Waveguide-FDFD: User Guidelines  
Ralf Mouthaan

**Introduction**

Waveguide-FDFD is a Matlab-based finite difference frequency domain (FDFD) routine for calculating propagation-invariant modes supported by arbitrary geometry waveguides such as photonic crystal fibres. The profile of the fibre is defined, and guided modes are obtained as well as the associated propagation constants.

Waveguide-FDFD assumes a translationally-invariant geometry to express Maxwell's equations on a 2D Yee cell as done by Zhu & Brown [1] and by Yu & Chang [2]. Boundary conditions are implemented based on [3]. The implemented algorithms as well as benchmarks are described this paper [4] as well as my thesis [5].

Waveguide-FDFD performs a similar calculation to Lumerical’s FDFD solver, although Waveguide-FDFD runs faster. This is thanks to the use of sparse matrices as well as the strength of Matlab’s eigensolver.

This document aims to get a new user started with running a simulation using Waveguide-FDFD. The inner workings of Waveguide-FDFD are not discussed, but some Matlab knowledge is assumed.

**Downloading Waveguide-FDFD**

Waveguide-FDFD is available under an MIT license in the following github repository: <https://github.com/ralfmouthaan/Waveguide-FDFD>. Those familiar with Git can clone the repository. Those less familiar with Git can instead click on the “Code” button and select “Download ZIP”.

**Directory Overview**

The main directory contains the following FDFD implementations:

* *ModeSolverFD:* This is the Matlab file that contains the basic implementation of the FDFD code. It solves for vector fields (Ex, Ey, Ez, Hx, Hy, and Hz), and assumes a refractive index profile that is not birefringent (nx = ny = nz).
* *ModeSolverFD\_Anisotropic:* This FDFD implementation is for birefringent media (ny <> ny <> nz).
* *ModeSolverFD\_LP:* This FDFD implementation solves for linearly polarised modes only. This is more efficient and less memory-intensive.

The main directory also contains the following functions that give examples of how data can be displayed after *ModeSolverFD* has been run:

* *ModePlotter\_Magnitude:* This function plots the magnitude of the electric fields of a given mode.
* *ModePlotter\_Quiver:* This function displays a quiver plot for a given mode profile. A quiver plot captures both magnitude and E-field directionality.
* *PropConstantPlotter:* This function plots the real and imaginary components of the propagation constants of the obtained modes. Real and imaginary components are both plotted. The imaginary component corresponds to loss and are converted from Np/m to dB/m by multiplying by 8.68.

The Examples folder contains a folder of different example calculations.

The LP Eigenequations folder contains a LateX file with the derivation for the *ModeSolverFD\_LP* equations. This work is novel and should probably be published at some point.

The References folder contains pdfs of the papers I referred to when developing this work.

**ModeSolver\_FD**

A copy of the *ModeSolverFD.m* file must be present in the folder you are working in. There is no need to open it, and it does not rely on other files. It merely needs to be in your working directory. This file defines a single function:

[RetVal] = ModeSolverFD(dx, n, lambda, beta, NoModes)

The user passes the following parameters into the function:

* dx, real scalar. Coordinate step size, i.e. distance in metres between neighbouring points.
* N, matrix of reals. Transverse refractive index profile.
* lambda, real scalar. Operating wavelength in meters.
* beta, real scalar. Estimate of the real component of the propagation constant of the fundamental mode.
* NoModes, real scalar. Number of modes we wish to calculate. The function starts at beta and finds the NoModes modes for which the propagation constant is larger than beta.

The function returns a single structure RetVal. This structure contains:

* Ex, Cell array of complex matrices. Describes the x-polarised E-fields for each mode. The x-polarised E-fields for the first mode are in Ex{1], the x-polarised E-fields for the second mode are in Ex{2}, and so forth. Each cell contains an Nx x Nx matrix mapping the x-polarised E-fields in the coordinate system is defined by x.
* Ey. Same as above, but for y-polarised E-fields.
* Ez. Same as above, but for z-polarised E-fields.
* Hx. Same as above, but for x-polarised H-fields.
* Hy. Same as above, but for y-polarised H-fields.
* Hz. Same as above, but for z-polarised H-fields.
* Etot. Same as above, but for total E-field. Etot^2 = Ex^2 + Ey^2 + Ez^2.
* Htot. Same as above, but for total H-field. Htot^2 = Hx^2 + Hy^2 + Hz^2.
* Beta, complex vector. Array of propagation constants for each mode. Beta(1) is the propagation constant for the first mode, Beta(2) is the propagation constant for the second mode, etc.
* N, real matrix. Refractive index profile. Same as what was passed in.
* dx, real scalar. Coordinate step size. Same as what was passed in.
* lambda, real scalar. Operating wavelength. Same as what was passed in.
* k0, real scalar. 2\*pi/lambda.
* Nx, real scalar. Size of problem space.
* PML\_Depth, PML\_TargetLoss, PML\_PolyDegree, PML\_SigmaMax: related to how the perfectly matched layer was defined.

**Workflow Overview**

* Define coordinate system, refractive index profile and operating wavelength.
* Estimate propagation constant of fundamental mode and define number of modes we wish to search for.
* Call *ModeSolverFD.*
* Plot fields and/or propagation constants.

**Example Calculation**

The Examples folder contains a number of example calculations. Examples/Erlangen PCF/Erlangen\_PCF.m is a good instructional example that will be used here to show how to set up an FDFD model.

The user starts by defining a number of constants.

lambda = 633e-9; % Wavelength in nm

k0 = 2\*pi/lambda; % Wavenumber

beta = k0; % Fundamental mode prop. constant will be close to that of free space.

Nx = 1000; % Size of problem space. Assumed to be square, i.e. Nx x Nx

NoModes = 30; % Number of modes we wish to find.

As seen above, it is important to give an estimate of the propagation constant of the fundamental mode beta. Here, we are working with a hollow-core fibre and the propagation constant will be close to that of free space k0. In a liquid-filled hollow-core fibre, we can scale this by the refractive index of the liquid. We also define NoModes, which is the number of modes we wish to find. ModeSolverFD function will try to find the first NoModes modes where the propagation constant is greater than beta.

Next, the user defines the problem geometry. The problem geometry is a two-dimensional transverse refractive index map of the fibre facet. In this case, this is done by calling a local user-defined function [x, n] = GenerateFibreProfile(Nx). This function returns a one-dimensional vector x of length Nx which describes the coordinates along the x or y axis of the problem geometry in units of meters. This function also returns a matrix n of size Nx x Nx which describes the refractive index profile of the fibre facet. It is emphasised that this function is user-defined at the bottom of the Erlangen\_PCF file. The refractive index profile and coordinate system could equally been defined in the main code.

Next, the user calculates the grid step size using a simple difference calculation: dx = x(2) - x(1);

As a sanity check, the refractive index profile can be plotted:

figure;

imagesc(x\*1e6, x\*1e6, n);

axis square;

xlabel('\mum');

ylabel('\mum');

hold on

This gives the plot below:



Next, the ModeSolverFD function is called, which performs the calculation. The tic and toc statements time the execution and are not necessary:

tic

RetVal = ModeSolverFD(dx, n, lambda, beta, NoModes);

toc

Finally, we can plot the real components of the obtained propagation constant:

figure; plot(real(RetVal.beta)/k0);

We can plot the imaginary components of the obtained propagation constants, which correspond to the loss experienced by each mode. In doing so, we should convert from Np/m to dB/m:

figure; plot(imag(RetVal.beta)\*20/log(10));

Finally, we can plot the total E-field profiles for each mode:

for i = 1:NoModes

figure;

imagesc(x\*1e6, x\*1e6, RetVal.Eabs{i});

title(['\beta = ' num2str(RetVal.beta(i))]);

axis square;

end

Alternatively, we could use Ex, Ey, Ez, Hx, Hy, Hz or Htot to plot other aspects of the mode profile.

**References**

[1] Zhu & Brown, "Full-Vectorial Finite-Difference Analysis of Microstructured Optical Fibers", Optics Express Vol. 10, Issue 17 (2002).

[2] Yu & Chang, "Yee-Mesh-Based Finite Difference Eigenmode Solver with PML Absorbing Boundary Conditions for Optical Waveguides and Photonic Crystal Fibers”, Optics Express, Vol. 12, Issue 25 (2004).

[3] Sing & Fan “Choice of the Perfectly matched Layer Boundary Condition for Frequency-Domain Maxwell’s Equations Solvers” Journal of Computational Physics, Vol. 231, Issue 8 (2012).

[3] Mouthaan et al., “Efficient Excitation of High-Purity Modes in Arbitrary Waveguide Geometries”, Journal of Lightwave Technology, Vol. 40, Issue 4 (2021).

[4] Mouthaan “Holographic Control of Light Propagation in Optical Waveguides”, PhD Thesis, Cambridge University (2021).