Introduction

Transmission lines are used to efficiently transfer electromagnetic energy over a distance. The simplest form of transmission line is “twin lead” or “ladder line”, which consists of two conductors running side by side, carrying current in opposite directions. It is cost effective for lower frequencies (HF and below), where the attenuation is acceptable. At higher frequencies, fields in the unconfined directions tend to radiate energy, as illustrated in Figure 1, greatly reducing efficiency. This makes twin lead generally unsuitable for use above VHF.

Coaxial cable is a transmission line consisting of an inner conductor and an outer conductor carrying currents in opposite directions, separated by an insulating dielectric. Since the fields are completely confined as illustrated in Figure 2, coax is more efficient at higher frequencies, at an increased cost per unit length.

Coax may be used to the top of UHF, and into the millimeter microwave bands. At the higher frequencies, power handling is decreased as the skin effect increases ohmic loss, and dielectric loss becomes significant for any appreciable length. At microwave frequencies and and above, the waveguide is the only transmission line suitable for medium and high powered applications.

In the context of communications engineering, the term waveguide may refer to any linear structure designed to direct an electromagnetic wave between its ends. [1] Conceptually, waveguides work on the principle of total internal reflection, that is, EM waves bounce down their length in a zig-zag pattern, as illustrated in Figure 5, and in the case of the rectangular guide, this abstraction may be made precise. [2]
Reflection occurs at the boundary of an inner dielectric with the waveguide shell, which is either a dielectric with a lower refractive index, in the case of a dielectric waveguide, where the angle incident must be greater than a critical to cause total internal reflection, or by a conductor, which will reflect the wave unconditionally.

Dielectric waveguides are employed primarily for use at optical frequencies, though dielectric guides for sub-millimeter microwave have been produced. At these wavelengths, as machining an appropriately sized hollow channel becomes impractical and diffusion losses caused by surface roughness would be higher than scattering within the (imperfect) dielectric.

Waveguides designed for the radio frequencies are typically the conductive type, with walls made from a round or rectangular metal tube, like those pictured in Figure 4. These structures are the most efficient type of transmission line, but are rigid, bulky (a waveguide for use at 1MHz would be about 500 feet wide), and expensive, so their use is reserved for cases of high power microwave transmissions in the 1 GHz - 100 Ghz range where the attenuation in two conductor lines is prohibitively high.
Analysis

Electromagnetic waveguides are analyzed thorough the solution to the electromagnetic wave equation, a second order partial differential reduction of Maxwell's equations, with a set of boundary conditions derived from the geometry of the waveguide and the properties of its constituent materials. The wave equation has multiple solutions, or modes. Each mode is an eigenfunction of the system, with a corresponding eigenvalue which determines the cutoff frequency of that mode. [3]

Waveguide modes are characteristic as either longitudinal, a standing wave formed in confinement of the interior dielectric normal to the direction of propagation, or transverse, an oscillation in the direction of propagation. The transverse modes of an propagating electromagnetic wave are named according to the component not in the direction of propagation[4]:

- TEM mode has neither E or H components in the direction of propagation
- TE mode has no E components in the direction of propagation
- TM mode has no H components in the direction of propagation
- Hybrid modes are typically undesirable and have both E and H components in the direction of propagation.

The mode with the lowest cutoff frequency is termed the dominant mode of the waveguide. In metallic waveguides, the dominant mode is designated TE_{1,0} for rectangular, and TM_{1,1} for circular, cross sectional geometry. In contrast to the two-conductor transmission lines where TEM propagation is dominant, inside a metallic (single-conductor) waveguide Maxwell's equations require both divergence and curl of the electric field to be zero, and thus forbid the TEM mode of propagation. [3]
In this section we shall investigate the general case of a rectangular metallic waveguide operating near the microwave frequencies (where the magnetic field components are still significant). We consider the hollow (i.e. air filled, \(\sigma \simeq 0\)) metallic (\(\sigma \simeq \infty\)) waveguide over two dimensions, as depicted in Figure 5, operated in the \(\text{TE}_{1,0}\) mode. Recall in this case only the magnetic field has a component in the plane of propagation, i.e. \(E_z = 0\). The expected fields (at an arbitrary point in time\(^1\)) are qualitatively illustrated in figure Figure 6.

![Figure 6: Vector field illustrations](image)

As noted above, quantitative field values may be found in the solutions to the partial differential electromagnetic wave equation. For this mode, we must find the magnetic field in the Z direction (normal to propagation), Case 1:

\[
c^2 \frac{\partial^2}{\partial z^2} H_z = \frac{\partial^2}{\partial t^2} H_z^0
\]  

(1.1)

---

1 Note that due to our choice of coordinate system, in all cases an advance in time is indistinguishable from an advance in the Z direction. If calculated on a realistic scale, the time for one period would equal the speed of light (\(c \text{ m/s}\)), while the distance for one period would be equal to the EM wavelength (\(\text{m}\)). For \(\text{dt} = \text{dz}\), one must traverse significantly more distance in position than in time to see the same \(\text{dE}\) or \(\text{dH}\).

2 A convenient mnemonic for the plane of each field (from amateur radio) is that an “E-plane” bend is “Easy” and an “H-plane” bend is “Hard.” If you can’t figure out what direction bending a rectangular metal bar is easier, you most likely should not be playing with waveguides ;)

The magnetic field in the X direction (normal to the 'b' dimension), Case 2:

$$c^2 \frac{\partial^2}{\partial z^2} H_x = \frac{\partial^2}{\partial t^2} H_x^0$$  \hspace{1cm} (1.2)

And the electric field in the Y direction (normal to the 'a' dimension), Case 3:

$$c^2 \frac{\partial^2}{\partial z^2} E_y = \frac{\partial^2}{\partial t^2} E_y^0$$  \hspace{1cm} (1.3)

We may seek an exact solution to these equations algebraically, or use a numerical approach known as the method of finite differences to produce an approximate solution. Both tactics are described below.

**Finite Difference Method**

The finite difference method (FDM) is a simple numerical technique used in solving partial differential equations with a solution region surrounded by a set of known initial or boundary conditions ([4]sec 14.3). In the case of a waveguide, we can safely make assumptions about the boundaries and initial conditions – namely, (1) that the walls are perfect conductors and thus must be at equal (relatively zero) electric potentials, (2) that the dominant mode in a rectangular waveguide is $TE_{1,0}$ and thus the vector field component $E_z$ is zero, and (3) that the for propagation, the initial potentials should be non-zero.

Once the boundary conditions are known, we establish a grid (implemented as a matrix) to represent discrete points of a two dimensional scaler field (position and position, or time and time; see') as shown in Appendix B - Indexing the Finite Difference Algorithm. The distance between two neighboring points on the grid is a finite difference (dx, dy, dz, etc...) such that if the number of points were increased to infinity, the change in scalar value between the same two points would approach the derivative of the field located between them.

Initial conditions for each case are discussed in the relevant sections below, however all cases use the same algorithm for calculating the interior values. The algorithm is:

$$u(x,t) = \alpha[u(x-1,t-1) + u(x+1,t-1)] + 2(1-\alpha)u(x,t-1) - u(x,t-2)$$

Where $\alpha = \left[\frac{c \Delta t}{\Delta x}\right]^2$, $\Delta t$ and $\Delta x$ are the finite difference steps, and $c$ is the constant from the wave equation (set to 1). This algorithm must compute all values of $x$ (an entire “row”) before moving to the next value of $t$ (the next time step).
Stability

Even with a consistent algorithm, the choice of \( c \), \( \Delta t \), and \( \Delta x \) determine if a particular FDM scheme will work. The criteria for stability is that compound error in the current computed solution will not be amplified in subsequent computations. Intuitively, this could be stated:

\[
\sum_{x=0}^{N-1} |u(n,t+1)| < \sum_{x=0}^{N-1} |u(n,t)|
\]

Using von Neumann analysis and setting \( u(n,t+1) = a \cdot u(n,t) \), the limit for stability may be formally shown to be \( a \leq 1 \) [5], thus \( c \) must be chosen according to the ratio of \( \Delta t / \Delta x \) to keep \( \alpha \leq 1 \). In the case that \( \Delta t = \Delta x \), we can set \( c = 1 \) without compromising calculation stability.

Implementation

There are several ways this algorithm can be implemented in MATLAB. The first, and most obvious, method is a loop over each element as shown in Snippet 1 and as outlined in [4], which requires two loops and that the inner \( (x) \) boundary conditions be populated beforehand:

```
for t = 2 : TMAX  % We rely on 2 past values of T
    for x = 1 : XMAX -1  % We rely on +-1 values of X
        grid(x +XMIN, t +TMIN) = ...  % One row back, left and right
            a*(grid(x-1 +XMIN, t-1 +TMIN) + grid(x+1 +XMIN, t-1 +TMIN)) ...
        ...  % One row back, center
            + 2*(1-a)*grid(x +XMIN, t-1 +TMIN) ...
        ...  % Two rows back, center
            - grid(x +XMIN, t-2 + TMIN);
    end
end
```

**Snippet 1: Traditional FDM loop**

This simple approach is oblivious to the vector/matrix capabilities present in MATLAB. Instead one may, as highlighted in Snippet 2, compute an entire inner \( (x) \) row at once, as well as populate the necessary boundary conditions for the next row. The impetus for implementing the calculation row-wise, rather than element-wise, stems from the nature of the MATLAB environment. MATLAB code is an interpreted language, that is, statements are broken down into machine-level chunks line-by-line during execution (in contrast to a compiled language like C, where the entire program is assembled into machine language at the time of compilation). Whereas a C compiler may unroll (create a series of sequential statements replacing the iteration variable with a constant in each) a loop, the MATLAB interpreter can do nothing in the way of “for loop” optimizations, and it must interpret the inner statement \( N*N \) times. When expressed as vector indexing operations, as shown in Snippet 2, the interpreter can optimize the
operation: it compiles the vector operation down series of low-level single dimensional index operations and executes them sequentially (or often in parallel), having to interpret the statement only once.

```matlab
% boundary scalar
bcs = 2*cos(pi*dy);

% boundary scalar
r2b = 1*bcs;
grid= zeros(n*cyc,n); % preallocate

grid(1,:) = r1; grid(2,:) = r2; % set row 1 and 2 (initial conds)

% loop for remainder of rows
for y = 3:n*cyc
    % calculate next row (0 back)
    r0b = a*([r1b(2:n), 0] + [0, r1b(1:n-1)]) + (1-a)*[0,r1b(2:n-1),0] - r2b;
    % propagate left and right boundary condition
    r0b(1) = r1b(1)*bcs - r2b(1);
    r0b(n) = r1b(n)*bcs - r2b(n);
    % advance one row
    grid(y,:) = r0b;
    r2b = r1b;
    r1b = r0b;
end
```

**Snippet 2: Initial MATLAB optimization**

The code in Snippet 2 may be further improved by implementing the boundary condition operators as part of the next row equation using indexing operators and vector concatenations. Since the second row back (r2b) is subtracted from both the current row and boundary conditions, this may be done all at once in the sum. The astute reader will also note that the use of local variables r0b, r1b, and r2b is unnecessary. These changes are shown in Snippet 3 below.

```matlab
for y = 3:n*cyc
    % calculate next row, including boundary conditions
    grid(y,:) = a * ([
        [0, grid(y-1,3:n), 0] + [0, grid(y-1,1:n-2), 0] ... shift left
    + [0, grid(y-1,1:n-2), 0] ... shift right
    ) ... %
    + [
        bcs * grid(y-1,1), ... left boundary
        (1-a) * grid(y-1,2:n-1), ... one row back*(1-a)
        bcs * grid(y-1,n) ... right boundary
    ] ... %
    - grid(y-2,:); % all minus 2 back
end
```

**Snippet 3: Final optimization - FDM inner loop in one line**

The author concedes that the single-line optimized version is more difficult to read, even spread over ten lines (the ellipsis in MATLAB indicates that a command continues on the next line of text). The source code used to generate the plots in the following section, as listed in Appendix A – Project Code Listing under subheading A.1 Assignment Code, has, for clarity, been left unoptimized.
Exact Solution

In the majority of “real-world” EM problems, boundary geometry is too complex to approach a solution analytically. However, due to the inherent simplicity of the cases presented here, it is possible to arrive at the closed form of an exact solution algebraically. As the aim of this report is to study the method of finite differences, the intermediate algebra will be left as an exercise to the curious reader.

Being a second order partial differential equation, we predict that the exact solution to all three cases take the form of a sinusoid on one independent axis modulated by a second sinusoid on the other independent axis. Under normalized dimensions (see note 1 regarding the choice of the spacial independent variables here) the solutions take the form:

\[ U_d(x, t) = \cos(\pi x + \phi) \cos(\omega t - \gamma z + \theta) \]  

(1.4)

Where \( U_d \) is the field in question, \( \phi \) is a positional phase shift, \( \theta \) is a temporal phase shift, \( \omega \) is the frequency (which is normalized to \( \pi \)) and \( \gamma \) is the group velocity accounting for the speed of light in the medium (also normalized to \( \pi \)). Specifics for each case are described in detail below.

Results

Case 1

In case Case 1 we arrive at the solution to the \( H_z \), the component of the magnetic field in the Z direction. This solution provides one dimension of values, those pointed in the Z direction. In the \( TE_{1,0} \) mode these magnitudes are constant in the Y direction at any arbitrary point \( H_z(x, t) \), thus the scalar contribution adds to all vectors originating from the line \( X_0 \) in the direction shown in Figure 7, below.

Figure 7: The vectors described by \( H_z(x, t) \) at an arbitrary point \( Z=0 \)
The exact solution to this case at $Z=0$ (as illustrated) takes the form of (1.4) with $\phi=0$ and $\omega=\pi$, that is:

$$H_z(x,t) = \cos(\pi x) \cos(\pi t) \quad (1.5)$$

Both results from the FDM and exact (1.5) solutions over two periods ($4\pi$) are depicted in Figure 8 below. In the three-up subplot, from left to right, are the results from the FDM, the exact solution, and the error between the numerical and exact solutions.

As shown in Figure 7, the scalar value on the $z$ ('up') axis of the plots in Figure 8 is actually the magnitude of the vectors pointed in the $Z$ direction ('out') axis as labeled in Figure 7. Taken alone, this is not enough information to draw the vector magnetic field which, in the $TE_{10}$ mode, also has components in the X direction$^3$ (studied in Case 2).

Note that the units on the error scale are $10^{-4}$, whereas the other scales are over unity (0-1), and with $N=32$, over two periods, the maximum error in the approximation was about 0.02% (0.0002). The average error was much lower. Over additional periods the error does indeed increase, though not in a linear or predictable manner as it does in the remaining two cases.

**Case 2**

For case Case 2 we derive the solution for $H_X$, the magnetic field component in the X direction. This solution also provides only one dimension of values, those pointed in the X direction. Like Case 1, there is no dependence on

$^3$ Taken blindly this is true, however with knowledge that $\nabla \cdot B = 0$ (no magnetic monopoles) i.e. $B = \nabla \times A$, we may take the curl of this result to produce the correct field lines without a solution to the other vector component. Code to do this is in the Appendix A.6 – Calculating H field vector potential using only one wave equation. Note that this would not be the case for a TM wave, as the E field needn't oblige such rules.
the Y direction so at an arbitrary point \( Z \), the scalar value of \( H_x(x, t) \) represents a magnitude contribution to the all \( H \) field vectors originating at the line \( X_0 \), as shown in Figure 9 below.

![Figure 9: The vectors described by \( H_x(x, t) \) at an arbitrary point \( Z=0 \)](image)

The exact solution to this case at \( Z=0 \) (as illustrated) takes the form of (1.4) with \( \phi = \pi/2 \) and \( \omega = \pi \), that is:

\[
H_x(x, t) = \sin(\pi x) \cos(\pi t) \quad (1.6)
\]

Both results from the FDM and exact (1.6) solutions over two periods (4\( \pi \)) are depicted in Figure 10 below. In the three-up subplot, from left to right, are the results from the FDM, the exact solution, and the error between the numerical and exact solutions. Note that the units on the error scale are \( 10^{-3} \), whereas the other scales are over unity (0-1), and with \( N=32 \), over two periods, the maximum error in the approximation was about 0.15% (0.0015). In this case, the average error was not much lower. Over additional periods the error increases as a sinusoidal exponentially. This behavior is expected because the error is compounded with each time step. It remains unknown how Case 1 achieved linear error (rather than exponential).

![Figure 10: Case 2 results: (L-R) Method of finite differences, Exact solution, Error (<0.015%)](image)

Together with the results obtained by the simulation which produced Figure 8, we now have enough information to plot the exact vector magnetic field of the X-Y plane in our waveguide. This field is illustrated in Figure 11, below.
Here, we are looking down at the wide part ('a' dimension) of our waveguide, normal to the H plane. The vertical axis corresponds to the X direction as labeled in Figure 6. For the purposes of visualization, it is easiest to consider spatially⁴, that is, a snapshot of the fields at a fixed time. In this case, the horizontal axis runs down a length of the waveguide in the Z direction. The rings under the vector field arrows indicate equipotential magnitudes of the X and Z components, and are the “X-Y” view of Figure 8 and Figure 10. The series of peaks and valleys that run in a line across the center of the illustration are those of the X component, the series running down the top and bottom are the Z component, as may be surmised by the length and direction of the vector arrows at the summit (smallest circle) of each.

**Case 3**

In case Case 3 we derive the solution for $E_Y$, the electric field component in the Y direction. This solution provides only one dimension of values, those pointed in the Y direction. Unlike the H field, which has components in both X and Z directions and requires two scaler contributions (Case 1 and Case 2) to represent the vector magnetic field, the E field only has a component in the Y direction, thus a single scalar field may completely specify the vector electric field. The function has no dependance on Y, so at an arbitrary point $Z$ the scalar value of $E_Y(x,t)$ represents the magnitude of all vectors along the line $X_0$ in the direction depicted in Figure 12 below. The direction of all such vectors is $a_Y$ (normal to the X-Z plane).

---

⁴ It should be noted that for any given position $Z$, it would be equally valid to plot time on the horizontal axis.
The exact solution to this case at Z=0 (as illustrated in Figure 13) takes the form of (1.4) with $\phi = \pi/2$ and $\omega = \pi$, that is:

$$E_y(x, t) = \sin(\pi x) \cos(\pi t) \quad (1.7)$$

One should note that the scalar fields $H_X (1.6)$ and $E_y (1.7)$ are numerically indistinguishable. The difference is that they contribute to their respective vector fields in perpendicular directions ($a_x$ and $a_y$, respectively). In the general case for the $TE_{1,0}$ mode, the electric field equation is reminiscent of Ohm's law: $E_y = Z_H H_X$, where $Z_H$ is the impedance.

**Figure 12**: The vector described by $E_y(x, y)$ at an arbitrary point $Z=0$

**Figure 13**: Case 3 results: (L-R) Method of finite differences, Exact solution, Error (<0.015%)
We have demonstrated calculation of the three scalar component fields present in a $\text{TE}_{1,0}$ wave, and presented each as a three dimensional scalar plot. Enough information has been collected to plot both the vector electric and vector magnetic fields of our waveguide. We were able to create a satisfactory plot in two dimensions for the vector magnetic field, but visualizing both vector magnetic and vector electric is non-trivial. The reason for this is that we must somehow represent two vector fields, which is a total of 6 component values (although many are zero), at each point on an NxNxN cubic grid. Any such attempt would surely be futile. However, we may make compromises in precision in the pursuit of clarity; one such attempt is depicted in Figure 14, above.

In this illustration, only the magnetic field vectors in the plane $Y=1/2$ (as in Figure 11) are drawn, although this is representative of any slice within the waveguide. For the magnetic vector field to be visible, only half of the vector electric field (either the half above $Z=0$, or the half below $Z=0$) is drawn at any given time, with positive $[a_Y$ component] values on the top, and negative values on the bottom, to better illustrate the propagating wave. Also, only one electric field vector is drawn for any point on the line $(x_0,t_0,z)$, since others would not convey new information. Time and $Z$ both run left to right.

Perhaps the most useful visualization to aid in gaining an intuitive understanding of the $\text{TE}_{1,0}$ mode, however, is one animated against time. This is the dimension that we (humans) are most accustomed to seeing things propagate, and the extension from three dimensions to four is natural. The author has prepared three examples from the data presented here, which are available online\(^5\), and in doing so has gained further intuition than would have been possible from any static illustration or equation. The process of animating MATLAB output is discussed later in this report.

\(^5\) http://mason.gmu.edu/~jdilles/classes/ece305/index.html
Generalization of FDM

Unequal Time Steps

As noted under the Stability section of Finite Difference Method Analysis, the constant c must be chosen such that $\alpha \leq 1$. For the normalized case where $\Delta t = \Delta x$, we can set $c = 1$ without much thought. In the general case where $\Delta t \neq \Delta x$, we can quickly run into trouble. In terms of $\Delta t$, and $\Delta x$, we can find an upper bound for c:

$$\alpha = \left[ \frac{c \Delta t}{\Delta x} \right]^2 \rightarrow c = \frac{1^2 \Delta x}{\Delta t}$$

As it turns out, for a stable calculation we want $\alpha \approx 1^-$, that is, very close, but slightly smaller than 1. In this case, the value for c in the general case may be computed in MATLAB using the code in Snippet 4, which ensures that the value of $\alpha$ is always just less than 1.

```matlab
c = (dx/dy) - eps;
alpha = ((c*dy)/dx).^2;
```

Snippet 4: Generalized calculation of constant from time-steps

Corrections for an Arbitrary Dimension

The corrections for an arbitrary 'a' or 'b' dimension are more difficult to approach, due to the implications associated with the dimensions. Until now, we have neglected the fact that the subject of our model has a wavelength $\lambda$ meters, with frequency $f = c/\lambda$ Hertz, where c is the speed of light (299792458 meters per second). Of principle concern is the ratio between $\lambda$ and the 'a' dimension, which determines the the operating range of frequencies which may be used with the guide. As a rule, wavelengths longer than twice the 'a' dimension will not propagate, instead decaying effervescently. Shorter wavelengths may propagate, though a wavelength shorter the 'a' dimension will allow for the higher modes, making analysis prohibitively difficult. As such, waveguides are most often operated within the range of frequencies having wavelengths 2.0 to 1.0 times the 'a' dimension. [3]

For any case other than $\lambda = 2.0 \, a$ it is not possible to model the propagation based on one-dimensional “wave on a ideal tensioned string” equations presented above. For these cases, the wave bounces along the guide as shown in Figure 3. Accurate numerical modeling of this scenario has proven elusive, and developments in this area are sparse. The author hopes to approach this problem in later work.

The 'b' dimension, by definition always smaller than 'a', is of secondary concern for the purposes of
generalization, since it's prime implication is in the power-handling capability of the waveguide. This is a function of the electric field potential difference between the top and bottom of the guide, which may overcome the ionization potential of the dielectric (air) inside. Because the walls of the waveguide are not perfect conductors, with sufficient potential, arcing can occur, transmitting a broad spectrum of noise and potentially damaging connected equipment [6].

**Digital Waveguide (Z-transform) Approach**

It is worth briefly mentioning that the MFD may be modeled in the “Z” domain, allowing the use of sophisticated techniques developed for the field of digital signal processing. While the two dimensional Z transform is outside the scope of this report, the curious reader is encouraged to investigate further into [7].

**Animating MATLAB output**

One of our greatest limitations as humans is two dimensional vision. While we can interpret the world in three spacial dimensions through stereo perception, visualizing four or more spatial dimensions are beyond our intuition, presenting a challenge when attempting to illustrate a function which depends on three or more variables. In general, we do have a good grasp of the progression of time, which can be thought of as a fourth dimension (although it may be used only for independent positively incremented variables to preserve causality), especially in context of velocity.

As such, the temporal dimension can be a powerful tool when adding another independent variable to graphical data would add clutter, or simply be impossible. In recognition of this, MATLAB offers a simple interface to create movies and animations, directly from existing plots. As illustrated in Snippet 5, it is possible to create an animated GIF, a highly portable, or system independent, file that is handled well by almost any web browser. The animated GIF can also be created as to loop continuously, which for periodic functions, can give the impression of infinite and continuous motion using only a finite number of frames. The key to this illusion is to create frames over one period, with the first and last frame being one time step apart.

---

6 Example code from http://www.mathworks.com/matlabcentral/fileexchange/21944-animated-gif
Snippet 5: The basic animation loop

For longer simulations (especially at higher resolutions) the code in Snippet 5 will cause the JVM in MATLAB to run out of memory. When this happens the quickest recourse is to write the frames individually to disk and animate them using a separate utility, as was done in the code listed in Appendix A.2. More advanced animation methods are possible, including having MATLAB export an AVI directly. Readers are encouraged to experiment further with this rich functionality.

Conclusion

In this report, we have examined basic principles of the electromagnetic wave guide, comparing it to other transmission lines, noting appropriate applications, and reviewing its fundamental properties. We have introduced numerical and analytic methods of analysis for computing the vector magnetic and vector electric fields within a rectangular metallic waveguide operating in the TE_{1,0} mode, and compared the two solutions. We have proposed methods for visualizing these solutions in a manner such that a reader may gain intuition into the processes involved, beyond the algebraic notation. We have noted the implications of generalizing the result presented herein, in both variable time steps and arbitrary dimensions. Finally we introduced a method for representing a fourth dimension in plotted data by animating MATLAB output. In conclusion, this exercise is considered a success, as results derived here match those generally accepted in the field. Overall, the project was enjoyable and the author hopes to peruse the subject further in the future.
Bibliography


Photograph Credits

1. Figure 4 (Page 4) - Microwave Engineering Services Corp. (http://www.microwaveengservices.com/)
Appendix A – Project Code Listing

A.1 Assignment Code

%% ECE 305 Project
%%% Jacob Dilles (G00513892)
%%% 18 APR 2011

clear % variables. Good form

pcase = 3; % Set case to 1, 2, or 3, as outlined in the project
plotn = 4; % 1=MoFD, 2=exact, 3=error, 4 = all on subplot

% This is for arbitrary case 4
phi = 0;

% Constant (from 2nt order diffeq)
c = 1; % 0.9 is pretty good, 1 is perfect

% Number of half-periods of time to plot
cycles = 4;

% Resolution: steps per lambda/2. (32 is more than enough)
N = 16;

% Makes computations more readable by allowing 0-based indecies
% NOTE: All calls to hgrid should include a hgrid(a +XMIN, b +TMIN)
XMIN = 1;
XMAX = N-1;
TMIN = 1;
TMAX = cycles * N -1;

% Make all computations over unity (0 <= x <= 1)
dt = cycles/TMAX;
dx = 1/XMAX;

% Gain/loss alpha. Should be less than 1 in a dielectric
a = ((c*dt)/dx)^2;

% Generate UCS
[JJ, II] = meshgrid(0:TMAX, 0:XMAX);
XX = II*dx;
TT = JJ*dt;

hgrid = XX.*0; % zeros(size(xx))

%% Initial conditions %
% The initial conditions for each of the three cases are defined here

% Case 1 computes H_z
if pcase == 1
% Conditions for exact solution
ex_timeshift = 0; % cos
ex_phaseshift = 0; % cos

% Compute x boundary conditions (across the waveguide, two rows)
hgrid(:, 0 +TMIN) = cos(pi * XX(:, 0 +TMIN)); % initial value
hgrid(:, 1 +TMIN) = cos(pi * dt) * hgrid(:, 0  +TMIN) ; % one after

% Compute t boundary conditions (down sides of waveguide, one row each side)
for t = 0 : TMAX - 2;
% Left size
hgrid(0   +XMIN, t+2 +TMIN) = ...
2*cos(pi*dt)*hgrid(0   +XMIN, t+1 +TMIN) - hgrid(0   +XMIN, t +TMIN);
% Right side
hgrid(XMAX +XMIN, t+2 +TMIN) = ...
2*cos(pi*dt)*hgrid(XMAX +XMIN, t+1 +TMIN) - hgrid(XMAX +XMIN, t +TMIN);
end
end

% Case 2 computes H_x
if pcase == 2
% Conditions for exact solution
ex_timeshift = pi;  % -cos
ex_phaseshift = pi/2;  % sin

% Compute x boundary conditions (across the waveguide, two rows)
hgrid(:, 0 +TMIN) = sin(pi * XX(:, 0 +TMIN));  % initial value
hgrid(:, 1 +TMIN) = cos(pi * dt) * hgrid(:, 0 +TMIN);  % one after

% Compute t boundary conditions (down sides of waveguide, one row each side)
for t = 0 : TMAX - 2;
  % Left size
  hgrid(0 +XMIN, t+2 +TMIN) = ...  2*cos(pi*dt)*hgrid(0 +XMIN, t+1 +TMIN) - hgrid(0 +XMIN, t +TMIN);
  % Right side
  hgrid(XMAX +XMIN, t+2 +TMIN) = ...  2*cos(pi*dt)*hgrid(XMAX +XMIN, t+1 +TMIN) - hgrid(XMAX +XMIN, t +TMIN);
end

% Case 3 computes E_y
if pcase == 3
% Conditions for exact solution
ex_timeshift = pi;  % cos
ex_phaseshift = pi/2;  % sin

% Compute x boundary conditions (across the waveguide, two rows)
hgrid(:, 0 +TMIN) = sin(pi * XX(:, 0 +TMIN));  % initial value
hgrid(:, 1 +TMIN) = cos(pi * dt) * hgrid(:, 0 +TMIN);  % one after

% Compute t boundary conditions (down sides of waveguide, one row each side)
for t = 0 : TMAX - 2;
  % Left size
  hgrid(0 +XMIN, t+2 +TMIN) = ...  2*cos(pi*dt)*hgrid(0 +XMIN, t+1 +TMIN) - hgrid(0 +XMIN, t +TMIN);
  % Right side
  hgrid(XMAX +XMIN, t+2 +TMIN) = ...  2*cos(pi*dt)*hgrid(XMAX +XMIN, t+1 +TMIN) - hgrid(XMAX +XMIN, t +TMIN);
end

% Case 4 computes with an arbitrary phase
if pcase == 4
% Conditions for exact solution
ex_timeshift = 0;  % cos(0) = 1
ex_phaseshift = phi;

% Compute x boundary conditions (across the waveguide, two rows)
hgrid(:, 0 +TMIN) = cos(pi * XX(:, 0 +TMIN) + phi);  % initial value
hgrid(:, 1 +TMIN) = cos(pi * dt) * hgrid(:, 0 +TMIN);  % one after

% Compute t boundary conditions (down sides of waveguide, one row each side)
for t = 0 : TMAX - 2;
  % Left size
  hgrid(0 +XMIN, t+2 +TMIN) = ...  2*cos(pi*dt)*hgrid(0 +XMIN, t+1 +TMIN) - hgrid(0 +XMIN, t +TMIN);
  % Right side
  hgrid(XMAX +XMIN, t+2 +TMIN) = ...  2*cos(pi*dt)*hgrid(XMAX +XMIN, t+1 +TMIN) - hgrid(XMAX +XMIN, t +TMIN);
end

return
end

%% Finite Difference Implementation
% Note this implementation is the same for all three initial conditions
for t = 2 : TMAX
% We rely on 2 past values of T
for x = 1 : XMAX -1
% We rely on +1 values of X
hgrid(x +XMIN, t +TMIN) = ...
  ... % One row back, left and right
  a*( hgrid(x-1 +XMIN, t-1 +TMIN) + hgrid(x+1 +XMIN, t-1 +TMIN) )
  ... % One row back, center
  + 2*(1-a)*hgrid(x +XMIN, t-1 +TMIN)
  ... % Two rows back, center
  - hgrid(x +XMIN, t-2 + TMIN);
end

%% Compute the exact solution
% Note that this handles all four cases
exgrid = cos(pi*XX + ex_phaseshift) .* cos(pi*TT + ex_timeshift);

%% Compute the error
% (experimental-theoretical)/theoretical
elemerror = (hgrid - exgrid);

%% Plot results
% determine appropreate axis labels and titles
switch pcase
  case 1
    stitl = 'H field vs z';
    szlab = 'H_z';
  case 2
    stitl = 'H field vs x';
    szlab = '\sigma H_x';
  case 3
    stitl = 'E field vs y';
    szlab = 'E_y';
  case 4
    szlab = ['\phi = ', num2str(phi)];
  otherwise
    szlab = 'Invalid case number';
end

figure(1)

% Plot and label MoFD
if plotn == 4
  subplot(1,3,1)
end
if plotn == 1 || plotn ==4
  surf(XX,TT,hgrid);
  title(['MoFD numerical solution - ', stitl]);
  xlabel('Position');
  ylabel('Time');
  zlabel(szlab);
end

% Plot and label exact
if plotn == 4
  subplot(1,3,2)
end
if plotn == 2 || plotn ==4
  surf(XX,TT,exgrid);
  title(['Exact Solution - ', stitl]);
  xlabel('Position');
  ylabel('Time');
  zlabel(szlab);
end

% Plot case (error)
if plotn==4
  subplot(1,3,3)
end
if plotn==3 || plotn==4
  surf(XX,TT,elemerror)
  title(['Error - ', stitl]);
  xlabel('Position');
  ylabel('Time');
  zlabel(['\sigma ',szlab]);
end
clear

% Number of half-periods of time to plot
cycles = 4;

% Resolution: steps per lambda/2. (32 is more than enough)
N = 16;

% Makes computations more readable by allowing 0-based indicies
% NOTE: All calls to hgrid shoud include a hgrid(a +XMIN, b +TMIN)
XMIN = 1;
TMIN = 1;
XMAX = N-1;
TMAX = cycles * N -1;

% Make all computations over unity (0 <= x <= 1)
dt = cycles/TMAX;
dx = 1/XMAX;

% Generate UCS
[JJ, II] = meshgrid(0:TMAX, 0:XMAX);
XX = II*dx;
YY = JJ*dt;
ex_timesthift = pi; %cos
ex_phaseshift = pi/2; %sin

figure(1)

% maximum index value (scaler int)
tmax = (2*pi)/dt;

% ensure we get a full 2pi via ceil
for t = 0:ceil(tmax)
    % exact solution to 1d wave eqn over grids XX and YY at the given time t
    [x, y, z] = ewave(XX, YY, pi/2, t*dt);
    % surface plot (like MESH, but fills in the spaces between grids)
    surf(x,y,z);

    % set a nice vantage point (az -70, ele 24) as read off bottom left corner
    % of plot window whilst holding the 'pan 3d' cursor down.
    view(-70,24);
    % get the frame object from the curent axis (created by SURF)
    f = getframe();
    % converts the MATLAB frame object into a bitmapped image
    [im,map] = frame2im(f);
    % write the image to a file named "ewave_0001.png, 0002.png, etc...
    imwrite(im, sprintf('~/Users/theshadow/temp/mla/ewave_%04d.png',t),'png');
    % this will later be turned into an animated GIF using sw from the GNU dist:
    % first "mogrify -format gif *.png" (using ImageMagick)
    % then "ls -l *.png > list.txt" (list new .gif paths in a text file)
    % then "whirlgif -o animation.gif -time 5 -i list.txt" (animate gif)
    % Or QuickTime 7 Pro on the Mac can turn a sequence of PNGs into a .mov
end
A.3 – Impedance Matching Code

clear
% frequency over sm
fr = 1;
% phase
ph = 0;
% width (left, right) of obstruction
wr = [0.1 0.1];
w1 = [0 0]; % start stop for obstruction
%Dp = 0.8;
cyc = 8;
N = 64;
dp = 1;
% always live zero to 1
xv = linspace(0, dp, N);
% here we can go further
yv = linspace(0, cyc, N*cyc);
% dy = dx, but it is easier to see when explicitly stated
dx = xv(2) - xv(1);
dy = yv(2) - yv(1);
% preallocate
gr = zeros(N*cyc, N);
% set row 1 and 2 (initial conds
gr(1,:) = r1;
gr(2,:) = r2;
% boundary conditions
bcm = ones(size(r1));
bcm(1) = 0;
bcm(N) = 0;
ls = floor(wr(1)*N);
rs = floor(wr(2)*N);
ms = N-1-ls-rs;
machm = [zeros(1,ls), ones(1, ms), zeros(1,rs)];
machi = eps;
% loop for remainder of rows
for y = 3:(N*cyc)
    if y > N*w1(1) && y < N*w1(2)
        [r2, r1] = mofd(a,r2, r1, machm, machi);
    else
        [r2, r1] = mofd(a,r2, r1, bcm, 0);
    end
    gr(y,:) = r2;
end
figure(1);
[x,y] = meshgrid(xv,yv);
mesh(x,y,gr)
energy = sum(gr.^2,2)/(N);
figure(2)
plot(yv, energy);

A.4 - MOFD.m

function [ r0b, rlb ] = mofd(a, rlb, r2b, bcm, bcs)
% MOFD computes one row of MOFD for wave eqn, including boundaries
% % arguments:
% % a      alpha value 0 <= a <= 1 to apply to a*m(x, t-1)
% % rlb    row (:,t-1)
% % r2b    row (:,t-2)
% % bcm    boundary condition mask (zeros at boundaries, ones elsewhere)
% % bcs    boundary condition scaler 2*(t-1)*bcs - (t-2)
% % returns
% % r0b   new computed row
% % rlb   one row prior (for [rlb, r2b] = mofd(... rlb, r2b, ...) in loop)

n = length(rlb);

% calculate mofd
r0b = (a * ([rlb(2:n), 0] + [0, rlb(1:n-1)]) + 2*(1-a)*r1b - r2b) .* bcm + ... 2*(bcm==0).*bcs;
function [ ex, ey, ez ] = ewave( XX, YY, phi, t)
%EWAVE calculates the exact solution of the 2D EM wave eqn in the Z-direction

ex = XX;
ey = YY;
ez = cos(pi*XX-phi) .* cos(pi*YY-t);

A.5 – ewave.m

A.6 – Calculating H field vector potential using only one wave equation

The vector potential magnetic field may be calculated from only one of the two H wave equations (for example, $H_z$) because we know that there are no magnetic monopoles. In MATLAB, this is done like so:

\[
\begin{align*}
\{J_J, J_I\} &= \text{meshgrid}(0:TMAX, 0:XMAX); \\
XX &= J_I \times dx; \\
YY &= J_J \times dt; \\
\{x, y, z\} &= \text{ewave}(XX, YY, 0, t \times dt); \\
% must expand into 3x 3d matrices for MATLAB curl
zz &= \text{repmat}(z, [1 1 N]); \\
xx &= \text{repmat}(x, [1 1 N]); \\
yy &= \text{repmat}(y, [1 1 N]); \\
[\text{curlx, curly, curlz, cav}] &= \text{curl}(xx, yy, zz);
\end{align*}
\]

contour(x, y, z); hold on 
quiver(x, y, curlx(:,:,N), currly(:,:,N)); hold off

This code will calculate and plot the vector field lines, overlaid on the calculated value.
Appendix B - Indexing the Finite Difference Algorithm

Indexing The Finite Difference Algorithm

Initial conditions given for \( A(x,0), A(x,1), A(0,t), A(X_{\text{MAX}}, t) \)
Algorithm evaluated at \( A(x,t) \) for \( \{1 \leq x \leq X_{\text{MAX}} - 1, \ 2 \leq t \leq T_{\text{MAX}} \} \)

Given matrix \( A_0 \) of size \( M \) rows by \( N \) cols
the value of \( A_0(x,t) \) over \( 0 \leq x \leq X_{\text{MAX}}, 0 \leq t \leq T_{\text{MAX}} \)

In MATLAB \( \text{zeros}([M \ N]) \)
\( A_0(x,t) = A(t + T_{\text{MIN}}, x + X_{\text{MIN}}) \)
\( X_{\text{MIN}} = T_{\text{MIN}} = 1 \)
\( X_{\text{MAX}} = N - 1 \quad ; \quad T_{\text{MAX}} = M - 1 \)

In Sensible Languages \( \text{double}[M][N] \)
\( A_0(x,t) = A(t + T_{\text{MIN}}, x + X_{\text{MIN}}) \)
\( X_{\text{MIN}} = T_{\text{MIN}} = 0 \)
\( X_{\text{MAX}} = N - 2 \quad ; \quad T_{\text{MAX}} = M - 2 \)

Legend
- Boundary Condition
- Unknown Value
- First Computed Value
- Computed Value
- Position Being Computed
- Values Used for Current Computation