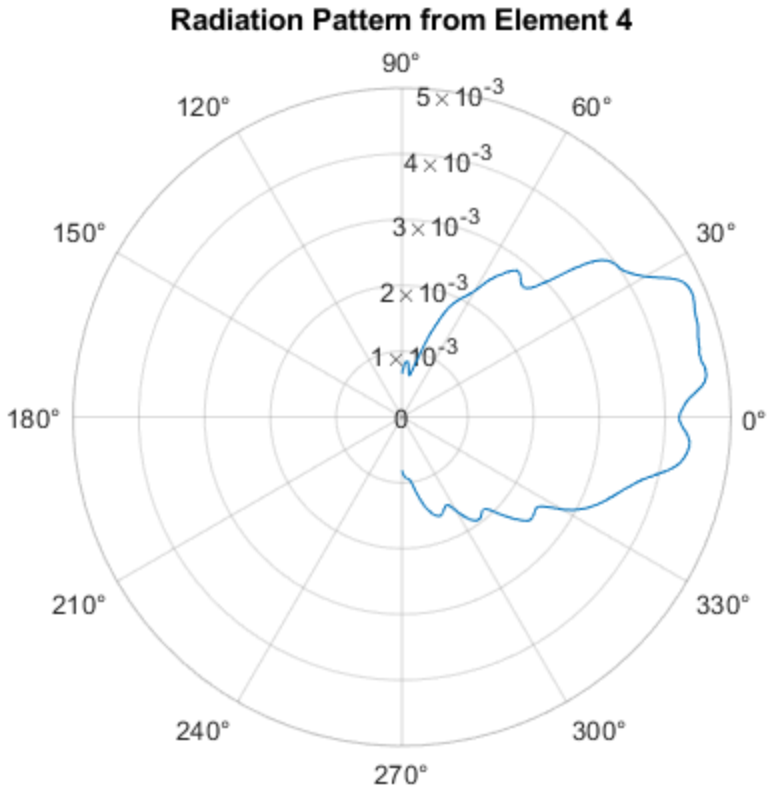


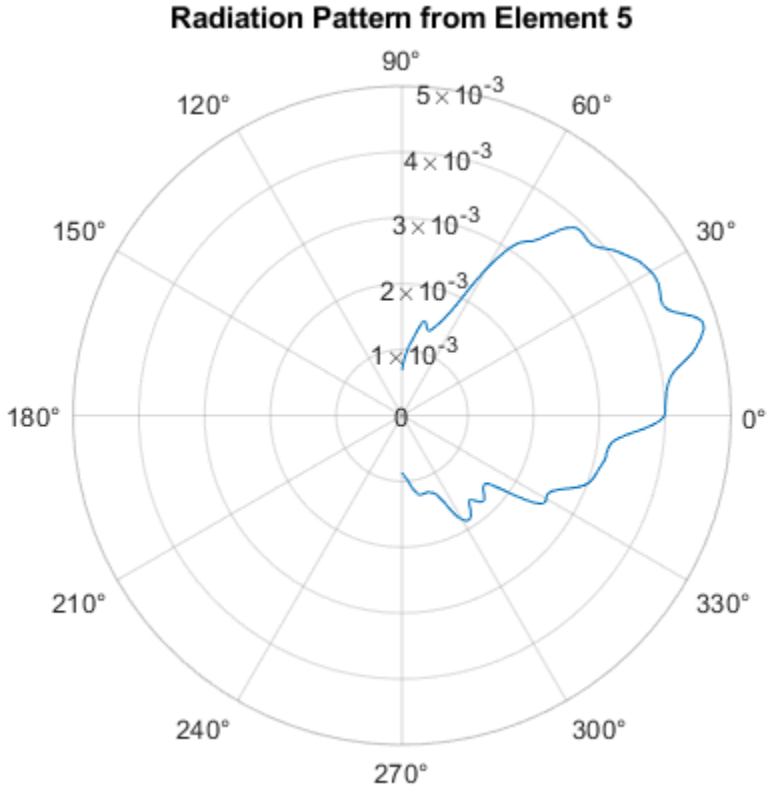
Radiation Pattern from Element 1

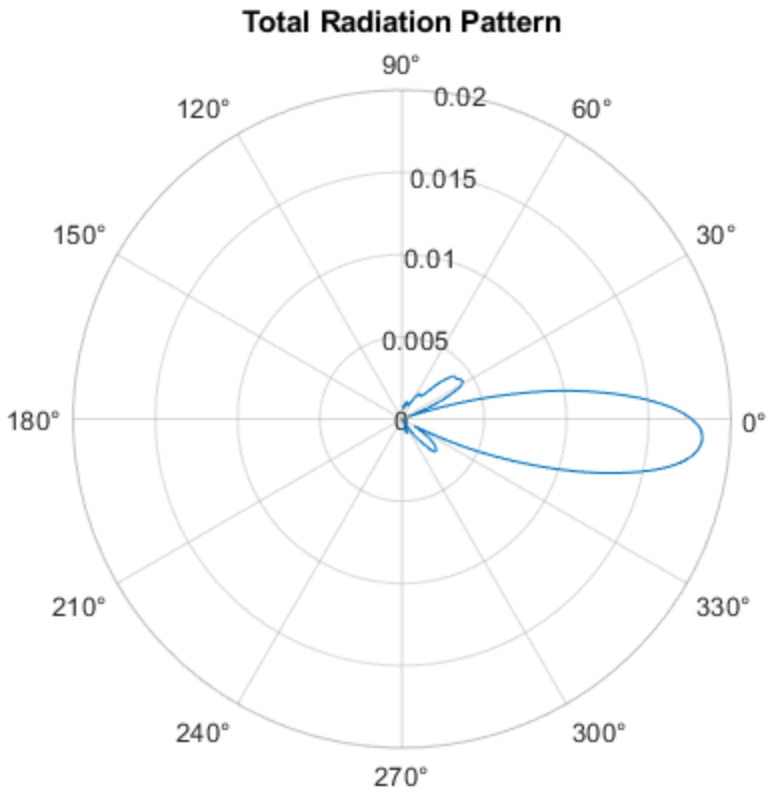



Radiation Pattern from Element 2
90°
120°
60°
150°
30°
180°
0°
210°
330°
240°
300°
270°
5×10-3
4×10-3
3×10-3
2×10-3
1×10-3
0


Radiation Pattern from Element 3
90°
120°
60°
150°
30°
180°
0°
210°
330°
240°
300°
270°
4×10-3
3×10-3
2×10-3
1×10-3
0


Radiation Pattern from Element 4
90°
120°
60°
150°
30°
180°
0°
210°
330°
240°
300°
270°
5×10-3
4×10-3
3×10-3
2×10-3
1×10-3
0


Radiation Pattern from Element 5
90°
120°
60°
150°
30°
180°
0°
210°
330°
240°
300°
270°
5×10-3
4×10-3
3×10-3
2×10-3
1×10-3
0



Plot Data

```
function PlotCartesian(data, zAxis, yAxis, xAxis, xAxisWidth, frequencyIndex)
```

compensate (removes excess that does not compose a grid pattern)

```
len = length(data.id) - rem(length(data.id), xAxisWidth);
```

compute x axis

```
x = [];
xLabel = 'None';
if (strcmp(xAxis, 'Polarization') == 1)
    xLabel = 'Polarization (degrees)';
    x = data.polarization(1:len);
elseif (strcmp(xAxis, 'Vertical') == 1)
    xLabel = 'Vertical (cm)';
    x = data.vertical(1:len);
elseif (strcmp(xAxis, 'Horizontal') == 1)
    xLabel = 'Horizontal (cm)';
    x = data.horizontal(1:len);
elseif (strcmp(xAxis, 'Depth') == 1)
```

```
    xLabel = 'Depth (cm)';
    x = data.depth(1:len);
elseif (strcmp(xAxis, 'Azimuth') == 1)
    xLabel = 'Azimuth (degrees)';
    x = data.azimuth(1:len);
elseif (strcmp(xAxis, 'Elevation') == 1)
    xLabel = 'Elevation (degrees)';
    x = data.elevation(1:len);
else
    return;
end
x = reshape(x, xAxisWidth, []);
x(:,2:2:end) = flipud(x(:,2:2:end));
X = transpose(x(:,1)); % get x values
```

compute y axis

```
y = [];
yLabel = ' ';
if (strcmp(yAxis, 'Polarization') == 1)
    yLabel = 'Polarization (degrees)';
    y = data.polarization(1:len);
elseif (strcmp(yAxis, 'Vertical') == 1)
    yLabel = 'Vertical (cm)';
    y = data.vertical(1:len);
elseif (strcmp(yAxis, 'Horizontal') == 1)
    yLabel = 'Horizontal (cm)';
    y = data.horizontal(1:len);
elseif (strcmp(yAxis, 'Depth') == 1)
    yLabel = 'Depth (cm)';
    y = data.depth(1:len);
elseif (strcmp(yAxis, 'Azimuth') == 1)
    yLabel = 'Azimuth (degrees)';
    y = data.azimuth(1:len);
elseif (strcmp(yAxis, 'Elevation') == 1)
    yLabel = 'Elevation (degrees)';
    y = data.elevation(1:len);
else
    return;
end
y = reshape(y, xAxisWidth, []);
y(:,2:2:end) = flipud(y(:,2:2:end));
Y = y(1,:); % get y values
```

compute z matrix

```
z = [];
if (strcmp(zAxis, 'Real') == 1)
    type = 'Real';
    z = transpose(data.real(1:len,frequencyIndex));
elseif (strcmp(zAxis, 'Imaginary') == 1)
    type = 'Imaginary';
    z = transpose(data.imaginary(1:len,frequencyIndex));
```

```
elseif (strcmp(zAxis, 'Magnitude (dB)') == 1)
    type = 'Magnitude';
    zr = transpose(data.real(1:len,frequencyIndex));
    zi = transpose(data.imaginary(1:len,frequencyIndex));
    z = 20*log10(sqrt(zr.*zr + zi.*zi));
elseif (strcmp(zAxis, 'Phase') == 1)
    type = 'Phase';
    zr = transpose(data.real(1:len,frequencyIndex));
    zi = transpose(data.imaginary(1:len,frequencyIndex));
    z = (180 / pi) * atan2(zi, zr);
elseif (strcmp(zAxis, 'Amplitude (V)') == 1)
    type = 'Amplitude';
    zr = transpose(data.real(1:len,frequencyIndex));
    zi = transpose(data.imaginary(1:len,frequencyIndex));
    z = sqrt(zr.*zr + zi.*zi);
else
    return;
end
z = reshape(z, [], xAxisWidth);
z(:,2:2:end) = flipud(z(:,2:2:end));
Z = z; % get z values
```

compute frequency

```
f = data.frequency(1, frequencyIndex); % get frequency
F = f / 1000000; % compute frequency in MHz
```

display data

```
figure()
surf(X, Y, Z)
% contour(X, Y, Z)
% mesh(X,Y,Z)
```

annotate plot

```
title(sprintf('Plot: %s   Index: %d   Frequency: %fMHz ', type,
 frequencyIndex, F))
xlabel(xLabel)
ylabel(yLabel)
zlabel(zAxis)

end

function PlotAzimuthElevationSpherical(data, dataType, elevation, azimuth,
 xAxisWidth, frequencyIndex)
```

compensate (removes excess that does not compose a grid pattern)

```
len = length(data.id) - rem(length(data.id), xAxisWidth);
```

compute x axis

```
x = data.azimuth(1:len);
x = reshape(x, xAxisWidth, []);
x(:,2:2:end) = flipud(x(:,2:2:end));
X = transpose(x(:,1)); % get x values
```

compute y axis

```
y = data.elevation(1:len);
y = reshape(y, xAxisWidth, []);
y(:,2:2:end) = flipud(y(:,2:2:end));
Y = y(1,:); % get y values
```

compute z matrix

```
z = [];
if (strcmp(dataType, 'Real') == 1)
    z = transpose(data.real(1:len,frequencyIndex));
elseif (strcmp(dataType, 'Imaginary') == 1)
    z = transpose(data.imaginary(1:len,frequencyIndex));
elseif (strcmp(dataType, 'Magnitude') == 1)
    zr = transpose(data.real(1:len,frequencyIndex));
    zi = transpose(data.imaginary(1:len,frequencyIndex));
    z = 20*log10(sqrt(zr.*zr + zi.*zi));
elseif (strcmp(dataType, 'Phase') == 1)
    zr = transpose(data.real(1:len,frequencyIndex));
    zi = transpose(data.imaginary(1:len,frequencyIndex));
    z = (180 / pi) * atan2(zi, zr);
elseif (strcmp(dataType, 'Amplitude') == 1)
    zr = transpose(data.real(1:len,frequencyIndex));
    zi = transpose(data.imaginary(1:len,frequencyIndex));
    z = sqrt(zr.*zr + zi.*zi);
else
    return;
end
z = reshape(z, [], xAxisWidth);
z(:,2:2:end) = flipud(z(:,2:2:end));
Z = z; % get z values
```

compute frequency

```
f = data.frequency(1, frequencyIndex); % get frequency
F = f / 1000000; % compute frequency in MHz
```

display data

```
X = deg2rad(x);
Y = linspace(-pi*0.5, pi*0.5, xAxisWidth); %deg2rad(y);
[X,Y,Z] = sph2cart(X,y,Z.');
```

```
% Z=R*sin(Phi);
% X=R*cos(Phi).*cos(Theta);
% Y=R*cos(Phi).*sin(Theta);

figure()
surf(X, Y, Z)
% contour(X, Y, Z)
% mesh(X,Y,Z)
```

annotate plot

```
title(sprintf('Plot: %s   Index: %d   Frequency: %fMHz ', dataType,
 frequencyIndex, F))
xlabel("Z")
ylabel("X")
zlabel("Y")

end
```

Published with MATLAB® R2022b