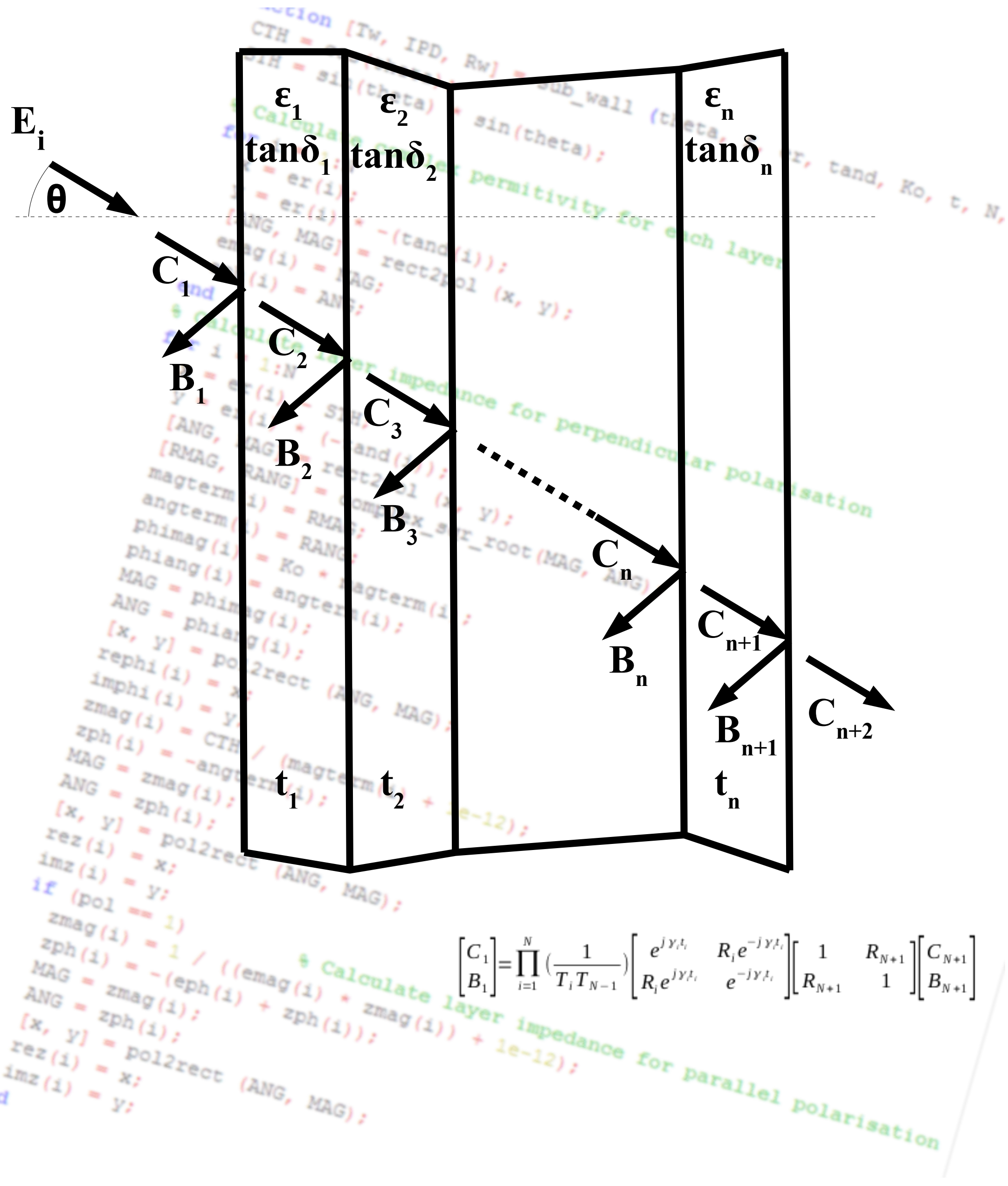
# A Software Tool for the Analysis of Multilayer Radomes

**Dr Paul Klasmann (2E0PMK)**

**May 2021**

**Version 1.2**



## Multilayer Radome Analysis Software Instructions

This document describes the usage of the software written as a script for GNU Octave but should also work with Matlab which will predict the transmission, reflection and insertion phase delay performance when a perpendicular or parallel polarized wave is incident upon a single or multilayer radome such as an A or C-sandwich radome, or a dielectric stack-up of any number of layers. It is assumed that the dielectric layers are homogenous and isotropic in nature as shown in Figure 1.

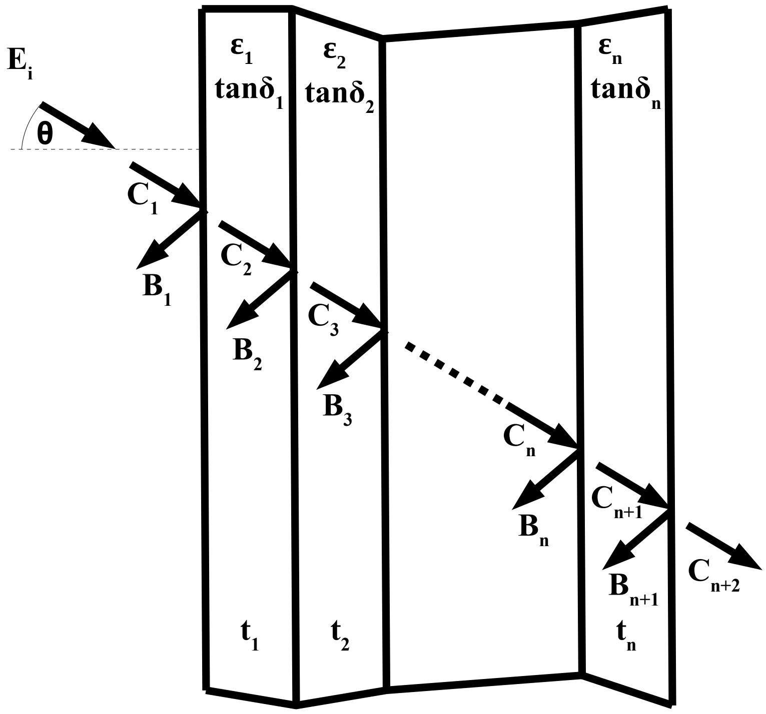



Figure.1

The analytical software is based on the boundary value solution of an N-layer dielectric wall using the Fresnel equations. This is described in the book "Analysis of Radome-Enclosed Antennas", Dennis Kozakoff (Artech House, 1997), and will be repeated here for completeness. If the forward and reverse propagating waves are $C_i$ and $B_i$ respectively, the solution is:

$$\begin{bmatrix} C_1 \\ B_1 \end{bmatrix} = \prod_{i=1}^{N} \left( \frac{1}{T_i T_{N-1}} \right) \begin{bmatrix} e^{j\gamma_i t_i} & R_i e^{-j\gamma_i t_i} \\ R_i e^{j\gamma_i t_i} & e^{-j\gamma_i t_i} \end{bmatrix} \begin{bmatrix} 1 & R_{N+1} \\ R_{N+1} & 1 \end{bmatrix} \begin{bmatrix} C_{N+1} \\ B_{N+1} \end{bmatrix} \quad (1)$$

where,

$t_i$ = Layer thickness of the $i^{th}$ layer.
$R_i$, $T_i$ = Fresnel reflection and transmission coefficients of the $i^{th}$ layer, respectively.
$\gamma_i$ = Propagation constant in the $i^{th}$ layer.

Angle of incidence θ is relative to the normal of the radome surface.

$$\gamma_i = k_0 \sqrt{\varepsilon_{ri} - \sin^2(\theta)} \tag{2}$$

The permittivity of the $i^{th}$ layer is given by $\varepsilon_i$ and $k_0$ is the wave number in a vacuum. Permittivity is given by:

$$\varepsilon_{ri} = \varepsilon'_{ri}(1 - j\tan(\delta_i)) \tag{3}$$

The Fresnel transmission and reflection coefficients are calculated from the following:

$$R_i = \frac{Z_i - Z_{i-1}}{Z_i + Z_{i-1}}; T_i = 1 - R_i \tag{4}$$

The wave impedance calculated depends on the polarization of the incident wave. For perpendicular polarization we have:

$$Z_i = \frac{\cos(\theta)}{\sqrt{\varepsilon_{ri} - \sin^2(\theta)}} \tag{5}$$

For parallel polarization we have:

$$Z_i = \frac{\sqrt{\varepsilon_{ri} - \sin^2(\theta)}}{\varepsilon_{ri}\cos(\theta)} \tag{6}$$

When the matrix (1) is multiplied out, the following is obtained:

$$\begin{bmatrix} C_1 \\ B_1 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} C_{N+2} \\ B_{N+2} \end{bmatrix} \tag{7}$$

The voltage reflection coefficient from the radome’s front surface is:

$$R_w = \frac{B_1}{C_1} | (B_{N+2} = 0) = \frac{A_{21}}{A_{11}} \tag{8}$$

The voltage transmission coefficient is:

$$T_w = \frac{C_{N+2}}{C_1} | (B_{N+2} = 0) = \frac{1}{A_{11}} \tag{9}$$

The reflection and transmission coefficients are both complex numbers, therefore it can be expressed as a magnitude and insertion phase delay (IPD), this is an angle. The radome transmission efficiency is:

$$|T_w|^2 \tag{10}$$

These equations are shown for completeness so that the program can be more easily followed for those who do not have access to “Analysis of Radome-Enclosed Antennas”. Buy the book if it’s available, I thoroughly recommend it!

The program is written in GNU Octave which should be compatible with Matlab. The master file calls a number of functions which are saved in separate files and must be present in the same folder/directory as the main function or any other function that calls them. An exhaustive description of the code will not be given but the complete listing of all functions is given at the end of the document and important sections are commented.

It is necessary to specify the problem with only changing these lines:

```
% Enter user variables here...
N = 5;                                  % Enter number of layers
fl = 17;                                % Enter lower frequency in GHz
fh = 32;                                % Enter upper frequency in GHz
no_of_freq_points = 301;                % Enter No. of frequency points
no_inc_ang = 3;                         % No. of incidence angles
ang_inc_step = 15;                      % Increment angle of incidence
% Enter the dielectric contatant/permitivity for each layer
er1 = 4.0;  er2 = 1.1;  er3 = 4.0;  er4 = 1.1;  er5 = 4.0;
% Enter the loss tangent/tand for each layer
tand1 = 0.003;  tand2 = 0.001;  tand3 = 0.003;  tand4 = 0.001;  tand5 = 0.003;
% Enter the layer thickness in mm for each layer
t1 = 0.24;  t2 = 2.1;  t3 = 0.48;  t4 = 2.1;  t5 = 0.24;

er = [er1, er2, er3, er4, er5]              % Enter dielectric constants
tand = [tand1, tand2, tand3, tand4, tand5]  % Enter tand values (loss tangent)
t = [t1, t2, t3, t4, t5]                    % Enter layer thickness in mm
% End of user variables
```

Values are entered as a vector (one dimensional array) in the order of dielectric layers. Add or remove layers as necessary. The above code is for a five layer stack-up.

If you require more or less plots for differing angles of incidence, set the “no_inc_ang”, increments can be set with “ang_inc_step”.

Note that this version of the program is written in a traditional program flow with for loops defined where necessary. It would be far more efficient to perform calculations using native complex number notation and matrix operations instead of converting numbers from Cartesian to polar form and vice versa and to use matrix operations. This may be done at a future date.

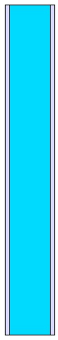
When the program is run, six plots will be generated. These are the reflection, transmission and insertion phase delay for both perpendicular and parallel polarization of the incident E-field. The code can be modified by the user to generate only what plots are necessary.

## Comparisons between Analytic Software and CST FEM Solver

To confirm that the analytical software is giving the correct result, two benchmarks are presented below with equivalent simulations using CST Microwave Studio and its FEM solver with unit cell boundaries. Although unit cell FEM simulations may take a couple of minutes, the analytical method takes just seconds to complete and generate plots with the option for several angle of incidences. The results below are for 0° incidence.

## A-Sandwich Radome (3-layer stackup)

A-Sandwich Sample



Outer Layers: plain weave GFRP, thickness 0.24mm, $\varepsilon_r = 4.0$.

Core Layer: Millifoam RHC71, thickness 2.5mm, $\varepsilon_r = 1.1$.

Values of tanδ are estimated to be 0.003 for GFRP, and 0.001 for the Millifoam, and these values were used for the analytical and numerical simulations for a like for like comparison.

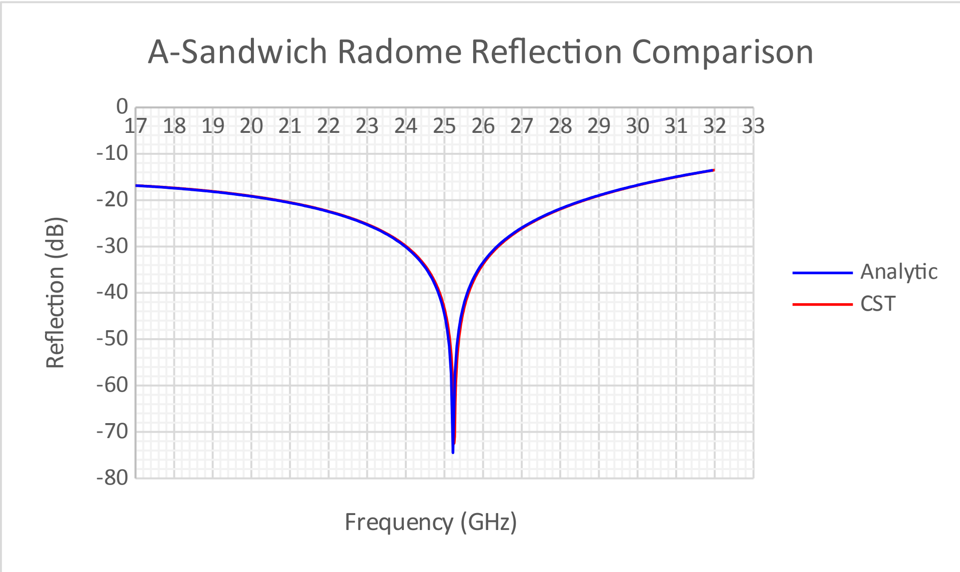



Figure. 2

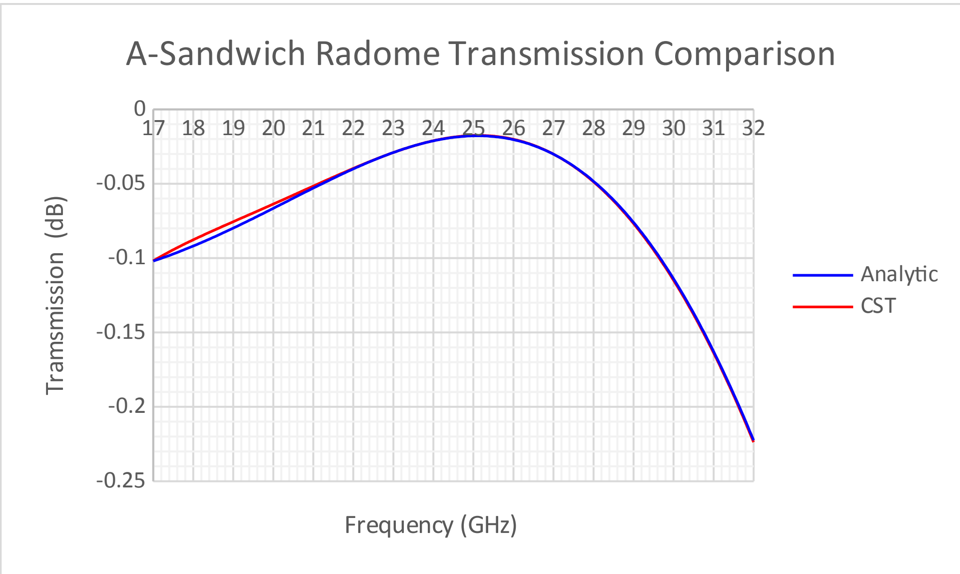



Figure. 3

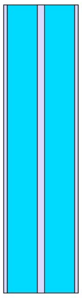
## C-Sandwich Radome (5-layer stackup)

C-Sandwich Sample

Outer Layers: plain weave GFRP, thickness 0.24mm, $\varepsilon_r = 4.0$.

Core Layers: Millifoam RHC71, thickness 2.1mm, $\varepsilon_r = 1.1$.

Middle Layer: plain weave GFRP, thickness 0.48mm, $\varepsilon_r = 4.0$.



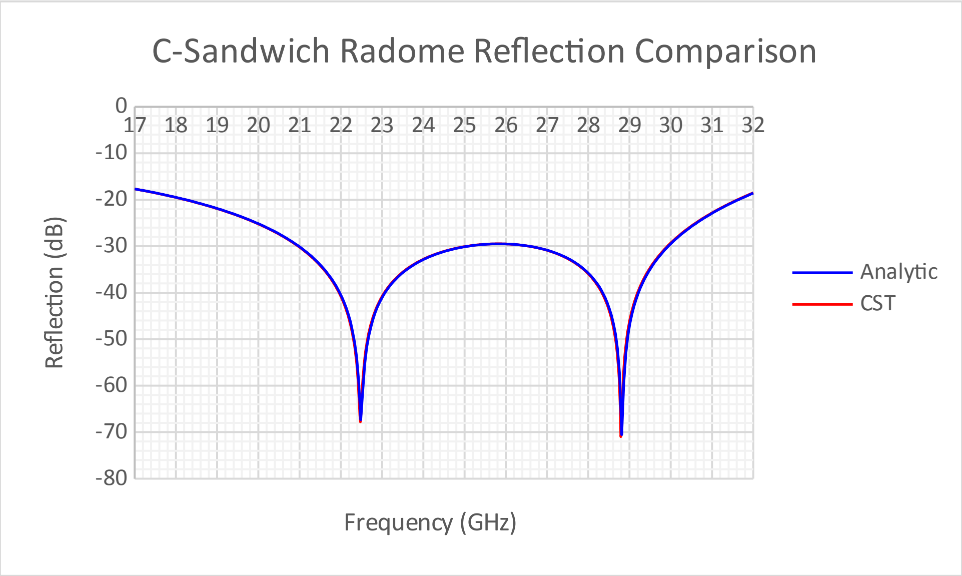



Figure. 4

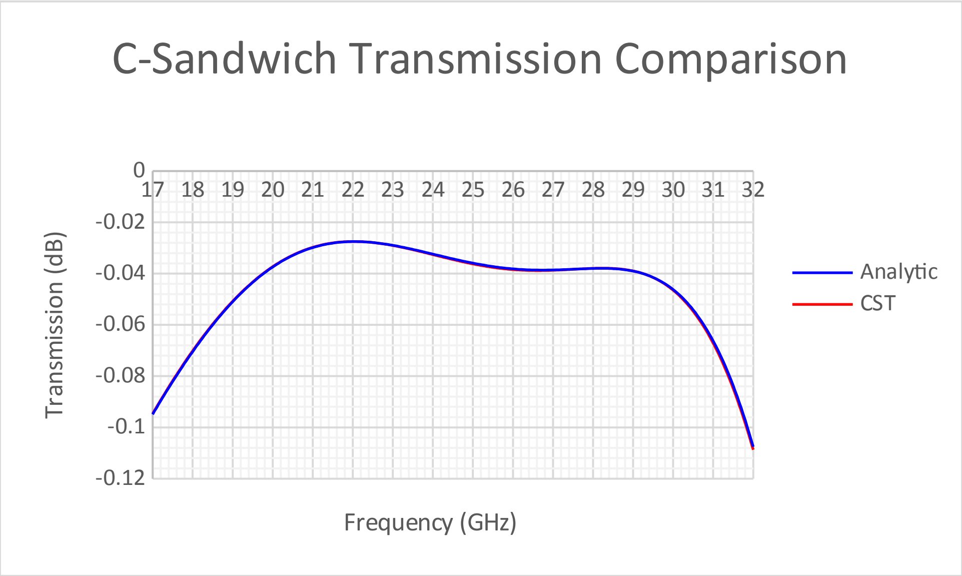



Figure. 5

## GNU Octave Code

```
% Program to calculate Transmission Loss, Insertion Phase Delay and Reflection
% for a multilayer radome.
% This program is a translation from 'Analysis of Radome-Enclosed Antennas'
% by Dennis J. Kozakoff, 1997, Artech House.
%
% Author: Dr Paul Klasmann
% Date: 05/11/2020      Version 1.1
%
% 1) Set N to equal the mumber of layers.
% 2) Set fl and fh to the lower and upper analysis frequency in GHz.
% 3) Set no_of_freq_points to a sensible value for the frequency range.
% 4) Setup the array values for the relative dielectric constant (er).
% 5) Setup the array values for the loss tangents (tand).
% 6) Setup the array values for the radome layer thicknesses (t) in mm.

close all
clear
clc
% Enter user variables here...
N = 5;                                    % Enter number of layers
fl = 17;                                  % Enter lower frequency in GHz
fh = 32;                                  % Enter upper frequency in GHz
no_of_freq_points = 301;                  % Enter No. of frequency points
no_inc_ang = 3;                           % No. of incidence angles
ang_inc_step = 15;                        % Increment angle of incidence
% Enter the dielectric contatant/permitivity for each layer
er1 = 4.0;  er2 = 1.1;  er3 = 4.0;  er4 = 1.1;  er5 = 4.0;
% Enter the loss tangent/tand for each layer
tand1 = 0.003;  tand2 = 0.001;  tand3 = 0.003;  tand4 = 0.001;  tand5 = 0.003;
% Enter the layer thickness in mm for each layer
t1 = 0.24;  t2 = 2.1;  t3 = 0.48;  t4 = 2.1;  t5 = 0.24;

er = [er1, er2, er3, er4, er5]                % Enter dielectric constants
tand = [tand1, tand2, tand3, tand4, tand5]    % Enter tand values (loss tangent)
t = [t1, t2, t3, t4, t5]                      % Enter layer thickness in mm
% End of user variables

finc = (fh-fl)/no_of_freq_points;      % Frequency increment
eo = 8.854e-12;                        % Permativity of free space (F/m)
RAD = pi/180;                          % For degree to radians conversion

% Begin Calculation
for pol = 0:1             % Pol=0 for Perpendicular Polarization
  array_index = 0;        % Pol=1 for parallel polarization
  for f = fl:finc:fh
    array_index = array_index + 1;
    LAM = 300/f;                  % Lambda is wavelength in mm
    Ko = (2*pi/LAM);              % Ko is wavenumber in mm^(-1)
    ANGLE = 0;                    % Initialize first incident angle to calculate
        for angle_index = 1:no_inc_ang;
        theta = ANGLE * RAD;      % Theta is angle of incidence in radians
        % Call function sub_wall to perform main calculations
        [Tw, IPD, Rw] = sub_wall(theta, f, er, tand, Ko, t, N, pol);
        TwdB(array_index, angle_index) = 20 * log10(Tw);
        IPDdeg(array_index, angle_index) = -IPD / RAD;
        RwdB(array_index, angle_index) = 20 * log10(Rw);
        if(IPDdeg(array_index, angle_index) == -360)
          IPDdeg(array_index, angle_index) = 0;
        end
        ANGLE = ANGLE + ang_inc_step;         % Increment angle of incidence
```

```
      end  % END ANGLE LOOP
  end      % END f LOOP

f = fl:finc:fh;
% Transmission Loss for perpendicular polarisation
if (pol == 0)
  % Plot TL for perpendicular polarisation
  figure
  for angle_index = 1:no_inc_ang;
  plot(f, TwdB(1:end,angle_index))
  hold on;
  end
  grid on
  set(gca,'linewidth',2, 'fontsize', 14, 'XTick', fl:1:fh)
  legendtext = ['0 degrees'; '15 degrees'; '30 degrees'; '45 degrees'; '60 degrees';
'75 degrees'];
  legend( legendtext, 'location', 'southeast' );
  title('Transmission Loss for Perpendicular Polarization','FontSize', 16);
  xlabel('Frequency (GHz)','FontSize', 14);
  ylabel('Transmission Loss (dB)','FontSize', 14);
  drawnow

  % Plot IPD for perpendicular polarisation
  figure
  for angle_index = 1:no_inc_ang;
  plot(f, IPDdeg(1:end,angle_index))
  hold on;
  end
  grid on
  set(gca,'linewidth',2, 'fontsize', 14, 'XTick', fl:1:fh)
  legendtext = ['0 degrees'; '15 degrees'; '30 degrees'; '45 degrees'; '60 degrees';
'75 degrees'];
  legend( legendtext, 'location', 'southeast' );
  title('Insertion Phase Delay for Perpendicular Polarization','FontSize', 16);
  xlabel('Frequency (GHz)','FontSize', 14);
  ylabel('Insertion Phase Delay (degrees)','FontSize', 14);
  drawnow

  % Plot Reflection for perpendicular polarisation
  figure
  for angle_index = 1:no_inc_ang;
  plot(f, RwdB(1:end,angle_index))
  hold on;
  end
  grid on
  set(gca,'linewidth',2, 'fontsize', 14, 'XTick', fl:1:fh)
  legendtext = ['0 degrees'; '15 degrees'; '30 degrees'; '45 degrees'; '60 degrees';
'75 degrees'];
  legend( legendtext, 'location', 'southeast' );
  title('Reflection for Perpendicular Polarization','FontSize', 16);
  xlabel('Frequency (GHz)','FontSize', 14);
  ylabel('Return Loss (dB)','FontSize', 14);
  drawnow

% Transmission Loss for Parallel Polarisation
elseif pol == 1
  % Plot TL for parallel polarisation
  figure
  for angle_index = 1:no_inc_ang;
  plot(f, TwdB(1:end,angle_index))
  hold on;
  end
  grid on
```

```
  set(gca,'linewidth',2, 'fontsize', 14, 'XTick', fl:1:fh)
  legendtext = ['0 degrees'; '15 degrees'; '30 degrees'; '45 degrees'; '60 degrees';
'75 degrees'];
  legend( legendtext, 'location', 'southeast' );
  title('Transmission Loss for Parallel Polarization','FontSize', 16);
  xlabel('Frequency (GHz)','FontSize', 14);
  ylabel('Transmission Loss (dB)','FontSize', 14);
  drawnow

  % Plot IPD for parallel polarisation
  figure
  for angle_index = 1:no_inc_ang;
  plot(f, IPDdeg(1:end,angle_index))
  hold on;
  end
  grid on
  set(gca,'linewidth',2, 'fontsize', 14, 'XTick', fl:1:fh)
  legendtext = ['0 degrees'; '15 degrees'; '30 degrees'; '45 degrees'; '60 degrees';
'75 degrees'];
  legend( legendtext, 'location', 'southeast' );
  title('Insertion Phase Delay for Parallel Polarization','FontSize', 16);
  xlabel('Frequency (GHz)','FontSize', 14);
  ylabel('Insertion Phase Delay (degrees)','FontSize', 14);
  drawnow

  % Plot Reflection for parallel polarisation
  figure
  for angle_index = 1:no_inc_ang;
  plot(f, RwdB(1:end,angle_index))
  hold on;
  end
  grid on
  set(gca,'linewidth',2, 'fontsize', 14, 'XTick', fl:1:fh)
  legendtext = ['0 degrees'; '15 degrees'; '30 degrees'; '45 degrees'; '60 degrees';
'75 degrees'];
  legend( legendtext, 'location', 'southeast' );
  title('Reflection for Parallel Polarization','FontSize', 16);
  xlabel('Frequency (GHz)','FontSize', 14);
  ylabel('Return Loss (dB)','FontSize', 14);
  drawnow

end
end         % END pol LOOP
```

## Function sub_wall

```
function [Tw, IPD, Rw] = sub_wall (theta, f, er, tand, Ko, t, N, pol)
  CTH = cos(theta);
  STH = sin(theta) * sin(theta);

  % Calculate complex permitivity for each layer
  for i = 1:N
    x = er(i);
    y = er(i) * -(tand(i));
    [ANG, MAG] = rect2pol (x, y);
    emag(i) = MAG;
    eph(i) = ANG;
  end
  % Calculate layer impedance for perpendicular polarisation
  for i = 1:N
    x = er(i) - STH;
    y = er(i) * (-tand(i));
    [ANG, MAG] = rect2pol (x, y);
    [RMAG, RANG] = complex_sqr_root(MAG, ANG);
    magterm(i) = RMAG;
    angterm(i) = RANG;
    phimag(i) = Ko * magterm(i);
    phiang(i) = angterm(i);
    MAG = phimag(i);
    ANG = phiang(i);
    [x, y] = pol2rect (ANG, MAG);
    rephi(i) = x;
    imphi(i) = y;
    zmag(i) = CTH / (magterm(i) + 1e-12);
    zph(i) = -angterm(i);
    MAG = zmag(i);
    ANG = zph(i);
    [x, y] = pol2rect (ANG, MAG);
    rez(i) = x;
    imz(i) = y;
    if (pol == 1)         % Calculate layer impedance for parallel polarisation
      zmag(i) = 1 / ((emag(i) * zmag(i)) + 1e-12);
      zph(i) = -(eph(i) + zph(i));
      MAG = zmag(i);
      ANG = zph(i);
      [x, y] = pol2rect (ANG, MAG);
      rez(i) = x;
      imz(i) = y;
    end
  end
% End of layer impedance calculation
  zmag(N+1) = 1;
  zph(N+1) = 0;
  rez(N+1) = 1;
  imz(N+1) = 0;
% Do next loop i = 1 serperately from the loop to include z for space before radome
% because indexing starts at 1 and not 0. Then proceed with loop starting i=2
  i = 1;
  x = rez(i) - 1;
  y = imz(i) - 0;
  [ANG, MAG] = rect2pol (x, y);
  NUMMAG = MAG;
  NUMANG = ANG;
  x = rez(i) + 1;
  y = imz(i) + 0;
  [ANG, MAG] = rect2pol (x, y);
```

```
  DENMAG = MAG;
  DENANG = ANG;
  RMAG(i) = NUMMAG / DENMAG;
  Rph(i) = NUMANG - DENANG;
  MAG = RMAG(i);
  ANG = Rph(i);
  [x, y] = pol2rect (ANG, MAG);
  reR(i) = x;
  imR(i) = y;
  reT(i) = 1 + reR(i);
  imT(i) = imR(i);
  x = reT(i);
  y = imT(i);
  [ANG, MAG] = rect2pol (x, y);
  Tmag(i) = MAG;
  Tph(i) = ANG;

% Continue from i=2
  for i = 2:(N+1)
    x = rez(i) - rez(i-1);
    y = imz(i) - imz(i-1);
    [ANG, MAG] = rect2pol (x, y);
    NUMMAG = MAG;
    NUMANG = ANG;
    x = rez(i) + rez(i-1);
    y = imz(i) + imz(i-1);
    [ANG, MAG] = rect2pol (x, y);
    DENMAG = MAG;
    DENANG = ANG;
    RMAG(i) = NUMMAG / DENMAG;
    Rph(i) = NUMANG - DENANG;
    MAG = RMAG(i);
    ANG = Rph(i);
    [x, y] = pol2rect (ANG, MAG);
    reR(i) = x;
    imR(i) = y;
    reT(i) = 1 + reR(i);
    imT(i) = imR(i);
    x = reT(i);
    y = imT(i);
    [ANG, MAG] = rect2pol (x, y);
    Tmag(i) = MAG;
    Tph(i) = ANG;
  end

% Matrix multiplications begin here
Amag(1) = exp(-imphi(1) * t(1));
Amag(4) = 1 / Amag(1);
Amag(2) = RMAG(1) * Amag(4);
Amag(3) = RMAG(1) * Amag(1);
Aph(1) = rephi(1) * t(1);
Aph(2) = Rph(1) - Aph(1);
Aph(3) = Rph(1) + Aph(1);
Aph(4) = -Aph(1);

for K = 2:N
  Bmag(1) = exp(-imphi(K) * t(K));
  Bmag(4) = 1 / Bmag(1);
  Bmag(2) = RMAG(K) * Bmag(4);
  Bmag(3) = RMAG(K) * Bmag(1);
  Bph(1) = rephi(K) * t(K);
  Bph(2) = Rph(K) - Bph(1);
  Bph(3) = Rph(K) + Bph(1);
```

```
  Bph(4) = -Bph(1);
  [Amag, Aph] = cmult(Amag, Aph, Bmag, Bph);
end

Bmag(1) = 1;
Bmag(4) = 1;
Bmag(2) = RMAG(N + 1);
Bmag(3) = Bmag(2);
Bph(1) = 0;
Bph(2) = Rph(N + 1);
Bph(3) = Bph(2);
Bph(4) = 0;
[Amag, Aph] = cmult(Amag, Aph, Bmag, Bph);
tranmag = 1;
tranph = 0;

for j = 1:N +1
  tranmag = tranmag * Tmag(j);
  tranph = tranph + Tph(j);
end

tranmag = 1 / (tranmag + 1e-12);

while (tranph > (2 * pi))
    tranph = tranph - (2 * pi);
endwhile

tranph = -tranph;
Tw = 1 / (tranmag * Amag(1));
Ttotph = -(tranph + Aph(1));
Rw = Amag(3) / Amag(1);
SUM = 0;
for j = 1:N
  SUM = SUM + t(j);
end

% Get IPD into correct quadrant
do
  if(Aph(1) < (CTH * Ko * SUM))
    Aph(1) = Aph(1) + (2 * pi);
  end
until (Aph(1) >= (CTH * Ko * SUM))

IPD = Aph(1) - (CTH * Ko * SUM);
endfunction
```

## Function cmult

```
function [Amag, Aph] = cmult(Amag, Aph, Bmag, Bph)
  for i = 1:2:3
    for j = 1:2
      R1 = Amag(i) * Bmag(j);
      R2 = Amag(i + 1) * Bmag(j + 2);
      E1 = Aph(i) + Bph(j);
      E2 = Aph(i + 1) + Bph(j + 2);
      R3 = (R1 * cos(E1)) + (R2 * cos(E2));
      E3 = (R1 * sin(E1)) + (R2 * sin(E2));
      Cmag(j + i - 1) = sqrt((R3*R3) + (E3*E3));
      if((R3 > 0) && (E3 > 0))
        PA = 0;
      elseif(R3 < 0)
        PA = pi;
      elseif((R3 > 0) && (E3 < 0))
        PA = 2 * pi;
      end
      Cph(j + i - 1) = atan(E3 / (R3 + 1e-12)) + PA;
    end
  end

  for i = 1:4
    Amag(i) = Cmag(i);
    Aph(i) = Cph(i);
  end

endfunction
```

## Function complex_sqr_root

```
function [RMAG, RANG] = complex_sqr_root(MAG, ANG)
  RMAG = sqrt(MAG);
  RANG = ANG / 2;
endfunction
```

## Function pol2rect

```
function [x, y] = pol2rect (ANG, MAG)

  x = MAG * cos(ANG);
  y = MAG * sin(ANG);

endfunction
```

## Function rect2pol

```
function [ANG, MAG] = rect2pol (x, y)

  MAG = sqrt((x^2) + (y^2));

  if ((x>0) && (y>0))
    corr = 0;
  elseif ((x<0) && (y<0))
    corr = pi;
  elseif ((x>0) && (y<0))
    corr = 0;
  elseif (x<0) && (y>0)
    corr = pi;
  elseif (x>0) && (y==0)
```

```
    corr = 0;
  elseif (x<0) && (y==0)
    corr = pi;
  elseif (x==0) && (y!=0)
    corr = 0;
  elseif (x==0) && (y==0)
    ANG = 0;
    return;
  end

  ANG = atan(y / (x + 1e-12)) + corr;

endfunction
```

**Bonus Material**
**1D FDTD Simulation of the C-Sandwich Radome**

In addition to the analytical method described in this document the following is a Matlab/Octave code for a 1D FDTD (Finite Difference Time Domain) numerical calculation that produced the plot shown in the figure below. This gives further validation of the analytical method and provides an alternative way to simulate multilayer radomes. The FDTD code assumes a plane wave travelling in the z direction with a direction normal to the radome.

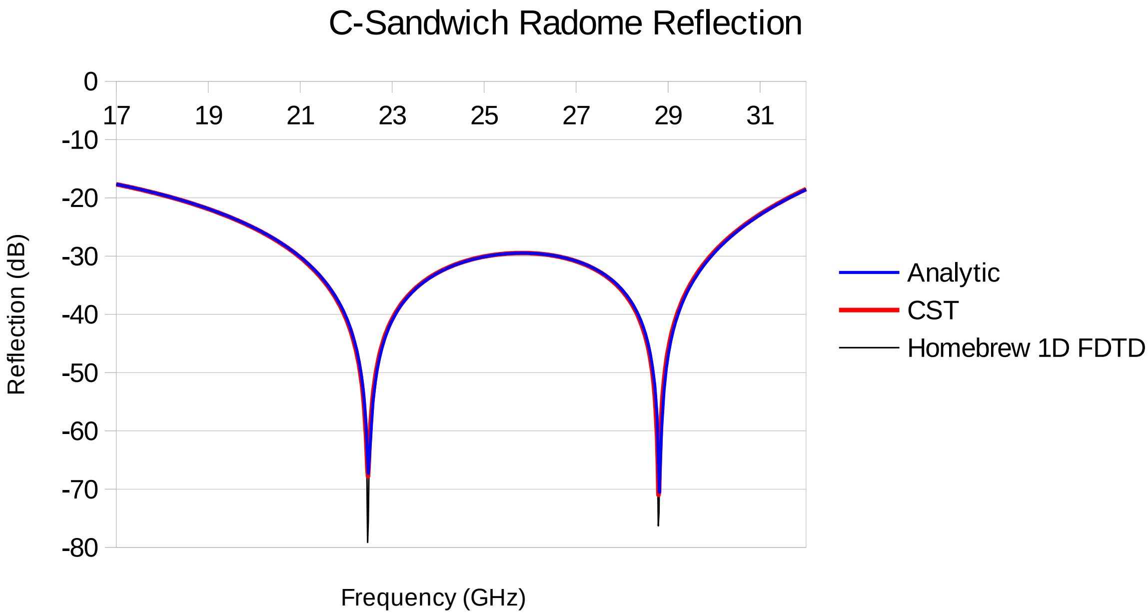



Figure. 6

The code listing for the FDTD simulation is shown on the following pages and the output is shown in Figure.7 noting that the simulation had converged.

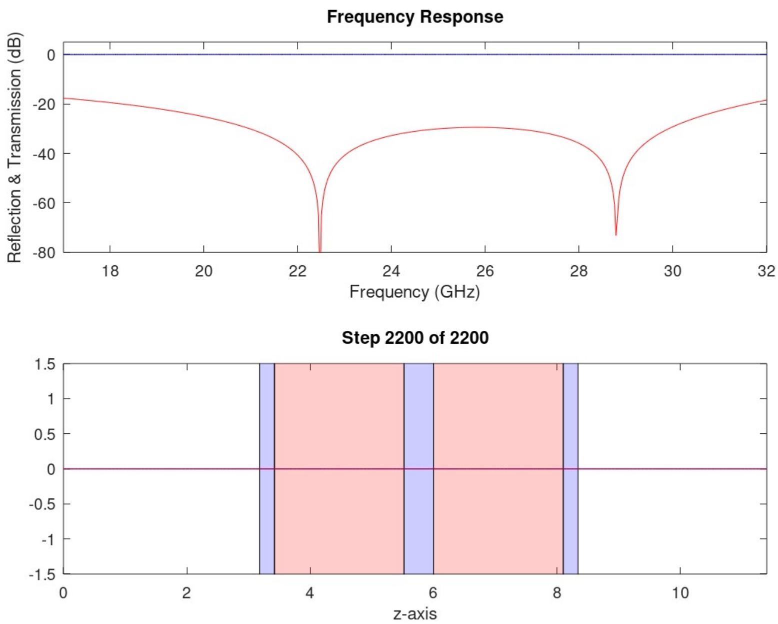



Figure. 7

```matlab
% Multilayer_Radome_1D_FDTD.m
% Calculate Reflectance and Transmittance of a multilayer radome
% Adapted from 1D FDTD course by Prof Rumph
% Author: Paul Klasmann, May 2021

% INITIALIZE MATLAB
close all;
clc;
clear all;

% UNITS
meters  = 1;
seconds = 1;

centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
inches      = 2.54 * centimeters;
hertz       = 1/seconds;
kilohertz   = 1e3 * hertz;
megahertz   = 1e6 * hertz;
gigahertz   = 1e9 * hertz;

% CONSTANTS
c0 = 299792458 * meters/seconds;
e0 = 8.8541878176e-12 * 1/meters;
u0 = 1.2566370614e-6 * 1/meters;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 1 -- DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SOURCE PARAMETERS
fmin  = 17.0 * gigahertz;
fmax  = 32.0 * gigahertz;
NFREQ = 500;                           % Number of frequency points for FFT
FREQ  = linspace(fmin,fmax,NFREQ);     % Setup frequency array

% DEVICE PARAMETERS FOR MULTILAYER RADOME
er1 = 1.1; % Millifoam
er2 = 4.0; % GFRP
er3 = 1.0; % Air
L1  = 2.1 * millimeters;
L2  = 0.24 * millimeters;
L3  = 0.48 * millimeters;

urmax = 1;                        % Non ferrous/magnetic material, ur=1

% GRID PARAMETERS
ermax = max([er1 er2 er3]);       % Find greatest dielectric constant
nmax  = sqrt(ermax * urmax);      % Greatest er used to calculate refractive index

DDAT  = [L2 L1 L3 L1 L2];         % Setup material length array
erDAT = [er2 er1 er2 er1 er2];    % Setup material dielectric constant array

NRES_LAM = 60;                    % Resolution in cells per wavelength
NRES_D   = 2;                     %
NSPC     = [50 50];               % Buffer of free space before and after device

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 2 -- CALCULATE GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CALCULATE INITIAL GRID RESOLUTION
```

```
Lmin    = min([L1 L2 L3]);  % Find minimum physical length
lam_min = c0/fmax/nmax;     % Calculate minimum lambda at fmax and in nmax
dz1     = lam_min/NRES_LAM; % Minimum dz based on resolution min lam / resolution
dz2     = Lmin/NRES_D;      % Find min dz based on Lmin physical dimention
dz      = min([dz1 dz2]);   % Choose dz to be smallest of dz1 and dz2

% SNAP GRID TO CRITICAL DIMENSIONS
nz = ceil(Lmin/dz)
dz = Lmin/nz

% CALCULATE NUMBER OF GRID CELLS
Nz = ceil(L2/dz) + ceil(L1/dz) + ceil(L3/dz) +ceil(L1/dz) + ceil(L2/dz)+ sum(NSPC) +
3

% CALCULATE ARRAY OF E FIELD POSITION
za = [0:Nz-1]*dz;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 3 -- MODEL DEVICE ON THE GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE MATERIALS
ERxx = er3*ones(1,Nz);  % er3 is free-space permitivity
URyy = ones(1,Nz);      % ur =  1 since non magnetic material

% CALCULATE START AND STOP INDICIES OF SLAB & INCORPORATE SLAB INTO ERxx
nz0 = 2 + NSPC(1) + 1;           % nz0 is start of radome
nz1 = nz0;                       % set nz1 to start index of radome
for nd = 1 : length(DDAT)
    nz2 = nz0 + round(sum(DDAT(1:nd))/dz) - 1; % Loop for start/stop of each layer
in DDAT
    ERxx(nz1:nz2) = erDAT(nd);        % Insert er for material
    x1     = nz1*dz/millimeters
    % Fill xx array with stop and start values for visualisation of layers
    xx(nd, 1) = x1;  % These lines draw closed rectangle with yy (fixed)
    xx(nd, 4) = x1;
    xx(nd, 5) = x1;
    % update next nz1 start index for next layer
    nz1    = nz2 + 1;

    x2     = nz1*dz/millimeters
    xx(nd, 2) = x2;
    xx(nd, 3) = x2;
end
%xx    % Uncomment to check xx array for visualisation of layers

% CALCULATE SLABS TO PLOT FOR VISUALISATION
yy = [ -2 -2 +2 +2 -2 ];  % Disitance of y dimension should be heigher than plot

%plot(za/millimeters,ERxx) % Uncomment to plot er of structure layers

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 4 -- CALCULATE THE TIME STEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DETERMINE REFRACTIVE INDEX AT BOUNDARIES
nbc = sqrt(URyy(1)*ERxx(1));

% CALCULATE TIME STEP FOR PABC
dt = nbc*dz/(2*c0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 5 -- CALCULATE THE SOURCE
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% POSITION OF SOURCE
k_src = 2;    % 2nd cell into simulation domain, from the left.

% CALCULATE GAUSSIAN PULSE PARAMETERS
tau = 0.5/fmax;
t0 = 3*tau;

% CALCULATE TIME ARRAY
N = 2200;
t = [0:N-1]*dt;

% CALCULATE Ex SOURCE
Exsrc = exp(-((t - t0)/tau).^2);

% CALCULATE Hy SOURCE
nsrc  = sqrt(ERxx(k_src)*URyy(k_src));
s     = 0.5*nsrc*dz/c0 - dt/2;
A     = sqrt(ERxx(k_src)/URyy(k_src));
Hysrc = A*exp(-((t - t0 + s)/tau).^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 6 -- CALCULATE UPDATE COEFFICIENTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CALCULATE UPDATE COEFFICIENTS
mEx = -(c0*dt/dz)./ERxx;
mHy = -(c0*dt/dz)./URyy;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 7 -- INITIALIZE FOURIER TRANSFORMS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CALCULATE KERNALS
K = exp((-1i*2*pi*dt)*FREQ);

% INITIALIZE FOURIER TRANSFORM ARRAYS
ExR = zeros(1,NFREQ);
ExT = zeros(1,NFREQ);
SRC = zeros(1,NFREQ);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STEP 8 -- INITIALIZE FIELDS TO ZERO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE FIELDS TO ZERO
Ex = zeros(1,Nz);
Hy = zeros(1,Nz);

% INITIALIZE BOUNDARY FIELD TERMS
E1 = 0;    E2 = 0;
H1 = 0;    H2 = 0;

%
% MAIN FDTD LOOP -- ITERATE OVER TIME
%
for n = 1 : N

  % Step a -- Update Ex Boundary Terms
  E2 = E1;
  E1 = Ex(Nz);
```

```
% Step b -- Update Ex
Ex(1) = Ex(1) + mEx(1)*(Hy(1) - H2);  % With Direchlet BC so Hy(0) = 0
for k = 2 : Nz
  Ex(k) = Ex(k) + mEx(k)*(Hy(k) - Hy(k-1));
end

% Step c -- Incorporate TF/SF Correction Term
Ex(k_src) = Ex(k_src) - mEx(k_src)*Hysrc(n);

% Step d -- Update Hy Boundary Terms
H2 = H1;
H1 = Hy(1);

% Step e -- Update Hy

for k = 1 : Nz -1
  Hy(k) = Hy(k) + mHy(k)*(Ex(k+1) - Ex(k));
end
Hy(Nz) = Hy(Nz) + mHy(Nz)*(E2 - Ex(Nz));  % With Direchlet BC so Ex(k+1) = 0

% Step f -- Incorporate TF/SF Correction Term
Hy(k_src-1) = Hy(k_src - 1) - mHy(k_src - 1)*Exsrc(n);

% Step g -- Update Fourier Transforms
for nf = 1 : NFREQ
  ExR(nf) = ExR(nf) + (K(nf)^n)*Ex(1);
  ExT(nf) = ExT(nf) + (K(nf)^n)*Ex(Nz);
  SRC(nf) = SRC(nf) + (K(nf)^n)*Exsrc(n);
end

% Step h -- Visualize Simulation
if ~mod(n,50)

  % Calculate Spectra
  REF = abs(ExR./SRC).^2;
  TRN = abs(ExT./SRC).^2;
  CON = REF + TRN;

  % Prepare Figure Window
  clf;

  subplot(212);
  % Draw the Rectangle
  for nd = 1 : length(DDAT)
  if ~mod(nd,2)
     fill(xx(nd,[1 2 3 4 5]),yy,'r','FaceAlpha',0.2);      %% loop for slabs
  else
     fill(xx(nd,[1 2 3 4 5]),yy,'b','FaceAlpha',0.2);
  end
  hold on;
  end
  % Plot Fields
  plot(za/millimeters,Ex, '-b');
  hold on;
  plot(za/millimeters,Hy, '-r');
  hold off;
  xlim([za(1) za(Nz)]/millimeters/meters);
  ylim([-1.5 1.5]);
  xlabel('z-axis');
  title(['Step ' num2str(n) ' of ' num2str(N) ]);
  % Show Spectra
  subplot(211);
  plot(FREQ/gigahertz,10*log10(REF), '-r');
```

```
    hold on;
    plot(FREQ/gigahertz,10*log10(TRN), '-b');
    plot(FREQ/gigahertz,10*log10(CON), ':k');
    hold off;
    xlim([fmin fmax]/gigahertz);
    ylim([-80 5]); % Change limits to suit...
    xlabel('Frequency (GHz)');
    ylabel('Reflection & Transmission (dB)');
    title('Frequency Response');
    drawnow;
  end
end
```

END OF DOCUMENT