Portland State University

ECE 413

Capstone

EM Characterization of Radar Absorbing Materials

Authors:
Kirk Jungles
Chris Toner
Jose Alvarez
David Eding

Sponsor:
Tangitek

May 31, 2020

"Knowledge is Good"

Contents

1 Radar Cross Section Measurement

Radar Cross Section(RCS) is a measure of the effective area of an object under the view of a radar system. It is dependent on a variety of factors, including the geometry of the object under test and the material the object is composed of. Tangitek has developed a material that can be placed on the exterior of various objects that would decrease their effective RCS and reflectivity without making alterations to their surface area and geometry, thereby reducing their visibility on radar systems.

Here, we propose a measurement system to determine the RCS of a target object using the S-parameter data gathered from two experimental setups. Ideally, both of these experiments will be performed in an anechoic chamber.

The first experiment will be the measurement of $S21$ for a pair of antennas separated by a measured distance R_1 for a specified frequency range. The next experiment will be the measurement of $S21$ for the same pair of antennas, this time approximately co-located, pointing at the target object at the previous distance R_1 for the same frequency range as in the first experiment.

After the experimental data is gathered, it will be processed by a MATLAB script that will return the RCS(σ) as a function of frequency. The experimental RCS values will then be compared to simulated RCS values as calculated by ANSYS HFSS.

To perform these experiments, one will need the following items:

- Network analyzer
- Pair of antennas
- Target object
- Antenna and target object stands suitable for EM testing environment(nonconductive)

1.1 Tx-Rx Gain Experiment

The purpose of this first experiment is to determine the experimental gain coefficients of the transmitting and receiving antennas that will be used to solve for σ from the data collected from the next experiment. We begin by noting the general case described by Frii's Equation[1], which is shown in Figure 1.

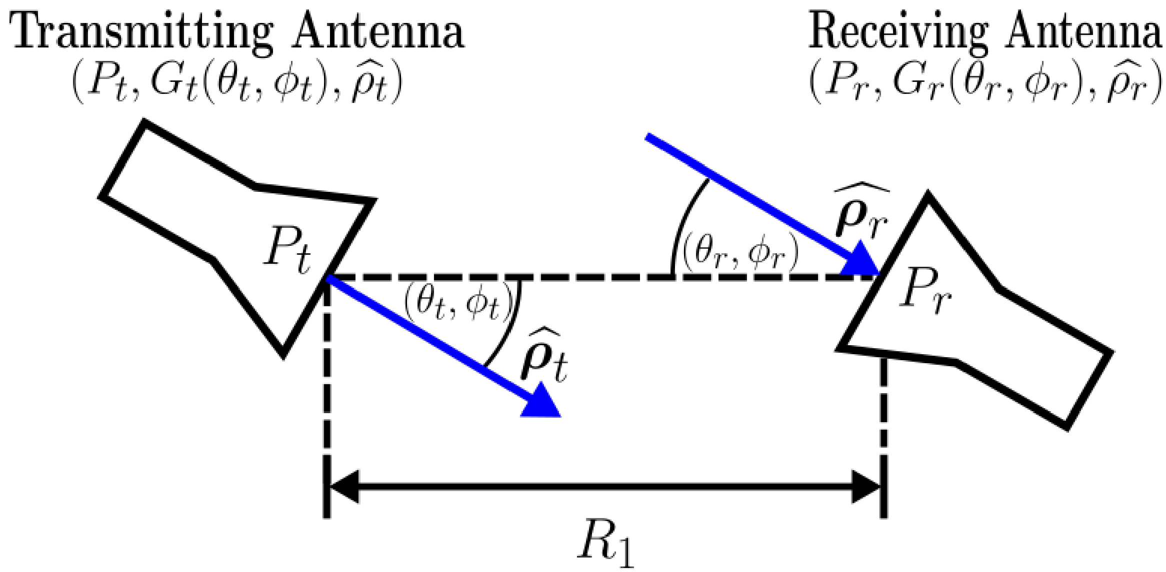


Figure 1: Tx-Rx Experimental Setup

Note that in Figure 1, P_t is the power transmitted by the transmitting antenna and P_r is the power received by the receiving antenna. Frii's Equation then gives us an analytical solution for P_r as

$$P_r = P_t \frac{G_t(\theta_t, \phi_t) G_r(\theta_r, \phi_r) \lambda^2}{(4\pi)^2 R_1^2} |\widehat{\rho_\mathbf{t}} \cdot \widehat{\rho_\mathbf{r}}| \tag{1}$$

For our experiment, we will be placing the two antennas on their respective stands at opposite sides of an anechoic chamber at a measured distance R_1[1]. The antennas will be directly facing each other such that $\theta_t = \phi_t = \theta_t = \phi_t = 0$. Then Equation 1 can be simplified as

$$P_r = P_t \frac{G_t G_r \lambda^2}{(4\pi)^2 R_1^2}$$

Dividing through by P_t yields $\frac{P_r}{P_t}$, then we can use the relationship $|S21|^2 = \frac{P_r}{P_t}$ to find the antenna pair gain coefficients.

$$|S21_{frii}|^2 = \frac{P_r}{P_t} = \frac{G_t G_r \lambda^2}{(4\pi)^2 R_1^2} \tag{2}$$

After successfully measuring $S21_{frii}$ in this experiment and saving our data, we may proceed to the next experiment[2].

[1] Ensure that the length R_1 has been properly measured before proceeding to the next experiment.

[2] G_t, G_r, and λ are all frequency dependent variables. Pay careful attention that the network analyzer performs the same frequency sweep for $S21_{frii}$ as in the next experiment to ensure accuracy of the final result.

1.2 Bistatic RCS Experiment

We begin this experiment by noting the general case described by the Radar Equation[1], which is shown in Figure 2.

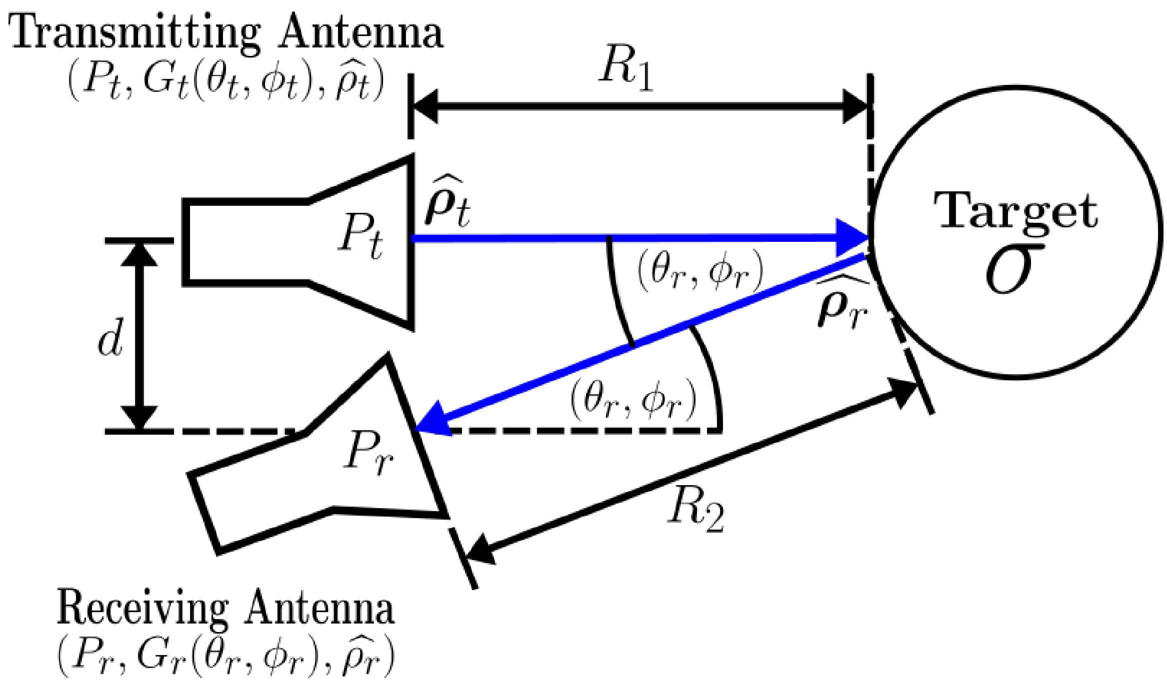


Figure 2: Radar Experimental Setup

In this general case, the Radar Equation gives us an analytical solution for P_r as

$$P_r = P_t \frac{G_t(\theta_t, \phi_t) G_r(\theta_r, \phi_r) \lambda^2}{(4\pi)^2 R_1^2} \frac{1}{4\pi R_2^2} |\widehat{\rho}_{\mathbf{t}} \cdot \widehat{\rho}_{\mathbf{r}}| \sigma \tag{3}$$

For our experiment, we will set the transmitting and receiving antenna as close together as possible and facing approximately the same direction, and the target under test will be set apart from the transmitting antenna the same distance R_1 as in the previous experiment used to measure $S21_{frii}$. Then with $R_1 >> d$, $R_2 \to R_1$, and $\theta_t, \phi_t, \theta_r, \phi_r \to 0$. Then we may use the simplified Radar Equation, which gives us P_r as

$$P_r = P_t \frac{G_t G_r \lambda^2}{(4\pi)^2 R_1^2} \frac{\sigma}{4\pi R_1^2}$$

Dividing through by P_t yields $\frac{P_r}{P_t}$,then we can use the relationship $|S21|^2 = \frac{P_r}{P_t}$.

$$|S21_{rad}|^2 = \frac{P_r}{P_t} = \frac{G_t G_r \lambda^2}{(4\pi)^2 R_1^2} \frac{\sigma}{4\pi R_1^2} \tag{4}$$

After successfully measuring $S21_{frii}$ in this experiment and saving our data, we can then solve for the RCS(σ) of the target[3].

1.3 Analytical Solution for Radar Cross Section

Having collected the data from the previous experiments, we begin with an analytical approach. We note that when the frequencies, antennas, and distance R_1 are the same in both the Tx-Rx and Radar experiments, we can divide Equation 4 by Equation 2 yielding

$$\frac{|S21_{rad}|}{|S21_{frii}|}^2 = \frac{\sigma}{4\pi R_1^2}$$

Solving for σ and noting the frequency dependence, we can then write

$$\sigma(f) = 4\pi R_1^2 \frac{|S21_{rad}(f)|^2}{|S21_{frii}(f)|^2} \tag{5}$$

Having arrived at an analytical solution for σ from our collected data, we can then begin to process the data in MATLAB to produce meaningful results.

1.4 RCS Extraction Script

After collecting the data from the previous experiments, we may use a simple MATLAB script to extract RCS as a function of frequency using the results from Equation 5.

The script follows a simple algorithm, which is outlined as follows:

- Accepts distance R_1 in meters
- Accepts filename and filepath of $S21_{frii}$, $S21_{rad}$, and RCS output files
- Reads frequency and S21 data for each file
- Checks that frequency arrays are identical for both, else ERROR
- Calculates RCS for each frequency using Equation 5
- Plots RCS vs frequency
- Writes RCS data and corresponding frequency array onto new csv file

The resulting RCS and paired frequency csv file will be saved for later use and imported by the RCS Comparison script to compare the experimental RCS with the HFSS simulated RCS.

The MATLAB code that performs this action is uploaded in the team Github account as S21_to_RCS.m and is embedded as follows:

[3]Before proceeding, be sure that both the distance R_1 and the frequencies used to collect $S21_{rad}$ are the same as those used in the Tx-Rx experiment to collect $S21_{frii}$

```
%Kirk Jungles
%S21_to_RCS.m
%4/27/2020
%
%Program accepts S21_frii measurements from Tx-Rx Experiment
%and accepts S21_rad Measurements from Radar Experiment
%Calculates RCS(sigma) as function of frequency

clc, clear, close all

%% User Input
%Experimental Parameters described in RCS document
R1 = 5 %Measured R1 for both experiments (meters)

%Accept S21 CSV from Tx-Rx setup
fname_frii = '2-40GHz-S21_frii.csv' %filename of S21_frii file
fpath_frii = '\\thoth.cecs.pdx.edu\Home03\kjungles\My Documents\MATLAB\Capstone\' %Folder where frii file is stored

%Accept CSV from Radar setup
fname_rad = '2-40GHz-S21_rad.csv' %filename of S21_rad file
fpath_rad = '\\thoth.cecs.pdx.edu\Home03\kjungles\My Documents\MATLAB\Capstone\' %Folder where radar file is stored

%Output filename and path
fname_out = '2-40GHz-RCS.csv' %filename of RCS output file
fpath_out = '\\thoth.cecs.pdx.edu\Home03\kjungles\My Documents\MATLAB\Capstone\' %Folder where radar file is stored

fpath_name_out = [fpath_out fname_out]

%% Extract data from files

%Assumes files are in csv format as vertical vectors [frequency, S21]
%May need changed
%If filetypes not CSV, do something similar(TBA)

%Otherwise, if as assumed, do:
```

```
%%%%frii data%%%%
%Read frii S21 and drequency data from csv file
fpath_name_frii = [fpath_frii fname_frii] %concatenate file
    path and file name
file_data = csvread(fpath_name_frii, 1,0) %Read data
   starting at Row offset = 1, Column offset = 0; omits
   text header

%Store frii data in respective frequency and RCS vectors
freq_frii = file_data(:,1)
S21_frii = file_data(:,2)

%%%%rad data%%%%
%Read S21_rad and drequency data from csv file
fpath_name_rad = [fpath_rad fname_rad] %concatenate file
   path and file name
file_data = csvread(fpath_name_rad, 1,0) %Read data
   starting at Row offset = 1, Column offset = 0; omits
   text header

%Store rad data in respective frequency and RCS vectors
freq_rad = file_data(:,1)
S21_rad = file_data(:,2)

%% Verify that frequency vectors are identical

%Check if frequency vectors same length, else error message
if length(freq_rad) ~= length(freq_frii)
    error('Frequency vectors not same length! Check data
       and consider repeating experiment.')
end

%Check if frequency vectors identical, else error message
for k = [1:length(freq_rad)]
    if freq_rad(k) ~= freq_frii(k)
        error('Frequency vector elements not matched! Check
           data and consider repeating experiment')
    end
end

%If network analyzer simply can't reproduce identical
   frequency sweeps for
%some reason, comment out previous error checks and
   interpolate the S21_rad
```

```
%array to match the S21_frii frequency components(RARE/Unexpected)

%S21_rad = interp1(freq_rad, S21_rad, freq_frii) %y_new = interp1(old x vector, old y vector, new x vector)

%% Calculate sigma as function of frequency

%Using equation for RCS(sigma) in RCS document, calculate sigma from data

sigma = 4*pi*R1^2*(S21_rad./S21_frii) %Units in meter^2
%A./B divides elementwise: [A(1)/B(1), A(2)/B(2), ... ]


%% Export results to CSV

%format will be same as expected csv files with 'frequency(MHz)' and 'RCS(m^2)' headers on first line
%and vertical frequency and RCS vectors:
%[frequency, RCS]
header = ['frequency(MHz)', 'RCS(m^2)']
data = [freq_frii, sigma]

%Concatenate header and data to single matrix
CSV_matrix = [header; data]

%Write matrix to user-specified filepath\filename
writematrix(CSV_matric, fpath_name_out)

fprintf('RCS data successfully written to %s \n', fpath_name_out)
```

2 Radar Cross Section Simulation in Ansys HFSS

2.1 Simulation Instructions

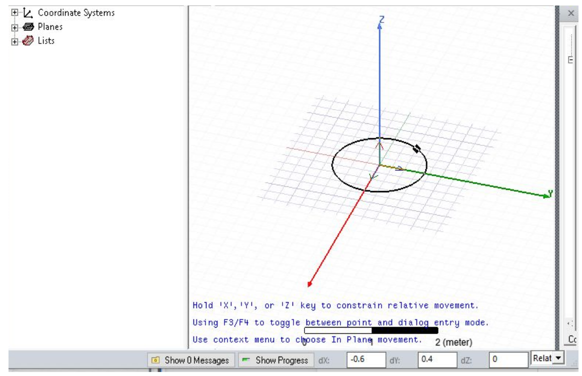
Monostatic and Bistatic RCS simulations can be performed in ANSYS HFSS. The geometry of the RCS targets and experiments can be adjusted for any experimental setup. What follows is a set of instructions that performs monostatic RCS simulations of a copper sphere using ANSYS Electronics Desktop 2018.1. The original powerpoint is also available in the team Github repository under the title `Simulation_Instructions.pdf`.

ANSYS HFSS RCS Simulation Instructions

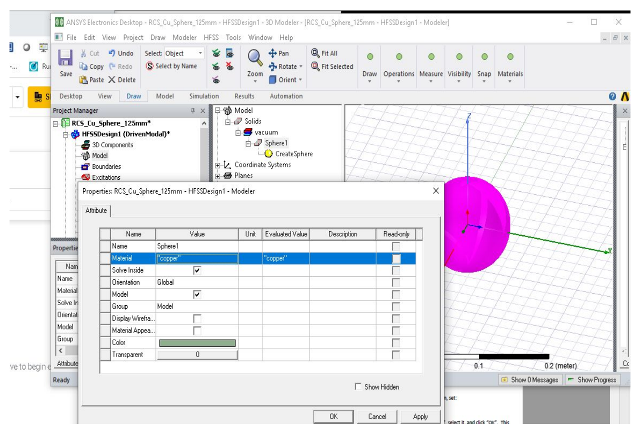
- Launch ANSYS HFSS from Start Menu or Icon
- Set Tool Options
  - On top menu bar, select Tools > Options > General Options
    - HFSS > Boundary Assignment
      - Check boxes that say “Use Wizards for data input when creating new boundaries” and “Duplicate Boundaries/mesh operations with geometry”
    - 3D Modeler > Drawing
      - Chex boxes that say “Edit properties of new primitives”

- Create and Save HFSS Design
  - On top menu bar, select **Project > Insert HFSS Design**
  - Select **File > Save As** and choose filename RCS_Cu_Sphere_125mm.aedt
- Set Solution Type and 3D Model
  - Select **HFSS > Solution Type**
    - Check Solution Type: Modal
    - Check Driven Options: Network Analysis
  - Select **Modeler > Units**
    - Select Meters
  - In the Modeler Toolbar, select **Materials**
    - Set default to Vacuum

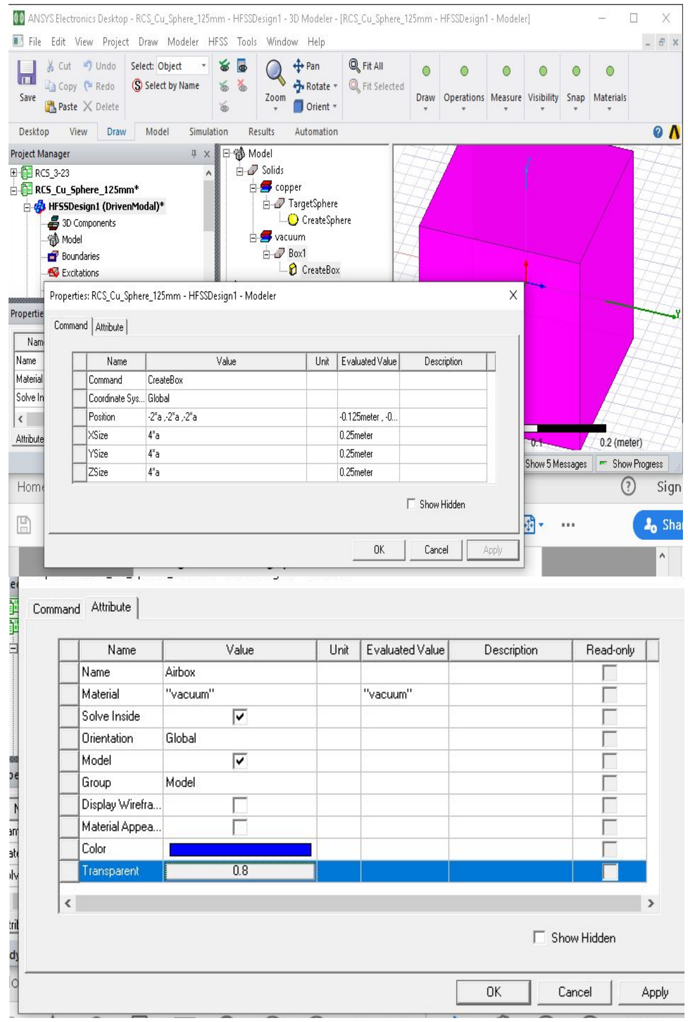
- Draw the Conducting Sphere
  - Select **Draw > Sphere**
    - Click on Origin of XYZ coordinates, then move mouse away from origin and click again to draw sphere
    - A **Properties** window should appear
      - Center Position: (0,0,0)
      - Radius: type ‘a’ as thevariable for radius
      - Unit Type: Length
      - Unit: meter
      - Value: 0.0625



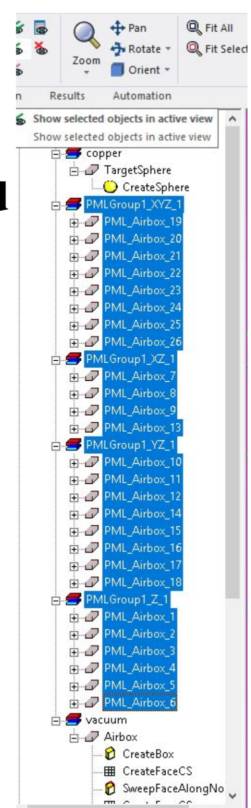
- Edit the Sphere Properties
  - In the modeler window, right-click **Sphere1 > Properties**
    - Name: TargetSphere
    - Material: “copper”
    - Color: Red



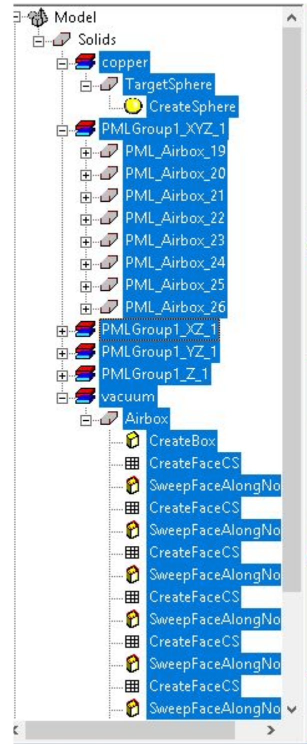
- Create the Airbox
  - Select **Draw > Box**
    - Draw 3D box anywhere in coordinate system
    - **Properties** window should come up again
      - Position: (-2*a, -2*a, -2*a)
      - XSize: 4*a
      - YSize: 4*a
      - ZSize: 4*a
  - In the Modeler Window, right click **Box1 > Properties > Attribute**
    - Name: Airbox
    - Material: Vacuum
    - Color: Blue
    - Transparent: 0.8

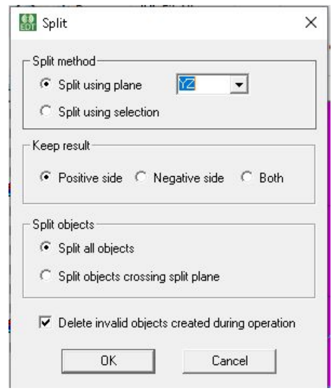


- Create PML (Perfectly Matched Layer)
  - A PML Box emulates an infinite vacuum or ideal anechoic chamber
  - In the Toolbar, select **Edit > Selection Mode > Faces**
  - **Edit > Select Objects > By Name**
    - Select **Airbox** then highlight all Faces and click OK
  - In the Toolbar, select **HFSS > Boundaries > PML Setup Wizard**
  - Check "Create PML Cover Objects On Selected Faces"
    - Uniform Layer Thickness : 0.250 meter (4*radius length)
  - Click **Next**
  - Check "PML Objects Accept Free Radiation"
    - Min Frequency: 0.04 GHz
    - Minimum Radiating Distance: 0.125 meter (2*radius length)
  - Click **Finish**
  - In Modeler Window, highlight all PMLGroup___ items
    - Click Green Eye up top to make visible

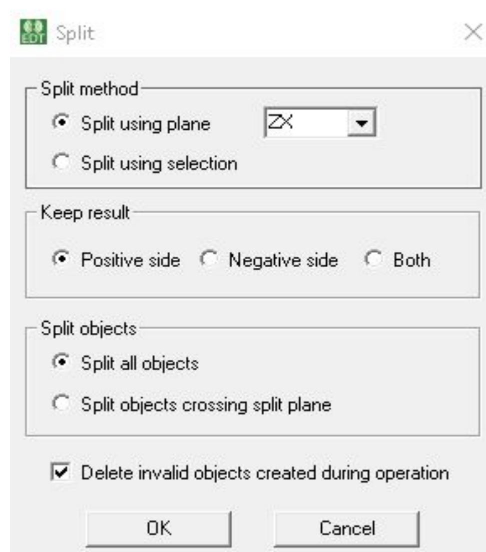


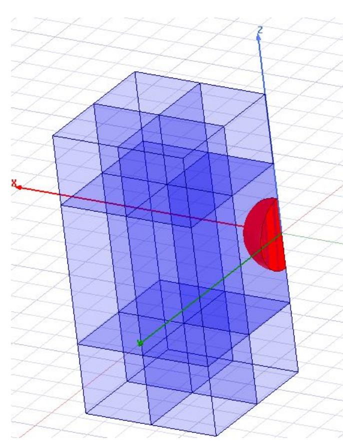
- Use symmetry to simplify solution
  - Because our target object is symmetrical, we can apply a symmetrical boundary condition on only a quarter of the sphere to reduce computation time
  - We will cut sphere, then set Perfect E and H boundaries on the appropriate sphere faces
  - In Modeler Window, highlight ALL Solids(include all items in drop-downs of PMLGroup_ objects)
    - In Toolbar, select **Modeler > Boolean > Split**
      - Split Plane: YZ
      - Keep Result: Positive Side
      - Splot Objects: Split all objects
      - Delete invalid objects



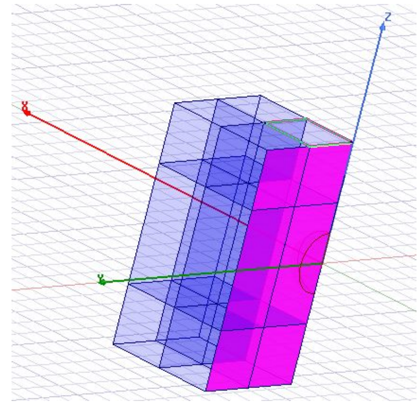


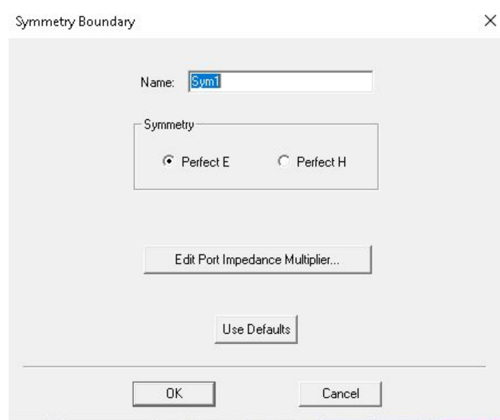
- Use Symmetry(continued)
  - In Modeler Window, highlight ALL Solids again (include all items in drop-downs of PMLGroup_ objects and new objects created from last split)
    - In Toolbar, select **Modeler > Boolean > Split**
      - Split Plane: ZX
      - Keep Result: Positive Side
      - Splot Objects: Split all objects
      - Delete invalid objects
  - You should now have a perfectly quartered sphere, Airbox, and PML Layer



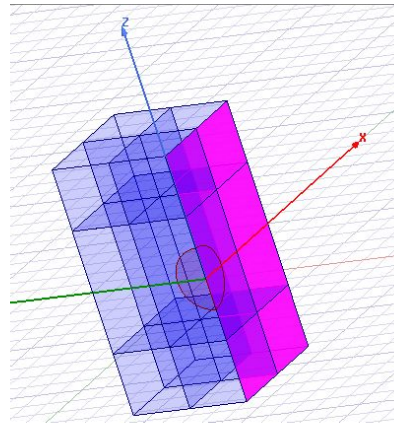


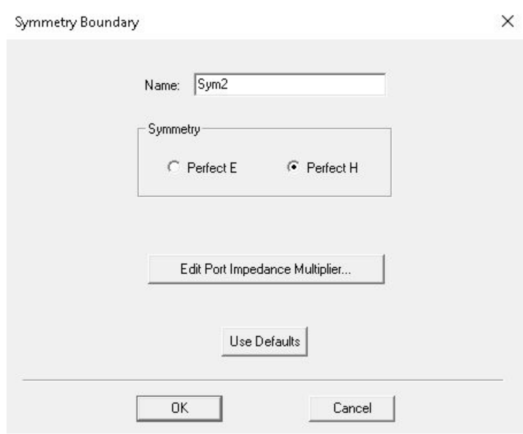
- Assign Boundary Conditions (YZ)
  - Select all the Faces in the YZ Plane:
    - On the Toolbar, select **Edit > Selection Mode > Faces**
    - In the 3D Model Window, hold CTRL and select all 6 faces on the YZ Plane
    - On the Toolbar, select **HFSS > Boundaries > Assign > Symmetry**
      - Select the “Perfect E” Button



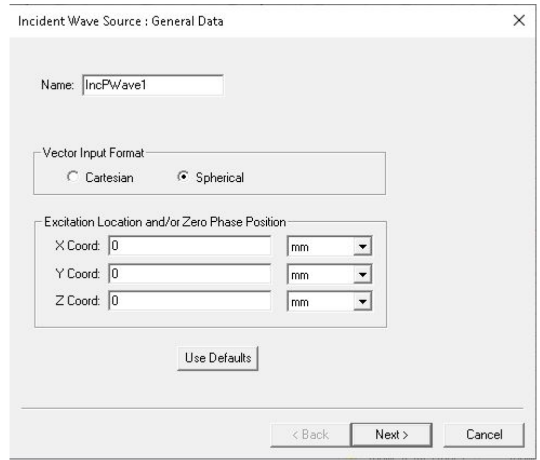


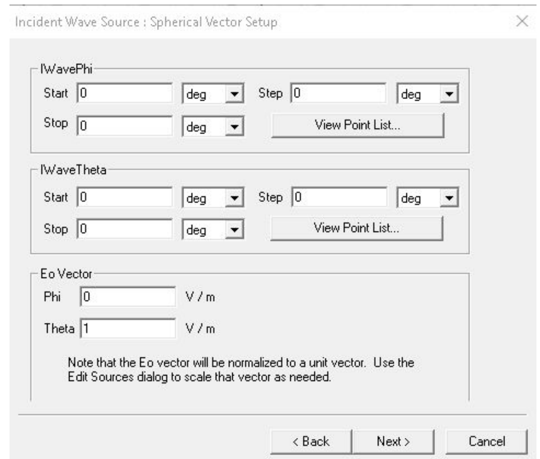
- Assign Boundary Conditions (XZ)
  - Select all the Faces in the XZ Plane:
    - On the Toolbar, select **Edit > Selection Mode > Faces**
    - In the 3D Model Window, hold CTRL and select all 6 faces on the XZ Plane
    - On the Toolbar, select **HFSS > Boundaries > Assign > Symmetry**
      - Select the “Perfect H” Button



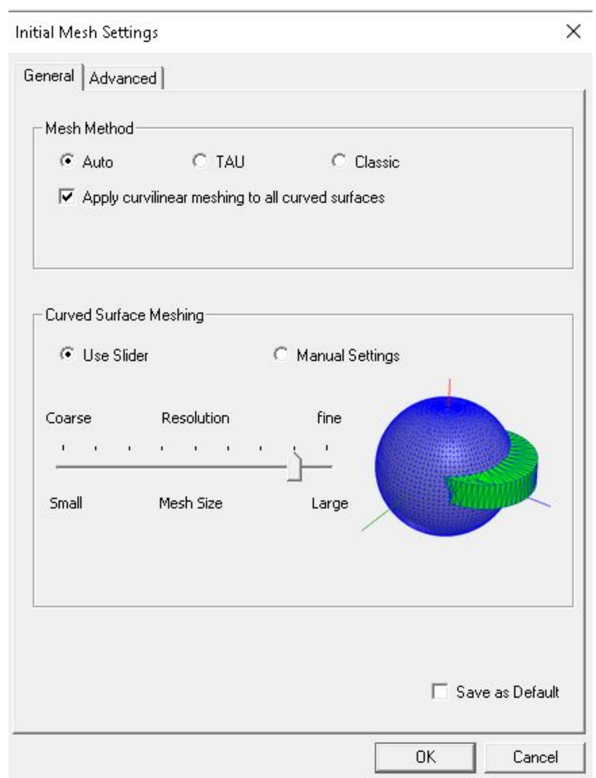


- Assign Plane Wave
  - On the Toolbar, select **HFSS > Excitations > Assign > Incident Wave > Plane Wave**
    - Vector Input Format: Spherical
  - Click Next
    - IWavePhi Start, Stop, Step: 0
    - IWaveTheta Start, Stop, Step: 0
    - Eo Vector
      - Phi: 0
      - Theta: 1
  - Click Next
    - Type of Plane Wave: Regular/Propagating
  - Click Finish

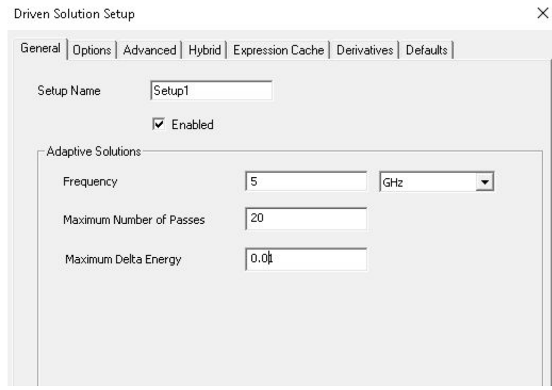


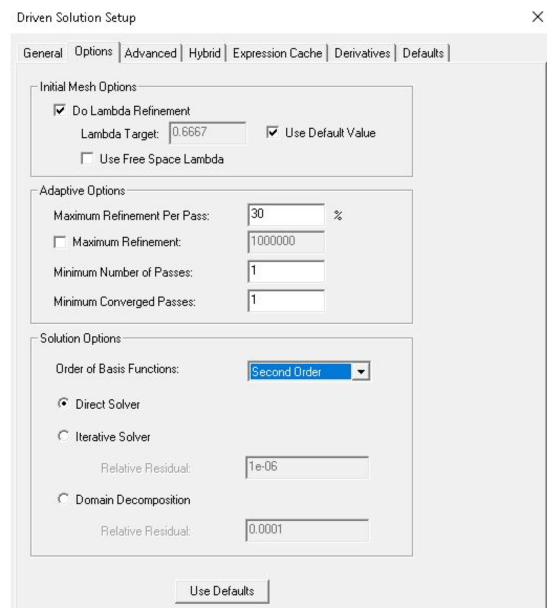


- Assign Meshing
  - On the Toolbar, select **HFSS > Mesh Operations > Initial Mesh Settings**
    - Check “Apply curvilinear meshing to all curved surfaces”
    - Set Mesh Size to Large/fine for greater accuracy at higher frequencies but longer computation time

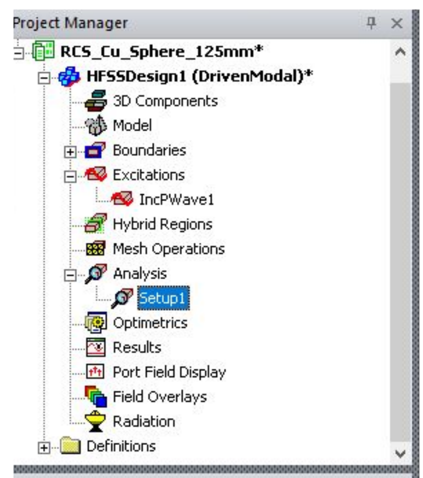


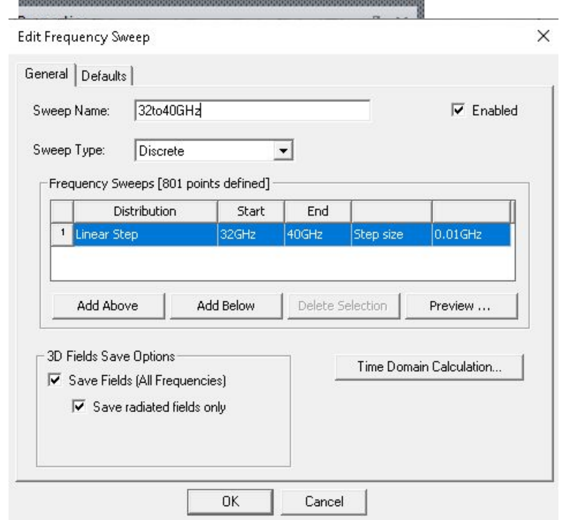
- Add Solution Setup
  - In the Toolbar, select **HFSS > Analysis Setup > Add Solution Setup**
    - Frequency: 5 GHz (This is not the excitation frequency)
    - Maximum Number of Passes: 20
    - Maximum Delta Energy: 0.01
  - Select Options tab
    - Order of Basis Functions: Second Order

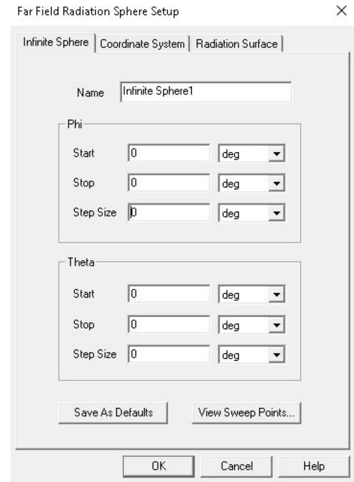




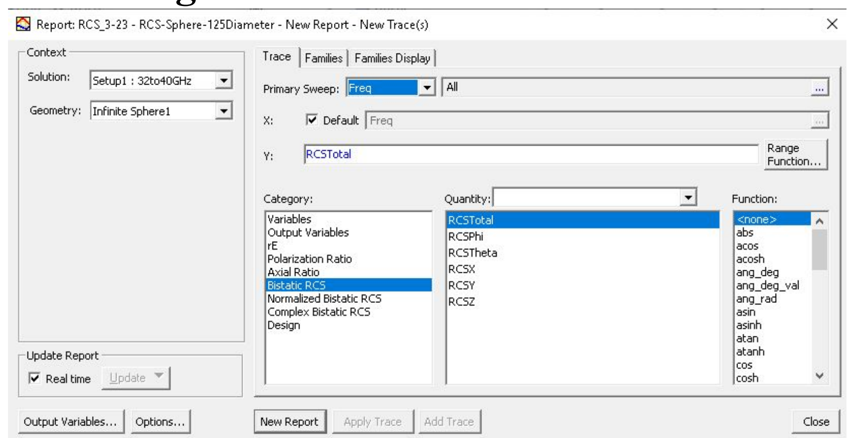
- Add Solution Setup
  - In the Project Manager window, expand the HFSSDesign1 dropdown menu
  - Expand "Analysis"
  - Highlight "Setup1"
    - In the Toolbar, select **HFSS > Analysis Setup > Add Frequency Sweep**
      - Sweep Name: 32to40GHz
      - Sweep Type: Discrete
      - Distribution: Linear Step
      - Start: 32 GHz
      - End: 40 GHz
      - Step Size: 0.01 GHz
      - Check "Save Radiated Field Only" box



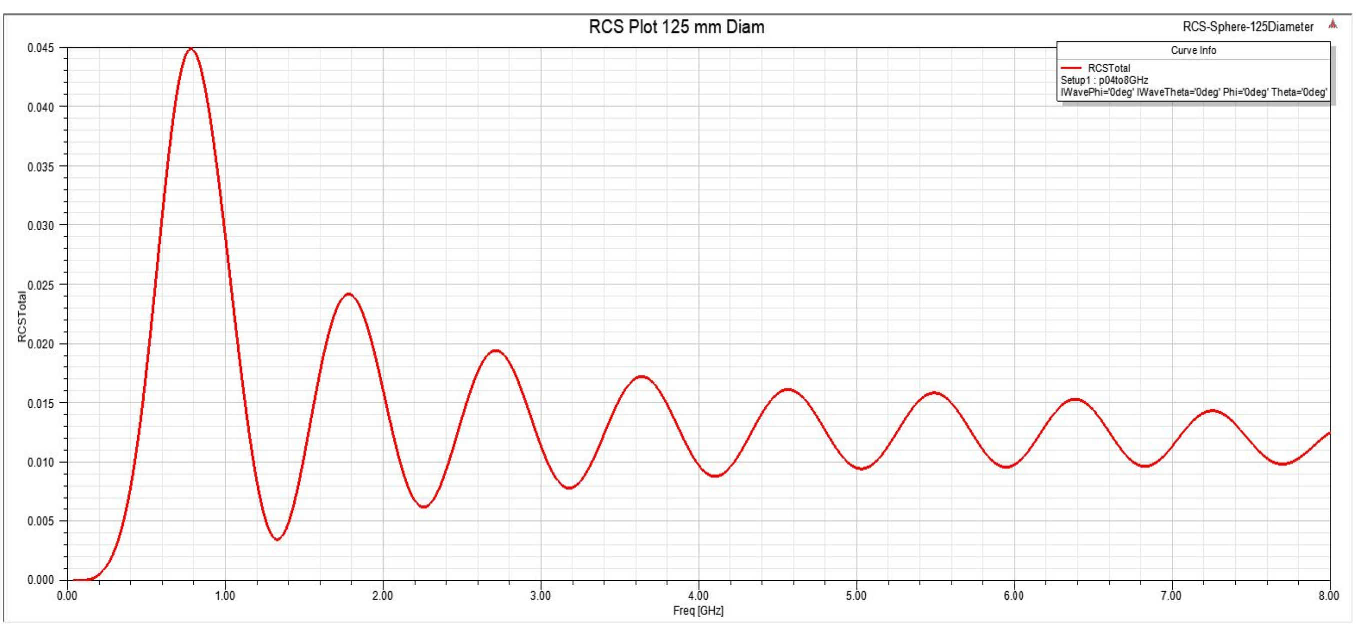




- Add Far-Field Setup
  - In the Toolbar, select **HFSS > Radiation > Insert Far Field Setup > Infinite Sphere**
    - Phi Start, Stop, Step Size: 0, 0, 0
    - Theta Start, Stop, Step Size: 0, 0, 0
- Add Plot
  - In the Toolbar, select **HFSS > Results > Create Far Fields Report > Rectangular Plot**
    - Primary Sweep: Freq
    - Category: Bistatic RCS
    - Quantity: RCSTotal
    - Click New Report



- Save RCS Data for Processing
  - Last step should have resulted in a Plot of RCS(in meters^2) as a function of Frequency
  - Right click plot to Export as image and/or .csv file in desired location



2.2 Plotting Normalized RCS from Simulated RCS

After running the simulations in ANSYS and exporting the results in csv format to a known directory, the Normalized RCS is plotted by running the MATLAB script `Plot_Normalized_RCS.m`, which is available in the team Github repository. An example output plot from `Plot_Normalized_RCS.m` is given in Figure 3.

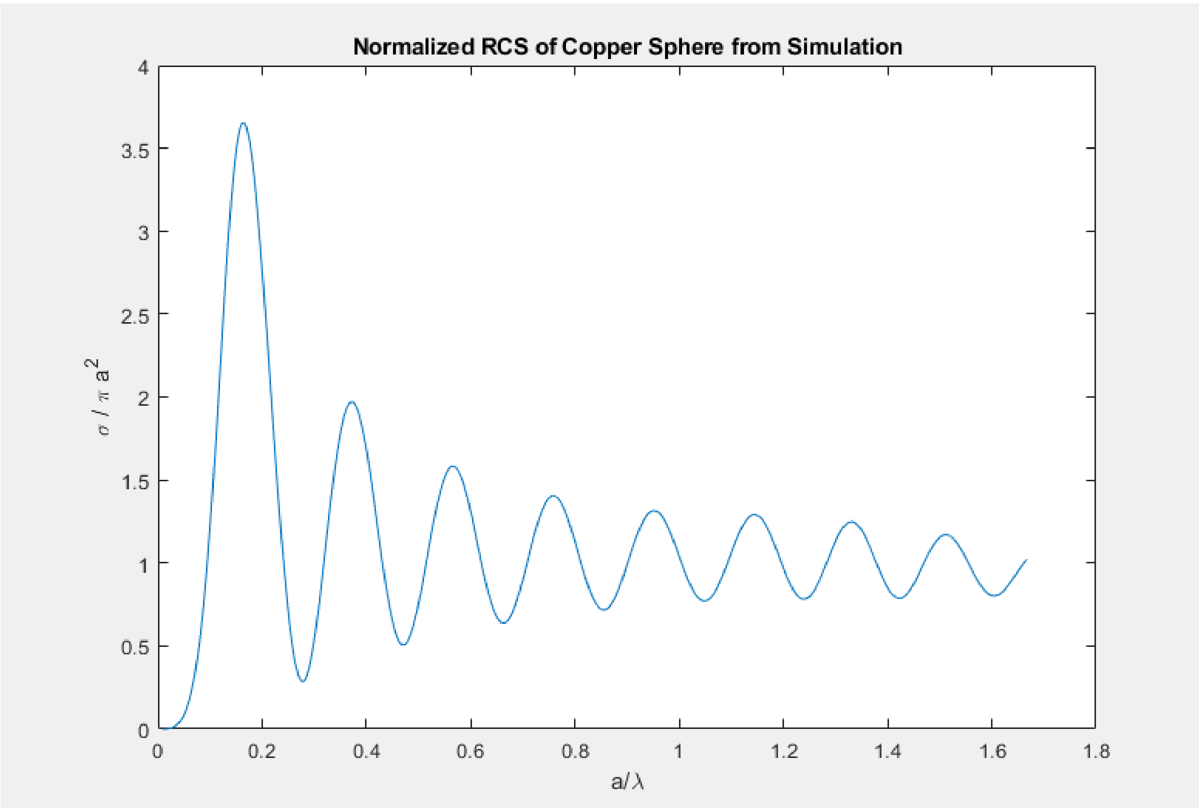


Figure 3: Normalized RCS from HFSS Simulations of 125 mm Diameter Copper Sphere

The code is is embedded as follows:

```
%Program accepts RCS csv files from ANSYS simulations and plots normalized
%RCS.

clc, clear, close all

%% User Parameters and filenames/filepaths

sphere_diam = 125; %Diameter of sphere in mm

%Load simulated frequency and RCS data into N x 2 matrix: [frequency, RCS_values]
```

```
file_name = '2-40GHz-full-125mm-diameter.csv'; %Name of csv file
%file_path = '\\thoth.cecs.pdx.edu\Home03\kjungles\My Documents\MATLAB\Capstone\'; %Folder where file is stored
file_path = '' %Leave uncommented if destination is PWD
fpath_name_sim = [file_path file_name]; %concatenate file path and file name

%% Read SIMULATED Data Files From CSV

file_data = csvread(fpath_name_sim, 1,0); %Read data starting at Row offset = 1, Column offset = 0; omits text header

%Store file data in respective frequency and RCS vectors
freq_sim = file_data(:,1);
RCS_sim = file_data(:,2);

%Plot RCS vs Frequency
figure(1)
plot(freq_sim,RCS_sim);
title('RCS of Copper Sphere, Simulated')
xlabel('Frequency (GHz)')
ylabel('Monostatic RCS (m^{2})')

%Plot normalized RCS vs Frequency
a = sphere_diam*10^-3/2
lam = 299792458./(freq_sim*10^9)
y_sim = RCS_sim/(pi*a^2)
x = a./lam

figure(2)
plot(x,y_sim);
title('Normalized RCS of Copper Sphere from Simulation')
ylabel('\sigma / \pi a^2')
xlabel('a/\lambda')
```

3 Comparing Experimental and Simulated RCS

Final analysis of the RCS measurements is performed by comparing the experimental RCS data with simulated RCS data. The MATLAB Script `RCS_compare.m`, which is available in the team Github repository, performs this action. With small changes to the code, experimental RCS data can also be compared to other experimental data. Such changes are indicated in the comments of the code.

An example output plot from `RCS_compare.m` is given in Figure 4.

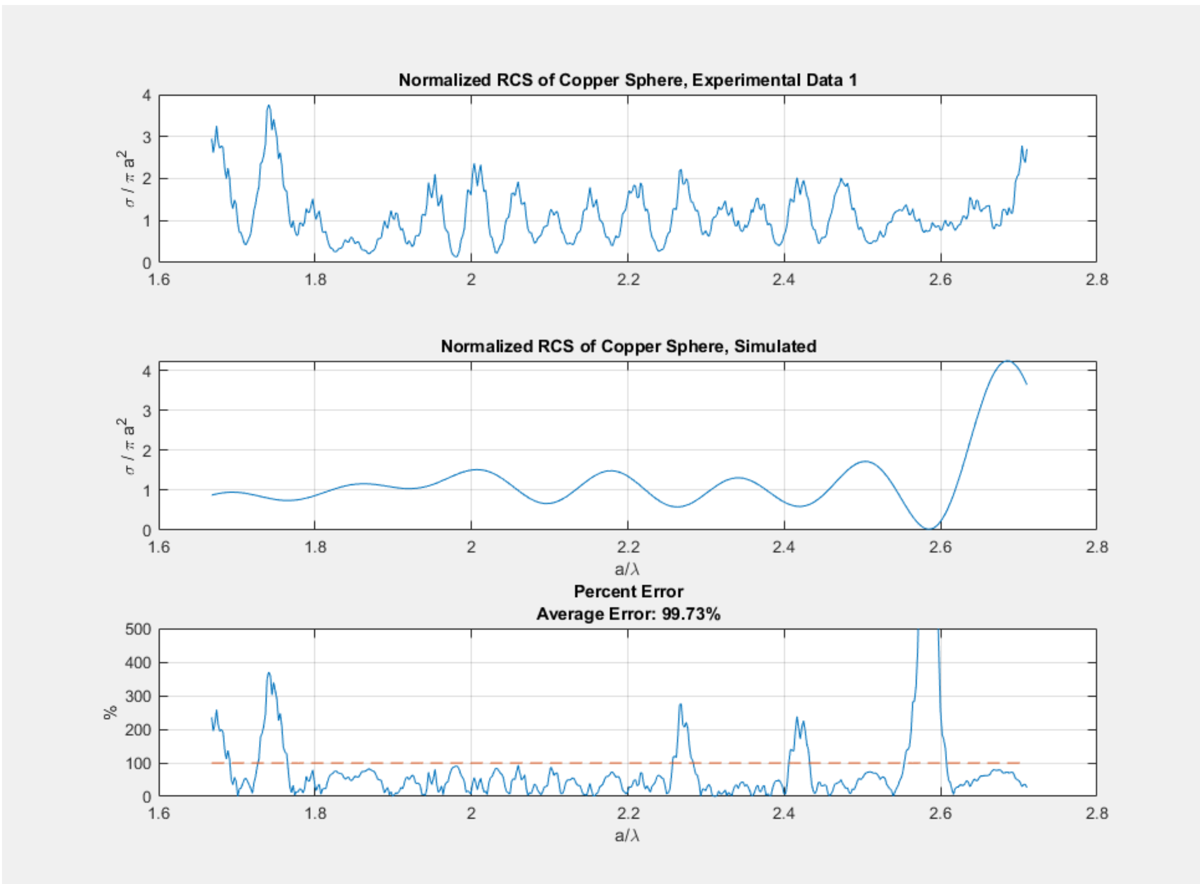


Figure 4: Normalized RCS Comparison

It is worth noting that the experimental RCS for Figure 4 was gathered in a living room, and the simulated RCS was performed with a coarse mesh, so this example plot should not to be regarded as ideal.

The program is embedded as follows:

```
%Program reads CSV files of RCS vs frequency for both
    measured and
%simulated data. Compares RCS values and determines how
    well measurement
```

```
%matches simulation.
clc, clear, close all

%% User Parameters and filenames/filepaths

sphere_diam = 125; %Diameter of sphere in mm

%Load frequency and EXPERIMENTAL RCS dataset 1 into N x 2 matrix: [frequency RCS_values]
file_name = '8-13GHz_RCS-EXP_5-19_NO-GATE.csv'; %Name of csv file
%file_path = '\\thoth.cecs.pdx.edu\Home03\kjungles\My Documents\Capstone\RCS_results\'; %Folder where file is stored
file_path = '' %Leave uncommented if destination is PWD
fpath_name_exp1 = [file_path file_name]; %concatenate file path and file name

% %Load frequency and EXPERIMENTAL RCS dataset 2 into N x 2 matrix: [frequency RCS_values]
% file_name = '8-13GHz_RCS-EXP_5-19_GATED.csv'; %Name of csv file
% %file_path = '\\thoth.cecs.pdx.edu\Home03\kjungles\My Documents\Capstone\RCS_results\'; %Folder where file is stored
% file_path = '' %Leave uncommented if destination is PWD
% fpath_name_exp2 = [file_path file_name]; %concatenate file path and file name

%Load frequency and SIMULATED RCS data into N x 2 matrix: [frequency RCS_values]
file_name = '2-40GHz-full-125mm-diameter.csv'; %Name of csv file
%file_path = '\\thoth.cecs.pdx.edu\Home03\kjungles\My Documents\MATLAB\Capstone\'; %Folder where file is stored
file_path = '' %Leave uncommented if destination is PWD
fpath_name_sim = [file_path file_name]; %concatenate file path and file name

%% Read EXPERIMENTAL Data File 1 From CSV

%Read and plot RCS for Experimental Dataset 1
file_data = csvread(fpath_name_exp1, 1,0); %Read data
```

```
starting at Row offset = 1, Column offset = 0; omits
   text header

%Store file data in respective frequency and RCS vectors
freq_exp1 = file_data(:,1);
RCS_exp1 = file_data(:,2);

%Plot RCS vs Frequency
figure(1)
plot(freq_exp1,RCS_exp1);
title('RCS of Copper Sphere, Experimental Data 1')
xlabel('Frequency (GHz)')
ylabel('Monostatic RCS (m^{2})')

%% Read EXPERIMENTAL Data File 2 From CSV
% file_data = csvread(fpath_name_exp2, 1,0); %Read data
   starting at Row offset = 1, Column offset = 0; omits
   text header
%
% %Store file data in respective frequency and RCS vectors
% freq_exp2 = file_data(:,1);
% RCS_exp2 = file_data(:,2);
%
% %Plot RCS vs Frequency
% figure(1)
% plot(freq_exp2,RCS_exp2);
% title('RCS of Copper Sphere, Experimental Data 2')
% xlabel('Frequency (GHz)')
% ylabel('Monostatic RCS (m^{2})')

%% Read SIMULATED Data Files From CSV

file_data = csvread(fpath_name_sim, 1,0); %Read data
   starting at Row offset = 1, Column offset = 0; omits
   text header

%Store file data in respective frequency and RCS vectors
freq_sim = file_data(:,1);
RCS_sim = file_data(:,2);

%Plot RCS vs Frequency
figure(2)
plot(freq_sim,RCS_sim);
title('RCS of Copper Sphere, Simulated')
```

```
xlabel('Frequency (GHz)')
ylabel('Monostatic RCS (m^{2})')

% %Plot normalized RCS vs Frequency
% a = sphere_diam*10^-3/2
% lam = 299792458./(freq_sim*10^9)
% y_sim = RCS_sim/(pi*a^2)
% x = a./lam
%
% figure
% subplot(2,1,1)
% plot(x,y_sim);
% title('Normalized RCS of Copper Sphere, Simulated')
% ylabel('\sigma / \pi a^2')
% xlabel('a/\lambda')
% xlim([0.5,3])
% ylim([0,4])
%
% subplot(2,1,2)
% plot(x,y_sim);
% title('Normalized RCS of Copper Sphere, Simulated')
% ylabel('\sigma / \pi a^2')
% xlabel('a/\lambda')



%% Shorten Simulated data to frequency range of
   experimental

a = find(freq_sim == freq_exp1(1));
b = find(freq_sim == freq_exp1(end));

freq_sim = freq_sim(a:b);
RCS_sim = RCS_sim(a:b);

%% Plot Both Experimental and Simulated RCS

figure(3)
subplot(2,1,1)
plot(freq_exp1,RCS_exp1);
title('RCS of Copper Sphere, Experimental Data 1')
xlabel('Frequency (GHz)')
ylabel('Monostatic RCS (m^{2})')

```

```
subplot(2,1,2)
plot(freq_sim,RCS_sim)
title('RCS of Copper Sphere, Simulated')
xlabel('Frequency (GHz)')
ylabel('Monostatic RCS (m^{2})')

%Uncomment and adjust subplot() arguments on previous to plot Experimental
%Data 2
% subplot(3,1,2)
% plot(freq_exp2,RCS_exp2);
% title('RCS of Copper Sphere, Experimental Data 2')
% xlabel('Frequency (GHz)')
% ylabel('Monostatic RCS (m^{2})')

%% Interpolate Experimental RCS to match Simulated

%Interpolate experimental data to match same frequency points as
%simulation
RCS_exp1_matched = interp1(freq_exp1, RCS_exp1, freq_sim);
%RCS_exp2_matched = interp1(freq_exp2, RCS_exp2, freq_sim);
freq = freq_sim;

figure(4)
subplot(2,1,1)
plot(freq,RCS_exp1_matched)
title(['RCS of Copper Sphere, Experimental Data 1'])
xlabel('Frequency (GHz)')
ylabel('Monostatic RCS (m^{2})')

subplot(2,1,2)
plot(freq,RCS_sim)
title(['RCS of Copper Sphere, Simulated'])
xlabel('Frequency (GHz)')
ylabel('Monostatic RCS (m^{2})')

%Uncomment and adjust subplot() arguments on previous to plot Experimental
%Data 2
% subplot(3,1,2)
% plot(freq,RCS_exp2_matched)
% title(['RCS of Copper Sphere, Experimental Data 2'])
% xlabel('Frequency (GHz)')
```

```matlab
% ylabel('Monostatic RCS (m^{2})')

%% Calculate Normalized RCS and Percent Error

%Calculate normalized RCS and a/lambda x axes
a = sphere_diam/2*10^-3;
lam = 299792458./(freq*10^9);
RCS_exp1_norm = RCS_exp1_matched/(pi*a^2);
%RCS_exp2_norm = RCS_exp2_matched/(pi*a^2)
RCS_sim_norm = RCS_sim/(pi*a^2);
x = a./lam;

%Calculate percent Error and average
percent_err = abs(RCS_exp1_norm-RCS_sim_norm)./RCS_sim_norm
    *100;
avg_err = mean(percent_err);
avg_err_plot = ones(1,length(x))*avg_err;

figure(5)
subplot(3,1,1)
plot(x,RCS_exp1_norm)
title(['Normalized RCS of Copper Sphere, Experimental Data
    1'])
ylabel('\sigma / \pi a^2')
grid on

subplot(3,1,2)
plot(x, RCS_sim_norm)
title(['Normalized RCS of Copper Sphere, Simulated'])
ylabel('\sigma / \pi a^2')
xlabel('a/\lambda')
grid on

subplot(3,1,3)
plot(x, percent_err,x,avg_err_plot,'--')
title({'Percent Error',['Average Error: ' sprintf('%.2f',
    avg_err) '%']})
ylabel('%')
ylim([0 500])
xlabel('a/\lambda')
grid on

%Uncomment and adjust subplot() arguments on previous to
    plot Experimental
```

```
%Data 2
% subplot(3,1,2)
% plot(x,RCS_exp2_norm)
% title(['Normalized RCS of Copper Sphere, Experimental Data 2(GATED)'])
% ylabel('\sigma / \pi a^2')
% grid on
```

References

[1] C. Balanis. *Antenna Theory: Analysis and Design*. Wiley-Interscience, Hoboken, New Jersey, 2005.