Myelin Sheath Waveguides:
Guide to Code and Simulation Data saved in GitHub repository

Emily Frede

March 6, 2023

1 Introduction

For any additional questions, I will try to be reasonably accessible via email, please use the Ucalgary address below.

emily.frede@ucalgary.ca

This document explains the files contained in the GitHub repository: ResearchProj_MyelinSheathWaveguides.

This repository contains all the code and data currently used for the project of studying polarization evolution and transmission in multi-Ranvier-node myelin sheath waveguides. This project used ANSYS Lumerical FDTD 3D Electromagnetic Simulator software to solve Maxwell's equations for the myelinated axon model, in order to obtain electric field and transmission results.

The myelinated axon model is composed of a cylinder of radius 0.6 μm to model the axon, and a cylindrical ring of inner radius 0.6 μm and outer radius 1 μm to model the myelin sheath. Optical properties of the axon, myelin sheath, and surrounding interstitial fluid are described in the paper draft. The refractive index of the myelin is the highest and we see in this simulation data, as expected, that light can be confined to this structure and may be guided through realistic distances. The myelinated axon model is 500 μm in length, with 2 μm gaps in the myelin sheath located at $z = 100, 200, 300, 400\,\mu$m to simulate Ranvier nodes. The myelinated axon is centered in the simulation region, which is 4 μm x 4 μm x 500 μm. A mode source for the input light ($\lambda = 0.4\,\mu$m) is located at the beginning of the simulation region at $z = 0\,\mu$m.

Note that during the data collection process, I used two different models – the homogeneous / non-birefringent myelin model, and the birefringent myelin model. In the homogeneous myelin model the refractive index of the myelin was taken to be uniform. In the birefringent myelin model I attempted to model the biological positive radial birefringence of the myelin sheath. The birefringent modes, simulation data, and relevant code – including the MATLAB code to generate matrix transform data for input into the Lumerical software model – are labelled in folders with "Biref" in the title. **Note that due to the weird appearance of the birefringent modes, it has presently been decided to exclude the birefringent results from the paper.** If one repeats these simulations in the future, it may be of interest to investigate why the birefringent results look this way. I have performed a lot of troubleshooting to investigate this, I believe that the software is providing a weird, non-standard basis of modes due to mode degeneracy.

2 Important Documents

I have made a folder called "ImportantDocuments" which contains important text documents related to this project. It includes the current paper draft "MyelinSheathWaveguides_Paper+Appendices_EF.pdf" and the associated supplemental information "MyelinSheathWaveguides_Supplemental_EF.pdf". If one attempts any analysis or reproduction of these results, these three documents – including this guide to the data and code – are the best resources for the project.

This folder also includes a useful spreadsheet "mode_info.xlsx" which records all the effective index data for all (both homogeneous and birefringent) modes – this information is included elsewhere but it is in a nice visual form here.

3 Simulation data and code to generate figures

Simulation data is collected from the mode source at $z = 0\,\mu$m, and at monitors spaced every $25\,\mu$m along the length of the model. At each cross-section, the mode source and the monitors, I have recorded and saved all possible simulation data. The most important data for this project is the electric field data, in which we can observe magnitude and vector field. We also use the transmission data at each monitor (at the mode we could simply say $T = 1$).

Lumerical script files were used to save the simulation data from the Lumerical software to MATLAB data files. I have also included these script files (as text files with the ending .txt) so one can see how the variables in the software were saved. Aside from a minor nuance in the data files with the ending "_Efield.mat", I have saved each software variable as the same name in the MATLAB files – e.g., "E" is the electric field, "H" is the magnetic field, "T" is the transmission coefficient, and so on.

The Lumerical FDTD product reference manual (located at the link below as of March 6, 2023) is very useful for understanding what each of these variables mean.

https://optics.ansys.com/hc/en-us/articles/360033154434-FDTD-product-reference-manual

Here I also highlight the MATLAB code which processes the simulation data files to produce the figures as currently seen in the paper draft (and supplemental), which are the electric field magnitude and vector plots.

3.1 Modes at $z = 0\,\mu$m source

The simulation data files for the modes calculated at the $z = 0\,\mu$m cross-section are located in the folder "SimulationModes". (For the birefringent case, this data is equivalently located in the folder "SimulationModesBiref". The following discussion applies to the birefringent data as well.)

The "SimulationModes" folder contains a file called "ModeInfoNB.txt" which is the manual collection of eigenvalue data regarding the calculated modes – most notably this contains the effective index for each mode which is analogous to the eigenvalue of the mode. The modes are ordered according to decreasing effective index (mode 1 has the highest effective index, mode 12 as a highest-order mode has the lowest effective index).

The "SimulationModes" folder also contains a file called "READ_SOURCE_DATA_final.txt". This is the text file of the Lumerical script which was used to save the data files for each of the modes.

The folder "SimulationModes" contains 12 folders for the first twelve modes calculated. The data for the first mode is

located in folder "mode1", data for the second mode is located in folder "mode2", and so on. In each of the folders for the modes, there are three files descriptively labelled as to which mode it is (and that it is the non-birefringent case). As may be observed in the "READ_SOURCE_DATA_final.txt" file, or simply by opening these data files in MATLAB, they save different variables. Every variable from the software is saved as under the same name in the MATLAB data file – with the minor exception of the file with the ending "_Efield.mat".

The files ending in "_Efield.mat", as seen here in "READ_SOURCE_DATA_final.txt", contains the variables Ex, Ey, Ez, x, y, z, x2, y2, z2. (Note that for the mode source, z = z2 = 0 will always be true as this corresponds to the location $z = 0\,\mu$m.) Variables x, y, z are coordinates of the simulation grid that was used to generate the results – note that this is not uniform because the Lumerical software uses an adaptive mesh which increases resolution around material interfaces for greater accuracy (and maintaining computational efficiency). The variables x2, y2, z2 are generated in "READ_SOURCE_DATA_final.txt" and they are the coordinates of a uniform grid for the simulation region. The variables Ex, Ey, Ez are the electric field components defined on the grid x2, y2, z2. These variables were found by taking the original electric field components, defined on the original grid x, y, z, and doing a linear interpolation to find the electric field components on the uniform grid x2, y2, z2.

The discussed linear interpolation is seen in the Lumerical script "READ_SOURCE_DATA_final.txt" that produces this data file, and uses the linear interpolation function interp() which can be read about at the link below.

https://optics.ansys.com/hc/en-us/articles/360034925893-interp-Script-command

In the "SimulationModes" folder, there is also another folder called "GeneratingFigures". In the "GeneratingFigures" folder, there are two folders entitled "MagnitudePlots" and "VectorPlots". In these folders, I have saved all the good copy electric field magnitude plots and electric field vector plots as currently presented in the paper draft (and the supplemental information document). These are of the type ".eps" which means they are vector, in contrast to raster/pixel, images (so increasing their size won't decrease their quality). To view them, list them as a figure in a Latex document. Also in the "GeneratingFigures" folder I have included the code "generate_mode_magnitude_and_vector_plot.m". This is MATLAB code which can load a MATLAB data file with the ending "_Efield.mat" and generate two ".eps" files. When running this script, ensure that the file reference to the data file with the ending "_Efield.mat" is correct. This code run repeatedly to generate all the magnitude and vector plots in those folders within the "GeneratingFigures" folder.

3.2 Monitors at cross-sections every $25\,\mu$m

The simulation data files for the monitors, which are located every $25\,\mu$m along the model starting at $z = 25\,\mu$m and ending at $z = 500\,\mu$m, are located in the folder "SimulationMonitorData". (For the birefringent case, this data is located in the folder "SimulationMonitorDataBiref". The following discussion applies to the birefringent data as well.)

The "SimulationMonitorData" folder contains a file called "READ_MONITOR_DATA_final.txt". This is the text file of the Lumerical script which was used to save the data files for each of the monitors.

The "SimulationMonitorData" folder contains many subfolders for each monitor, they are labelled by the z coordinate of the monitor. For instance, the "25" folder corresponds to the monitor at $z = 25\,\mu$m and contains files which save all the data the monitor recorded. These data files are saved by "READ_MONITOR_DATA_final.txt" and simply save the variables from the software, keeping the names of the variables the same, except for files with endings "_Efield.mat", "_fullResE.mat" and "_partialResE.mat". Files with the last two endings I have listed above are not needed, they were generated to test post-simulation processing capabilities of the Lumerical software. As discussed earlier, files with the ending "_Efield.mat" save

the variables Ex, Ey, Ez which are the linearly-interpolated electric field components onto the uniform grid defined by variables x2, y2, z2.

The "SimulationMonitorData" folder also contains a folder called "GeneratingFigures". The folder "GeneratingFigures" includes the folders called "MagnitudePlots" and "VectorPlots". These two folders contain the all good copy vector images of the electric field magnitude plots and the electric field vector plots. They were generated by the code "generating_monitor_magnitude_and_vector_plots.m" which processes a monitor data file with the ending "_Efield.mat". Again, if you are running this code check to ensure the reference to the data file is correct.

4 Matrix transform used to implement radial birefringence

To implement radial birefringence in the Lumerical software, one needs to use a matrix transform grid attribute. This is necessary because the refractive index matrix of the radially-birefringent myelin is not diagonal in Cartesian coordinates (although Cartesian is the software's default). More information about the matrix transform in the Lumerical software can be found at the link below.

https://optics.ansys.com/hc/en-us/articles/360034915173-Matrix-Transformation-Simulation-object

The matrix transform will specify, for each point in a three-dimensional grid placed upon the model, 3 unit vectors to define the coordinate system at that point. Our matrix transform specifies cylindrical coordinates for points within the myelin sheath. In this way we can directly specify a diagonal refractive index matrix for the myelin sheath, where we take the elements to be n_{rr}, $n_{\phi\phi}$, and n_{zz}. The matrix transform will map these three cylindrical/diagonal matrix elements into a full Cartesian matrix (which will have nonzero matrix elements – specifically n_{xy} and n_{yx}. More discussion of the matrix transform and how this is implemented is discussed in Appendix B of the current paper draft.

All relevant files for the matrix transform are included in the folder "MatrixTransformBiref".

The text file "importing_grid_attribute_FINAL.txt" contains the Lumerical script used to import the matrix transform MATLAB data file into the Lumerical software. Be sure to check that you have the correct file location, the essential file I use is called "UData_myelinRad1_res801.mat" and in the GitHub repository it is located within the folder "custom_transform_for_myelin_model_FINAL".

The folder "custom_transform_for_myelin_model" contains all the MATLALB code used to generate the essential matrix transform data file "UData_myelinRad1_res801.mat". This folder contains the main script called "generate_custom_transform.m", within which parameters can be specified (myelin radius, axon radius, grid resolution, refractive indices, etc.), and when run will produce the files "UData_myelinRad1_res801.mat" and "nData_myelinRad1_res801.mat". The first file, which is also located in this folder, contains the matrix transform data.

The file "nData_myelinRad1_res801.mat" contains the custom refractive index data for generating custom material. It is too large to upload to GitHub so it is not included in the folder. I did not use the refractive index data file whatsoever in the final simulations, as our case still allowed for direct specification of refractive index matrices in the software and I found this to have greater accuracy on the interfaces (fluid-myelin, and myelin-axon). If you want to investigate using this custom refractive index data into the Lumerical software, there is information and instructions at the link below.

https://optics.ansys.com/hc/en-us/articles/360034901993-Spatial-n-k-data-Simulation-object

The main script "generate_custom_transform.m" utilises many function files to produce these two data files, all these functions are included in the folder as well – the complete list being "getRegionCoords.m", "applyConstantIndex.m", "applyRadialBirefringence.m", "organizeToGrid.m", "generateMatrixFigure.m", and "writeToFile.m". The main script and the functions have each been commented in detail to make them easier to follow. When running the main script, it will likely take a few minutes for it to complete. This runtime will increase with grid resolution (currently set to 801). If one would like to make this shorter, they can comment out the lines of code within the "writeToFile.m" function that are used to produce the "nData_myelinRad1_res801.m" file. (Writing the refractive index data file takes up a good deal of the time for the script to run, so the lines can be eliminated if one is only interested in providing high-resolution matrix transform data.)

5 Ideas for quantifying polarization change between two cross-sections

The current idea that has been suggested to compare the polarization change between two cross-sections is to use an overlap integral. This metric (I've denoted it m here) would depend upon the electric field $\vec{E}$ as follows, where a quantity of 1 should correspond to identical polarization states.

$$m = \frac{\left| \int d^2x \vec{E}_{initial} \cdot \vec{E}_{final} \right|}{\sqrt{\int d^2x \vec{E}^2_{initial} \int d^2x \vec{E}^2_{final}}} \tag{1}$$

Note also that for a given polarization state, the electric field oscillates as it travels through space. So if it has azimuthal polarization, it will alternate between clockwise and counter-clockwise electric field vectors. If it has radial polarization, it will alternate between outward-pointing and inward-pointing electric field vectors. These changes are just due to propagation of the electric field, not change in polarization. So the top expression on the right-hand side has an absolute value which makes the metric ignore this. Note the bottom expression on the right-hand side is used to normalize the metric. If the initial and final polarization states are the same, I expect this metric to have a value of 1, if they are different and orthogonal, I expect this metric to have a value of 0.

I have included the start of a simple comparison between two states in the GitHub repository in the folder "QuantifyingPolarizationChangeExample". This folder includes two data files from the cross-sections at $z = 75\ \mu$m (before the first Ranvier node) and $z = 175\ \mu$m (after the first Ranvier node). This folder also contains a MATLAB file which contains the beginning of a script that can be used to calculate this metric using these two data files.

The two data files included in this folder "QuantifyingPolarizationChange" are located in the appropriate folders within the folder for simulation data "SimulationMonitorData". These files are titled "data_75microns_Efield.mat" and "data_125microns_Efield.mat". These are data files that have the ending "_Efield.mat", and as described above they contain the vatiables Ex, Ey, Ez, x, y, z, x2, y2, z2. The variables Ex, Ey, Ez are the electric field components at each point on the uniform coordinate grid defined by x2, y2, z2.

The script "comparing_polarization_script.m" loads these two MATLAB data files (make sure that the file location is correct, it is currently set up such that the script and the two data files should be in the same folder to run). Note that I save the electric field variables from each file in new descriptive variables to prevent the first set from being overwritten, since the variables have the same names in the data files (i.e., loading the second file will overwrite the Ex matrix from the $z = 75\,\mu$m file with the Ex matrix from the $z = 125\ \mu$m file). Between the files x2, y2, z2 will be the same – which should make comparison of the electric field easy, since there is a 1-to-1 mapping from one cross-section to the next. I have scripted

three lines of dot product for each of the components (x, y, z), which is the first step to computing the metric between these two data files.

A final note – when comparing polarization states between cross-sections, I recommend choosing cross-sections that are not located at the edge of a Ranvier node, i.e., at $z = 100, 200, 300, 400\,\mu$m. At these locations, monitors are "snapped" to the nearest point in the simulation mesh and so these cross-sections may be slightly within the Ranvier node.