Measuring Simulated Reflectance from a Surface Using XFDTD 7.3.2.7

Katherine Han

10-5-14

**Introduction**

XFDTD by Remcom is a software designed to simulate electromagnetic fields, primarily for antenna design and analysis. This software is based on the finite difference time domain numerical method for optical modeling. This method solves Maxwell’s curl equations that have been discretized using a central difference theorem. The output from this program is electric fields in the x, y, and z direction at each point over the space specified (usually a point or plane sensor) and/or corresponding magnetic field values.

XF can be used to analyze the anti-reflective properties of an anti-reflective coating or texture. The following document provides the means to determine the percent reflected or transmitted of light through an ARC.

**Tools**

* Active copy of XFDTD 7.3.2.7 or similar compatible version
* Active copy of MATLAB 2012 or similar compatible version
* NVIDIA Tesla GPU (recommended)
* The following files:
  + DataPath.xlsx
  + ImportAndFFTFreeSpace\_step0.m
    - PerformFFTFreeSpace.m
  + ImportAndFFT\_step1.m
    - Ff\_readXFssData.m
    - PerformFFT.m
    - ImportAndFFT\_MakeFigures.m
  + AnalyzeDataRefl\_step2.m
    - AM1.5.mat
  + Output results from XF

**Overall Procedure**

The overall actions in this procedure are to simulate light hitting a surface in XF, output the E-field and H-field intensities of the reflected and transmitted waves, Fourier transform these values at each point over the computational domain and transform the values to light power, average the light intensity values over all points for each wavelength of light in the simulation, and compare the transmitted and reflected light to the input light, giving a percent of total for each wavelength. The simulation of the light traveling through the computational domain and output of E-field and H-field intensities is done in XF and the post-processing is done in MATLAB.

Calculating the power of each wavelength that travels across a boundary or sensor can be done by calculating the Poynting vectors across each point in that sensor. The Poynting vector is calcuted using frequency-domain amplitude values of the E and H fields. Transforming E-field or H-field values from amplitude in the time domain to that in the frequency domain is done as follows:

Eqn. 1

where <P> is the average power for each wavelength crossing the sensor over the length of the simulation, Re is the real portion of the complex number, Ex is the complex result of the discrete Fourier transform (DFT) of the Ex values at one point over all time points, Hy\* is the complex conjugate of the DFT result for Hx values, the integral indicates a summation over all x and y points, x is the number of space points in the x direction, and y is the number of space points in the y direction.

For normal incidence light, as seen with 1-D simulations or ARCs significantly sub-wavelength that they do not induce any off-axis diffractions, power can be calculated simply by P = n|FFT(E)|2. However, as soon as any significant off-axis reflected or transmitted diffracted light is present Equation 1 must be used to produce accurate power calculations.

**Step-by-step Procedure**

Recent improvements in the post processing software files now allow quantification of any size of computational domain for any wavelength range. Several aspects of the XF set-up and output should be noted in order to successfully run the post processing software.

The first step to quantitative analysis of the power of reflected and transmitted light in a system is to set up a blank simulation that will allow transmission of 100% of all wavelengths of light in your desired wavelength range. To do this set up a simulation according to the geometry of Figure 1, making sure to record both E and H fields (Definitions ->Sensor Data Definitions -> Planar Sensor Definition -> check on all of E, Scattered E, H, Scattered H). To simulate free space assign an index of refraction of n=1 to all geometry parts. The +/- z boundaries should be PML and the +/- x and y should be periodic. The angle of incidence must, at this point, be zero (light should come from the +z direction). A convergence criteria of -30 dB will normally give less than 0.1% error, which is usually sufficient. Put the maximum simulation time as “100000\*timestep” or whatever number of time steps you would like the simulation to max out at. Generally between 5,000 and 50,000 is sufficient. Run the simulation with your desired wavelength range, making sure that the transmitted plane sensor is in a location where it will capture all of the transmitted light. The light must come from the +z direction. Now log the following information in the “Free Space Raw FDTD Data” tab of the DataPath.xlsx file:

1. Simulation folder path
2. Title for this particular simulation (suggested: something like “Free Space 700 to 1200 nm with 5 nm mesh”)
3. Index of refraction where the transmitted light sensor is located (the reflected light is always in free space)
4. Number of time steps (run length)
5. dT (time step size in seconds)
6. Whether or not you would like MATLAB to process the data (“to run,” either a 0 for no or a 1 for yes)
7. Minimum wavelength (nm)
8. Maximum wavelength (nm)
9. Data X length (number of data points in the x direction)
10. Data Y length (number of data points in the y direction)
11. Data storage path (where you want to save your analyzed data)
12. Reflected file stub (the beginning of the file name for each of your reflected light sensor time points)
13. Transmitted file stub (the beginning of the file name for each of your transmitted light sensor time points)

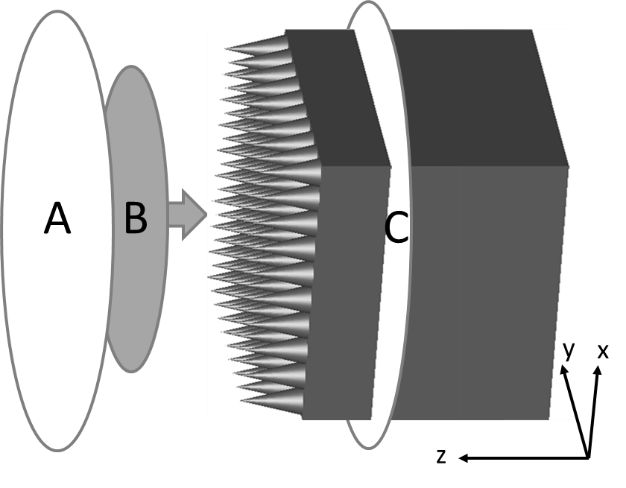


Figure : “A” indicates the location of the reflected light sensor, “B” is the input waveform, which is located eight mesh points inside the computational domain, and “C” is the transmitted light sensor.

Once the “free space” simulation has completed in XF and all of the relevant data has been logged in DataPath.xlsx, go into MATLAB and run the file “ImportAndFFTFreeSpace\_step0.m”. This only needs to be done once for each wavelength range, unless you want to explore the effects of different mesh sizes. The path to the correct free space data will need to be noted when running an analysis of one of your experimental simulations.

The XF simulation of your ARC should be set up as shown in Figure 1. There should be some space padding on the top of the simulation and the reflected light sensor must be within the very top 8 cells, as the light will be injected 8 cells from the very top. Light input should be a plane polarized total field/scattered field that is normal to the surface (angle of incidence of zero). Properties of the experiment that must be noted, and then input into the DataPath.xlsx file in these associated columns, are:

1. Simulation folder path
2. Title for this particular simulation
3. Index of refraction where the transmitted light sensor is located (the reflected light is always in free space)
4. Number of time steps (run length)
5. dT (time step size in seconds)
6. Whether or not you would like MATLAB to process the data (“to run,” either a 0 for no or a 1 for yes)
7. Minimum wavelength (nm)
8. Maximum wavelength (nm)
9. Data X length (number of data points in the x direction)
10. Data Y length (number of data points in the y direction)
11. Free space file path (specific to each different wavelength range)
12. Data storage path (where you want to save your analyzed data)
13. Reflected file stub (the beginning of the file name for each of your reflected light sensor time points)
14. Transmitted file stub (the beginning of the file name for each of your transmitted light sensor time points)

The XF simulation should run to completion of -30 dB or so. Once the simulation is finished and all of the columns in DataPath.xlsx sheet “Raw FDTD Data” have been filled out, then run ImportAndFFT\_step1.m to DFT the data (convert from time-based to frequency based). The output of this file will be three graphs that give you an idea of your results. The first graph is an intensity graph for the range of wavelengths in question that compares your transmitted, reflected, and free space data (see Figure 2). If you used an automatic form for light input in XF then this graph should be generally bell curve shaped. As long as you have no absorption or light trapping then the reflected and transmitted intensities should add up to 100%, or the same as the free space simulation.

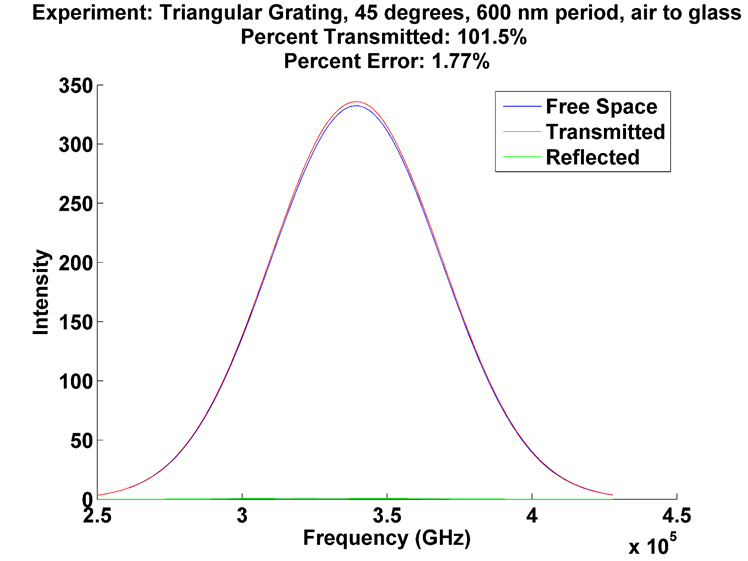
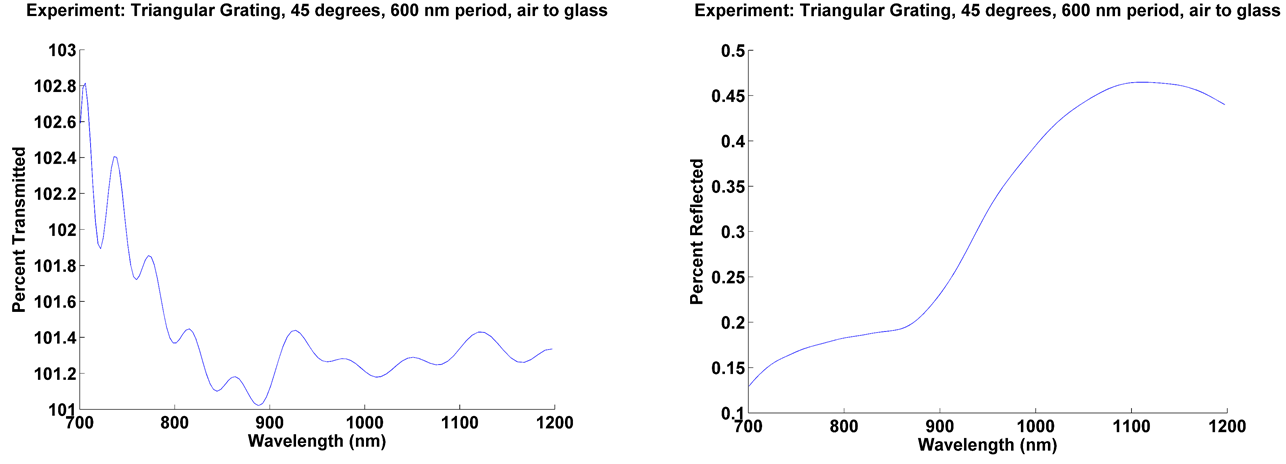


Figure 2: Intensity vs. frequency output of the DFT result from MATLAB.

ImportAndFFT\_step1.m will also output two more graphs, as seen in Figure 3. These graphs are created by calculating the percentage of transmitted (left) and reflected (right) light through the simulation. Due to numerical error the light can add up to more than 100%, but should be close to 100% for a more accurate simulation. Accuracy can be improved by decreasing mesh, decreasing the dB (from -30 to -40 or so) to which the experiment is run before termination, or moving the sensors further from the geometry.



Figure

Your data is now available to make into a graph. This step is easy once you get the hang of it, but does require that you fiddle with the MATLAB code. To make your graphs open “M:\Kat FDTD Data\MATLAB Functions\AnalyzeDataRefl\_step2.m”. At the bottom of this file there is a lot of commented code that can be used as an example on how to make the graphs. Here is a set of code that will make a graph, with explanation:

% %=========================375 nm Cylinders and thin film=====%Green text (after % sign) is a comment

Figure

hold on

plot(WaveLength, percentReflArray(51,:).\*100,'-k','LineWidth',2) % Bare 1.5 material (control)

plot(WaveLength, percentReflArray(168,:).\*100,'--g','LineWidth',2) % Describe run here so you don’t forget

plot(WaveLength, percentReflArray(169,:).\*100,'--b','LineWidth',2)% Description, notice the \*100 to use percent rather than the normalized scale of 1=100%

plot(WaveLength, percentReflArray(170,:).\*100,'-b','LineWidth',2)% Description this line will turn out to be blue and solid because of the ‘-b’

plot(WaveLength, percentReflArray(171,:).\*100,':b','LineWidth',2)% Description, notice this is line 171, which corresponds to the 171th row of data in the DataPath.xlsx file

plot(WaveLength, percentReflArray(172,:).\*100,':r','LineWidth',2)% Description

plot(WaveLength, percentReflArray(173,:).\*100,'--m','LineWidth',2)% Description

plot(WaveLength, percentReflArray(174,:).\*100,'-r','LineWidth',2)% Description

plot(WaveLength, percentReflArray(175,:).\*100,'-g','LineWidth',2)% Description

title ('5 Layer Thin Film With and Without Imperfections')

% legend(char([strLegend(1) strLegend(17:28)]),'Location','Best'); %You may choose to include a legend

ylim([0 5])

xlabel ('Wavelength (nm)');

ylabel ('Percent Reflected');

%Sets the font to 16, bold and saving the figure

set(findall(gcf,'-property','FontSize'),'FontSize',16)

set(findall(gcf,'-property','FontWeight'),'FontWeight','bold')

print(gcf,'-dpng','-r600',['5LayerImperfections\_3-13-14.png'])

%print(gcf,'-dpng','-r600',['M:\Kat FDTD Data\Figures\5LayerImperfections\_3-13-14.png']) % use this code to save your graph automatically to the folder indicated with the name given here.

AnalyzeDataRefl\_step2.m will also output a weighted average of reflectance that is weighted against the AM1.5 spectrum. This tells you how much power your material would lose outside when the sun is shining on it. To find this information you can open the output matrix “avgWeightedReflArray” that can be found in the Workspace. This will be a one row matrix that has a column for each run in your DataPath.xlsx file. To get percentages multiply these values by 100, as they are currently normalized to “1” being “100%”.

If you want to run several simulations in series without hanging out at your computer you can use the batch file included with the other files. Right now it’s called 12\_29\_13\_batch.txt. To use this open the file in Notepad and save it under a different name (I like to use the date and possibly some other information after that). Make sure you are referencing the correct directory (I use M: ) Change the code to reference the simulations you want to run (it will run all of the runs in each simulation you designate). Make sure the code referencing XF is correct (location, edition of the software, xstream for your Tesla GPU). Delete any extra simulations that might be left over from last time you ran a batch file. Save the file as a .bat. Go to the folder to which you saved the file and double click. There, now your computer can be hard at work running dozens of simulations, filling up your hard drive, while you sleep soundly, defying your graduate student status.

You now have the information you need to take the E-field and H-field outputs from XF simulations of anti-reflective coatings and process that data into percent reflected and transmitted graphs by frequency of light.