University of the Witwatersrand
Johannesburg, South Africa

Masters Dissertation

Lightning Return Stroke Electromagnetics - Time Domain Evaluation and Application

Author:
Carson William Ian McAfee

Supervisor:
Prof. Kenneth John Nixon

October 17, 2016

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering in the

Lightning and EMC Research Group
School of Electrical and Information Engineering
Faculty of Engineering and the Built Environment

Declaration of Authorship

I declare that this dissertation is my own unaided work. It is being submitted to the Degree of Master of Science in Engineering to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other University.

.....
(Signature of Candidate)

..... day of ..... year .....

Abstract

The work presented extends and contributes to the research of modelling lightning return stroke (RS) electromagnetic (EM) fields in the time domain. Although previous work in this area has focused on individual lightning electromagnetic pulse (LEMP) modelling techniques, there has not been an investigation into the strengths and weaknesses of different methods, as well as the implementation considerations of the models. This work critically compares three unique techniques (Finite Antenna, FDTD, and Single Cell FDTD) under the same ideal simulation parameters. The research presented will evaluate the EM fields in the range of 50 m to 500 m from the lightning channel. This range, often referred to as the near field distance, has a significant effect on lightning induced overvoltages on distribution lines, which are primarily created by the horizontal EM fields of the RS channel. These close distances have a significant effect on the model implementations, especially with the FDTD method. Each of these modelling methods is explained and tested through examples. The models are implemented in C++ and have been included in the Appendix to aid in future implementation. From the model simulations it is clear that the FDTD method is the most comprehensive model available. It allows for non-ideal ground planes, as well as complex simulation environments. However, FDTD has a number of numerical related errors that the Finite Antenna method does not suffer from. The Single Cell FDTD method is simple to implement and does not suffer from the same numerical errors as a full FDTD implementation, but is limited to simple simulation environments. This work contributes to the research field by comparing and evaluating three techniques and giving consideration to the implementation and the applicability to lightning EM simulations.

Acknowledgements

I would like to acknowledge Professor Ken Nixon and thank him for all the enthusiasm and commitment. I would also like to thank him for always having more faith in me than I did, and for constantly improving the engineering potential of those around him. I would like to thank Ms Yu-Chieh (Jessie) Liu, and Mr Hugh Hunt for all the help, advice and stimulating conversations over the years. I would like to thank Mr Kerren Ortlepp for the programming help, without which none of the simulations would have been possible.

Funding and support from the following organisations is also gratefully acknowledged:

- The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.
- The NRF for direct funding of the Research Group.
- Eskom for the support of the Lightning/EMC Research Group through the Tertiary Education Support Programme (TESP) programme.
- The Department of Trade and Industry (DTI) for Technology and Human Resources for Industry Programme (THRIP) funding.
- The CrunchYard (www.crunchyard.com) for providing computational resources.

Finally I would like to thank my parents for all their love and support. Thank you for making it possible for me to live an interesting life, filled by the pursuit of knowledge and adventure.

Contents

List of Figures

List of Tables

List of Symbols

$\vec{E}$		[V/m^1]	Electric Field Intensity Vector
$\vec{D}$		[C/m^2]	Electric Flux Density Vector
$\vec{B}$		[Vs/m^2] OR [T]	Magnetic Field Intensity Vector
$\vec{H}$		[A/m^1]	Magnetic Flux Density Vector
J		[A/m^2]	Current Density
σ		[S/m^1]	Conductivity of medium
ρ_q		[C/m^3]	Volume Charge Density
ϵ		[F/m^1]	Permittivity of a medium
ϵ_r			Relative Permittivity
ϵ_0	$8.854\,187\,817\,6 \times 10^{-12}$	[F/m^1]	Permittivity of free space
μ		[H/m^1] OR [N/A^2]	Permeability of a medium
μ_r			Relative Permeability
μ_0	$4\pi \times 10^{-7}$	[H/m^1] OR [N/A^2]	Permeability of free space
λ	$\lambda = c/f$	[m]	Wavelength
c	$299\,792\,458$	[m/s^1]	Speed of Light
h	$6.626\,069\,57 \times 10^{-34}$	[J s]	Planck's constant
ω	$\omega = 2\pi f$	[rad/s^1]	Angular Wave Frequency
k	$k = 2\pi/\lambda$	[rad/m^1]	Wavenumber
A			DE Current Scaling Factor
I_p		[A]	Peak Current for DE Current
α		[s^{-1}]	DE Fall Time Constant
β		[s^{-1}]	DE Rise Time Constant

Nomenclature

Transverse Wave	The motion of a wave is perpendicular to the direction of wave propagation.
Longitudinal Wave	The motion of a wave is parallel to the direction of wave propagation.
FDTD	Finite Difference Time Domain
EM	Electromagnetics
CEM	Computational Electromagnetics
PDE	Partial Differential Equations
FDE	Finite Difference Equations
LIOV	Lightning Induced Overvoltage
LEMP	Lightning Electromagnetic Pulse
RS	Return Stroke
TL	Transmission Line.
MTLE	Modified Transmission Line Exponential.
MTLL	Modified Transmission Line Linear.
DE	Double Exponential.
CP	Current Propagation.
CBC	Channel Base Current.
CFL	Courant-Friedrichs-Lewy.
TE	Transverse Electric.
TM	Transverse Magnetic.
ABC	Absorbing Boundary Conditions
RBC	Radiation Boundary Conditions
PML	Perfectly Matched Layer

Chapter 1

Introduction

Lightning electromagnetic (EM) fields are responsible for deleterious effects in electrical distribution networks. In order to design protection against these effects it becomes necessary to first model the EM fields surrounding the overhead distribution lines. EM fields in this case are directly linked to the current that flows in the lightning channel, known as the return stroke channel. Lightning EM fields are unique in the field of electromagnetic modelling due to the magnitude of return stroke current, as well as the high speed transient nature of the current waveform. Typically EM fields are modelled in the frequency domain, however due to the return stroke current there are a number of benefits to evaluating the problem in the time domain.

This work will review three different methods of evaluating lightning EM fields in the time domain: Finite Antenna, FDTD and Single Cell FDTD. The Finite Antenna Method is the classic approach used in this field, and the second is the Finite-Difference Time-Domain (FDTD) method. More specifically the FDTD method will be implemented in two different variations: 3D FDTD and 2D FDTD. The Single Cell FDTD method is the final technique and is a combination of the previous two. Each of these implementations have their own benefits for specific situations.

Simulating the EM fields that surround a lightning channel is not a trivial problem. There are many different sub system models, and hundreds of variables that construct a full model. Therefore this work answers the question of how EM fields are modelled at a single point in space (located near the lightning channel). This is not simply an implementation, but a methodology to de-constructing this complex problem into manageable components and understanding the limitations and strengths of models. In addition to this all the models used in this work are kept in the time domain.

Another important aspect to highlight is the distinction of Return Stroke electromagnetics (which result from the current flow in the lightning channel), and cloud potential electromagnetics (which result from the charge difference between cloud and ground). With reference to lightning protection on distribution lines it is only the return stroke electromagnetic fields that have a significant effect on the lightning induced overvoltages (LIOV) that develop on the lines [1–4]. Another important consideration is that

this effect is only relevant to lightning strokes that occur near (50 m to 500 m) distribution lines [5]. This distance restriction has an important effect on the models used to simulate lightning EM fields.

Chapter 2 will discuss the approach taken in this research, and add context to the research question and solutions. It will also discuss the concept of a model, and its applicability in the area of lightning research.

Chapter 3 will provide a review of the essential lightning process theory, as well as all the relevant sub models needed by the EM models. This is by no means a fully comprehensive review of these research fields, and is meant only to provide sufficient knowledge in these areas to construct later models. A number of assumptions and validations are also made in this chapter that are used throughout the document.

Chapter 4 presents the basic theory of the Finite Antenna Method, as well as all the mathematical functions describing a solution. Example cases are discussed with field solutions for reference. These fields will then be used for comparison against the FDTD model implementations. This method is the classical choice used in the research problem, however it is limited to simple simulation environments.

Chapter 5 and 6 discuss the 3D FDTD and 2D FDTD theory and implementations of lightning simulations respectively. Chapter 5 will present the 3D FDTD theory and lightning related assumptions in detail. The concepts from this chapter are equally applicable in the special 2D FDTD variation, which is discussed in Chapter 6. The primary advantage of using a 2D FDTD implementation over a 3D implementation is the simulation space size, which ultimately is limited by computer memory and computational time. However a 3D implementation allows for greater complexity in the simulation environment.

Chapter 7 presents the Single Cell FDTD method. This is effectively a hybrid between the Finite Antenna Method, and the FDTD method. It relies on a simple implementation of FDTD in combination with the simplified mathematics from the Finite Antenna Method. In addition to this the method avoids the complex numerical issues of a full FDTD implementation, as well as the complicated mathematics of the Finite Antenna method. The primary disadvantage of this method is that it only caters for simple simulation environments.

Chapter 8 discusses and summarises the findings of the presented research, and Chapter 9 provides the concluding remarks. By combining the findings of this research with the code examples in the Appendix, a reader will be able to not only understand the theory of lightning EM models, but also implement models. In addition to this there is enough insight added to aid in adapting models for more complex simulations.

Chapter 2

Approach Taken

This chapter describes the process involved in decomposing the research question. This is done by providing a problem description, and from this description formulating an appropriate solution. In this work the success of a solution will be judged by the strengths of the models. Lightning is a natural event that is difficult (near impossible) to recreate accurately in a lab environment [6, 7]. Therefore this research uses models as a means of evaluating the research problem. Typically a research project involves steps of identifying a problem, theorising a solution, testing and experimentation, and then critical analysis of results. This research relies on a model based approach for testing and analysis, without any real world experimental results.

2.1 Problem Description

Many areas of the world suffer from high lightning flash rate densities, and South Africa is one of them. The South African Highveld has a typical flash rate density of 15 flashes/year/km^2, and in the south of Mpumalanga a flash rate of up to 50 flashes/year/km^2 [8]. As a result of these high flash rate densities, there is on average R500 Million in lightning related insurance claims in South Africa each year, as well as a lightning death rate that is 4 times higher than the global average [8].

There are a number of areas in engineering where the effects of lightning needs to be considered. One such area is lightning protection for distribution lines [9]. There are two mechanisms of lightning that can affect distribution lines. The first is a "Direct Strike", where a lightning stroke attaches directly to a distribution line, and injects current into the line. This current pulse propagates along the line, and affects all the line components. The second mechanism is an "Indirect Strike", where a lightning stroke terminates on the ground at a location near a distribution line, within the range of 50 m to 500 m [5]. The effects of an indirect strike are typically caused by the coupling of the lightning EM fields to the distribution line [9]. This coupling induces a current along the distribution line which forms an impulse current. This pulse causes a rise in line voltage, and is often referred to as a lightning induced overvoltage (LIOV) [2].

When considering the design of lightning protection for distribution lines, it is the indirect strikes rather than the direct strikes that are of most concern [1–4, 10]. This is due to the fact that although a direct stroke injects a higher current into the line, the frequency of indirect strikes is far higher than direct strikes. In addition to this, the currents associated with a direct strike are high enough for most line protection to activate, and prevent a line surge damaging line components. However for an indirect strike the currents may be low enough to pass standard line protection, and still be high enough to cause damage to line components. On transmission lines the induced overvoltage levels are often not considered high enough to be of major concern, so this effect is primarily associated with low to medium voltage overhead lines. Additional examples of these can also include telecommunications lines, and railway power lines. Distribution line networks are generally quite extensive in most countries, and have a direct effect on many users. This is a real world problem that has a significant financial cost, as well as a significant effect on the lives of people who depend on a stable power supply [1–4].

In order to better protect line equipment, as well as improve electricity supply quality, it becomes necessary to better design lightning protection of distribution lines. This work evaluates the EM models needed in the design process. LIOVs are directly dependant on the lightning EM fields surrounding the lines [2]. This leads to the research question of the work: "**How are the electromagnetic fields from a lightning return stroke modelled at a single point in space near the channel**." By answering this question the process can be repeated for multiple points along a distribution line. These fields are then used in conjunction with a "Field to Line" coupling model [11] to determine the LIOV.

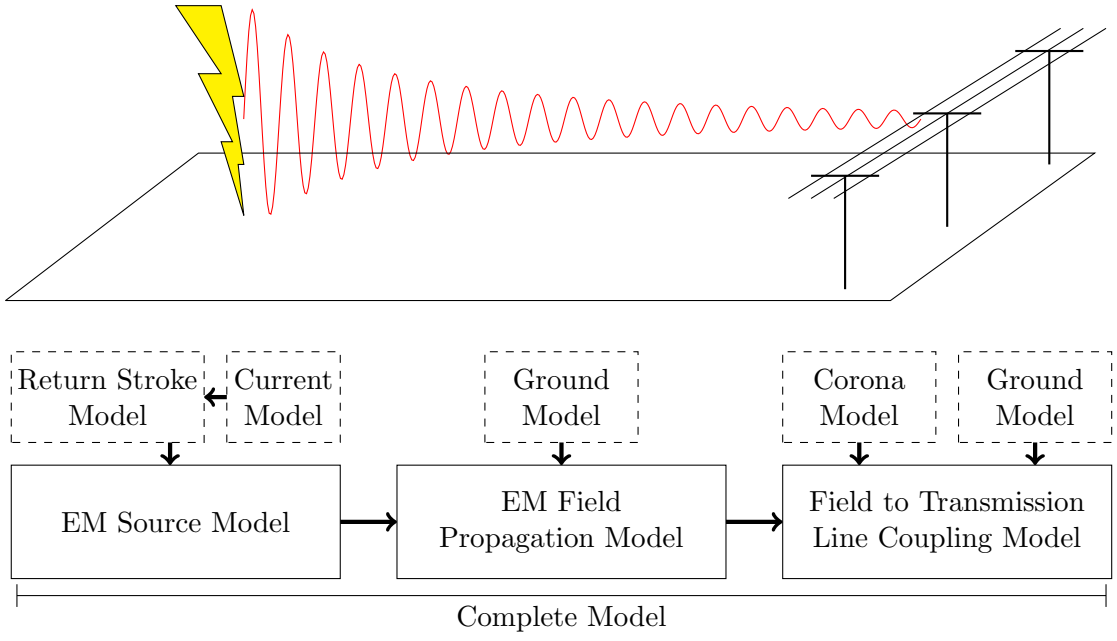


Figure 2.1: Physical system overview. Note: The sinusoidal wave illustrates the propagating EM wave that attenuates with distance. It does not indicate the waveshape of a LEMP.

The answer to the proposed question does not simply take the form of: "Use equation X". Lightning is a complex natural event, as seen in Figure 2.1, which shows the typical sub-models involved with a LIOV simulation [3]. This figure shows three primary models, and their subcomponents [2, 7, 12–14]. Each of these models and subcomponents form their own fields of research. Deciding what models to use, what variables to consider, and how to implement them is a non-trivial process.

This work will focus on the EM source model and the EM propagation models to describe the lightning EM field at a single point in space. However for a full LIOV problem the ideal solution would be to know the EM field at every point in space for every step in time. Therefore simply describing the fields has become a four dimensional (4D) problem before even considering the variables that affect the fields.

2.2 Solution Planning

The research question is now clearly defined, but in order to find a solution it is necessary to define a boundary for the available models. As seen in Figure 2.1, the EM source model is dependant on the lightning current. As will be discussed in Chapter 3, the lightning current waveform is a transient impulse. Because of this there are a number of advantages to evaluating the problem in the time domain. The frequency domain is typically used for EM models, and has been used for LEMP simulations using the method of moments (MoM) implementation [9, 15]. The frequency domain is best suited when dealing with a periodic single frequency source. For LEMP simulations in the frequency domain, the method requires a sample set of periodic sources being transformed into the frequency domain. The resulting fields are then computed in the frequency domain (for each of the sources), and then transformed back into the time domain. The total field is then a summation of multiple sources. This whole process is needlessly complex, and requires a fine resolution of periodic sources that aim to represent the broad band frequency response of a lightning impulse [16]. A time domain solution would not involve this complex management of the current source. This is the primary motivation for evaluating this problem in the time domain in addition to the benefit of the technique being more physically intuitive [13, 17]. Another reason for using the time domain is to include the effects of time varying mediums. The properties of the propagation medium surrounding the lightning channel may change with time due to heating effects and field intensity caused by the lightning current intensity.

With reference to Figure 2.1, the EM source relies on models for the lightning current, and the return stroke model. Both of these have many unique options available, and are already in a time domain format. These models are discussed in Chapter 3.

There are three time domain models for the EM propagation component. The first time domain method is the Finite Antenna Method, which is the classical approach taken for lightning EM fields in the time domain. The second method is the Finite-Difference Time-Domain (FDTD) method, which will be presented in two different implementations: 3D FDTD and 2D FDTD. The final method is Single Cell FDTD, which is a novel combination of the previous two methods. Each of these four implementations have strengths and weaknesses as shown in Table 2.1.

Table 2.1: EM Field Propagation Models - Evaluation Characteristics

	Finite Antenna	**3D FDTD**	**2D FDTD**	**Single Cell**
Single Point EM Fields	✓	×	×	✓
Multiple Points along a line	×	✓	×	×
Complex 3D Environment	×	✓	×	×
Computational Efficiency	✓	×	✓	✓
Non-Ideal ground	×	✓	✓	×
Large Distance Applications	✓	×	✓	✓

Each of these four implementations will be presented in detail in the following chapters. Normally a model is verified through an experimental process, but as stated this is not possible for this particular application. Instead these models will be compared against each other. The first two methods have uniquely different and accepted derivations, and therefore the FDTD implementations will be verified against the Finite Antenna Method. As will be shown, all methods produce the same results when using a simple simulation environment.

2.3 Model Concepts

Lightning is a highly complex natural event that occurs throughout the world in vastly different locations, and environments. Because of this it is important to try and predict the physical effects that result from lightning strokes. This is done by modelling a lightning stroke, as well as the resulting effects. A model can simply be described as a mathematical construct that is used to describe observed phenomena [18]. If every aspect of a physical event was understood, then there would be no need for a model. A model is used when some aspects of a physical process is not fully understood. An example of this is a lightning return stroke (RS) model. The current at every point in a lightning channel cannot be measured [19]. Therefore a RS model is used to mathematically describe the spatial and temporal variation of the current along the channel.

When using a model it is important to remember that it aims to describe the observed phenomena, and not the process itself. As such, a model may use unproven theories, and constants that have no physical meaning, but result in the observed outcomes. Because of this, a requirement of a model is that it must be able to predict alternative physical events (other than the observed case). The strength of a model is described by the range of observed outcomes that can be accurately described by a model.

The limitations of lightning models are important. There are so many variables that affect the physical effects, and additionally lightning is difficult to measure. This is due to the fact that lightning strokes are not repeatable. Each stroke is unique, and therefore its location, current path, current peak, current waveshape, as well as the environmental factors, cannot be controlled. This makes model reconstruction difficult due to the

lack of measurements. This also means that there are many different models that each operate under a select range of variables.

Regardless of this limitation the use of models in lightning simulations is invaluable. When creating a model there is always a feeling of needing to incorporate as many variables possible into the model, so as to better approximate the real results. However there is a limit to the value that this adds. For example, consider a real lightning stroke with a number of branches (shown in Figures 3.1 and 3.2). If a RS model were to consider the spatial and temporal variation of current in each of the branches, as well as the primary channel, then the resulting EM field models would probably match the expected fields more closely, however the benefit of this option needs to be considered. Firstly, would it be possible to measure or model this current distribution, and if it were, how complex would the model be. The second consideration is how much value would it add to the results. Do these branches have a significant effect. The third consideration is whether this model would add real value. Even if a single event could be modelled perfectly, and the results match recorded values perfectly, it would never be able to predict future events because it would not be possible to predict the exact lightning current path of a future event. Every lightning stroke is unique, and constructing a general model that applies well to most lightning strokes is more useful than a model that only fits a single event.

As such, the field of lightning electromagnetic research relies on theoretical models more than experimental models, to aid in predicting the effects caused by future lightning events. This choice is purely caused by the difficulty in making lightning EM measurements.

2.4 Chapter Summary

This chapter has described the research problem and identified the research question as being: “How are the electromagnetic fields from a lightning return stroke modelled at a single point in space near the channel?”. A guide to the proposed solution was derived from understanding the problem. The solution involves constructing time domain models for the systems shown in Figure 2.1. Four different EM field propagation models have been proposed, and the advantage of each are shown in Table 2.1.

The next chapter will discuss the relevant background theory to the lightning process, as well as the basics of electromagnetic theory. It will also discuss in detail the models required for the EM Source model in a lightning EM simulation.

Chapter 3

Background

This chapter discusses the theory of lightning and highlights the various different processes of a lightning event, as well as their respective model components. An experienced lightning researcher may skip this chapter, however for a new lightning researcher this chapter aims to present all the important theory relevant to the problem of lightning electromagnetics, and more specifically creating a model for the EM Source.

Section 3.1 defines the correct lightning terms and development processes involved in lightning strokes. Section 3.2 discusses the current impulse models used to represent lightning currents in models. Section 3.3 discusses the Return Stroke models used to describe the spatial and temporal current values along a lightning channel. And Section 3.4 and 3.5 gives a brief overview of Maxwell's equations, and the co-ordinate systems used in 3D models.

3.1 Lightning Flash

This section presents the theory regarding a lightning flash and lightning stroke. A lightning flash, or lightning event is a complex process with numerous physical reactions. As such, this section is not meant to be a comprehensive review of all the different physical events, but will rather focus directly on the physics that will aid in understanding how a lightning flash creates a lightning electromagnetic pulse. First the different terminology will be discussed, and then the full lightning stroke process will be presented.

There are four main classifications of cloud to ground lightning: Downward Negative, Downward Positive, Upward Negative, and Upward Positive [20]. Regardless of the type, lightning is simply described as the movement of electrical charge between clouds, and between clouds and ground. The process of how charge accumulates in clouds is beyond the scope of this work, but essentially it forms as a result of the physical movement between the three states of water that form and move through clouds during storms.

Figure 3.1 shows an example of a "Cloud to Cloud" lightning stroke. Notice that the stroke travels horizontally, and does not terminate on the ground.

Figure 3.1: Lightning - Johannesburg, South Africa. Figure shows the horizontal channel traversing the sky. Taken by Author.

Figure 3.2: Lightning - Johannesburg, South Africa. Figure shows the vertical channel with numerous downward branches. Taken by Author.

Figure 3.2 shows an example of a downward "Cloud to Ground" lightning stroke. Ninety percent of cloud to ground lightning is Downward Negative, which occurs when negative charge moves down towards earth from the cloud [20]. This work will only consider negative downward lightning.

It is important to recognize the difference between a lightning flash, lightning strike, and lightning stroke. The term "strike" is the incorrect terminology used in describing a lightning event, which should be described as a lightning "flash". The word "strike"

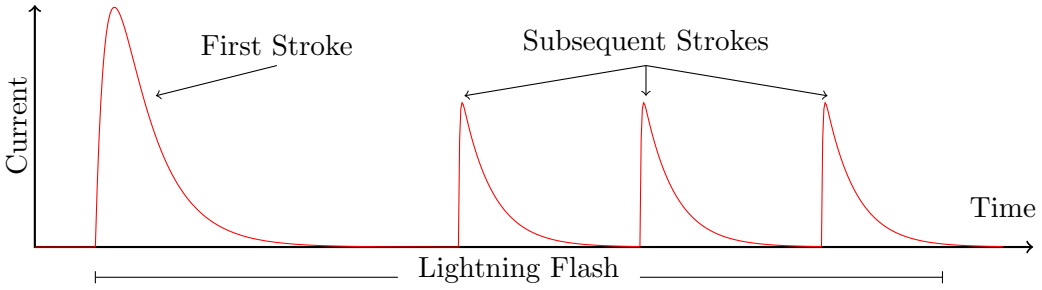


Figure 3.3: Lightning Flash, and its Stroke Components

insinuates a physical force exerted by the lightning, which is not the case. However there are certain terms such as "Direct Strike" and "Indirect Strike" that are commonly used. The difference between these two terms is purely based on English interpretation, but for the remainder of this dissertation the term lightning *flash* will be used to describe a lightning event [20]. A lightning flash is an event seen by an observer as a bright pulsing light typically followed by a loud clapping sound. A lightning flash consists of multiple lightning strokes, as seen in Figure 3.3. The relevance of the current and the waveshape in this figure will be discussed shortly.

A lightning flash typically consists of a first stroke followed by a number of subsequent strokes. Each stroke consists of a downward leader followed by an upward return stroke.

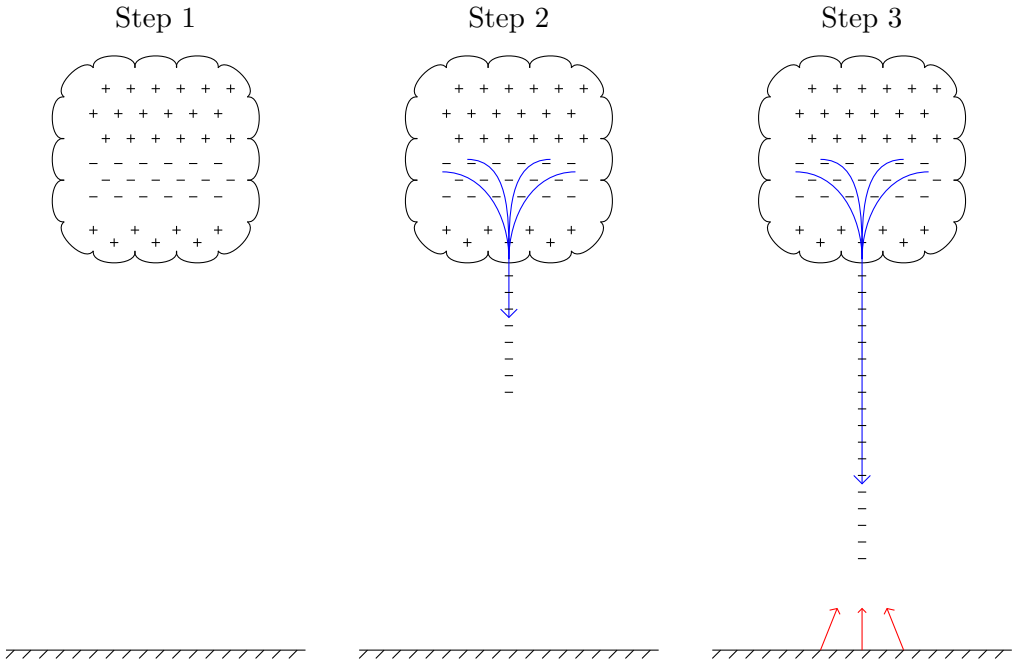


Figure 3.4: Lightning Stroke. Steps 1-3

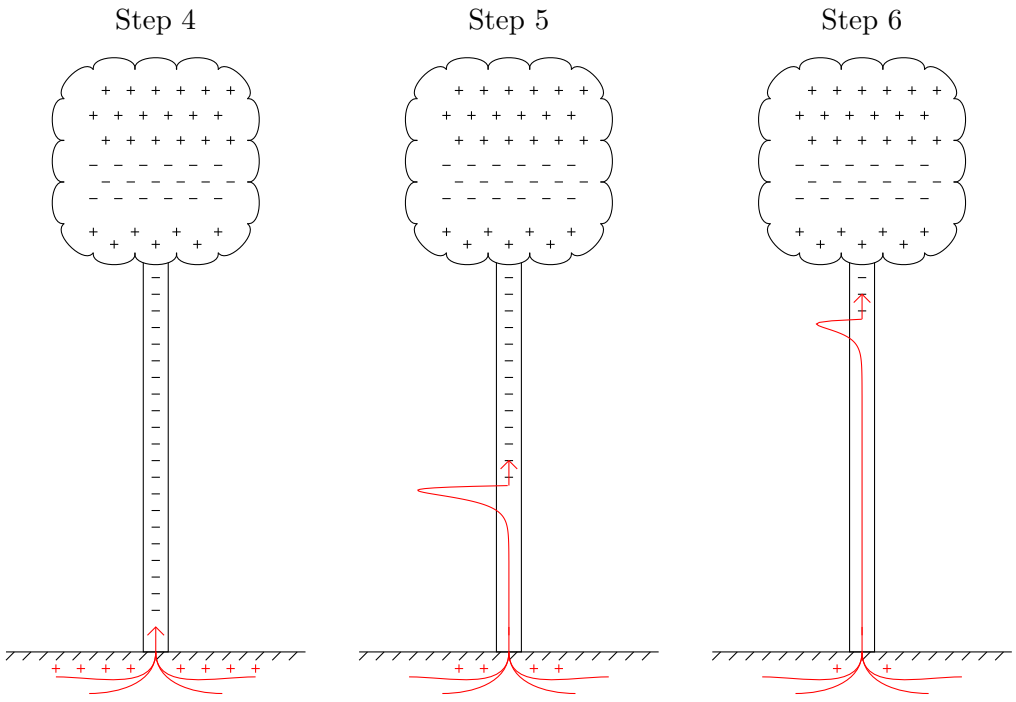


Figure 3.5: Lightning Stroke. Steps 4-6

The physics regarding a lightning first stroke is illustrated in Figures 3.4 and 3.5, and described in 6 steps [20–22].

Step 1 A charged cloud is formed, with a typical three layer charge distribution [23].

Step 2 A stepped downward leader is formed. The stepped leader propagates downward in bursts, and often forms branches (not shown in the figure). As each stepped path is formed, negative charge is deposited in the channel [20].

Step 3 As the stepped leader gets closer to ground, the electric field strength increases and upward leaders form. These upward leaders cause a slow increase in positive charge to flow up toward the leader.

Step 4 The stepped downward leader then attaches to an upward leader. The positive ground charge then "sees" a conductive path from the ground up to the cloud.

Step 5 An impulse of positive charge then rushes up into the channel to neutralize the negative charge. This impulse is known as the Return Stroke current.

Step 6 As the positive impulse travels up the leader channel, the current is attenuated due to charge neutralization, as well as charge loss in the corona sheath [24].

The upward flowing current impulse is known as the return stroke current. The current measured at the RS ground point is known as the channel base current (CBC). The

channel base current typically has a peak current value in the range of 4 kA - 90 kA [25], and it is this return stroke current (not leader current) that is responsible for lightning damage to ground objects, either through direct attachment to the ground object (direct strike) or induced effects (indirect strike) [18]. Figure 3.3 shows the first return stroke current, followed by subsequent return stroke currents. The first return stroke has a slower (longer) wavefront, but generally has a higher peak current. Subsequent strokes typically have a faster (shorter) wavefront, but a lower peak current [20]. This makes intuitive sense as the first return stroke has to form the conductive channel between the cloud and ground. The subsequent strokes are able to flow in the pre-established conductive channel.

It is worth mentioning that there are two distinct processes responsible for creating EM fields during a lightning flash. The difference in charge between the cloud and ground establishes a semi-static vertical electric field. This field intensity drops in steps after each stroke as charge is neutralised. This collapsing field results in a propagating EM field, with a predominantly vertical electric field. The second process is the EM fields radiated from the RS channel current. The RS channel is considered as a vertical antenna above the ground, which radiates EM fields in the Vertical and Horizontal planes. It is the horizontal fields that have the greatest effect on the LIOVs induced on distribution lines [9, 26]. This work is only concerned with the EM field components created by the RS channel current, and not the fields created by the charge distribution between the cloud and ground.

This section has highlighted the elements and theory of a lightning flash. To simulate the expected EM fields radiating from a lightning stroke it is necessary to create a model that describes the current distribution on the channel during the lightning stroke. This is known as a Return Stroke (RS) model. This model depends on a CBC model to describe the current impulse waveshape.

3.2 Current Impulse Models

A channel base current model is defined by a current impulse model. This model is a mathematical description of the observed current flow at the base of the lightning channel (point of attachment). Three current impulse models are reviewed here: Double Exponential, Heidler, and Terespolsky functions [27] (Appendix A).

1. **Double Exponential (DE)**

$$i_{DE}(t) = I_p A \cdot (e^{-\alpha t} - e^{-\beta t}) \tag{3.1}$$

Where

A	=	Scaling Factor
I_p	=	Peak Current
α	=	Time Constant
β	=	Time Constant

This function has been used in a number of lightning related research areas, however as seen in Figure 3.6 and 3.7, the wavefront rises instantaneously at time 0 s.

This is physically unrealisable in the real world, and does not match the physics described in Section 3.1. Section 3.1 explains that the current should rise slowly as the leader approaches the ground, and then rise rapidly after attachment occurs. This rising edge of the wavefront is important, and has a significant effect on the EM fields produced [28].

2. **Heidler Function**

$$i_H(t) = \frac{I_p}{\eta} \cdot \left(\frac{(\frac{t}{\tau_1})^n}{(\frac{t}{\tau_1})^n + 1} \right) \cdot e^{-\frac{t}{\tau_2}} \tag{3.2}$$

Where

η	=	Scaling Factor
I_p	=	Peak Current
n	=	Steepness Factor
$\tau 1$	=	Time Constant
$\tau 2$	=	Time Constant

The Heidler function is a widely used current impulse model, and is also used in lightning protection standards [25]. Figures 3.6 and 3.7 show that the waveshape of the Heidler function is in better agreement with the physics described in Section 3.1. The only problem with this function is that it cannot be analytically integrated [28–30]. This becomes relevant when evaluating the EM fields.

3. **Terespolsky Function**

$$i_{BT}(t) = \frac{I_p}{\eta} \cdot \left(1 - e^{-\omega_0 t} \left(\sum_{j=0}^{n_a} \frac{\omega_0^j t^j}{j!} \right) \right) \cdot e^{-\frac{t}{\tau_2}} \tag{3.3}$$

$$\int i_{BT}(t)\, dt = \frac{I_p \tau_2 e^{-t(\frac{1}{\tau_2}+\omega_0)}}{\eta} \cdot \left(-e^{\omega_0 t} + \sum_{i=0}^{n_a} \sum_{j=0}^{i} \frac{\omega_0^i \tau_2^j t^{i-j}}{(i-t)!(\tau_2\omega_0+1)^{j+1}} \right) + C \tag{3.4}$$

$$i'_{BT}(t)_{BT} = \frac{I_p}{\eta} \cdot \left[\frac{e^{-\omega_0 t} \omega_0^{n_a+1} t^{n_a}}{n_a!} - \frac{1}{\tau_2} + \frac{e^{-\omega_0 t}}{\tau_2} \left(\sum_{j=0}^{n_a} \frac{\omega_0^j t^j}{j!} \right) \right] \cdot e^{-\frac{t}{\tau_2}} \tag{3.5}$$

Where

η	=	Scaling Factor
I_p	=	Peak Current
n_a	=	Steepness Factor
ω_0	=	Rise Time Constant
$\tau 2$	=	Time Constant

Equation 3.3 shows the Terespolsky function, which is a Heidler Function approximation [30, 31]. Figures 3.6 and 3.7 show the Terespolsky function has a maximum error of 1.5% when compared to the Heidler function. This function has the benefit of an analytical integral solution (Equation 3.4), as well as an analytical differential solution (Equation 3.5).

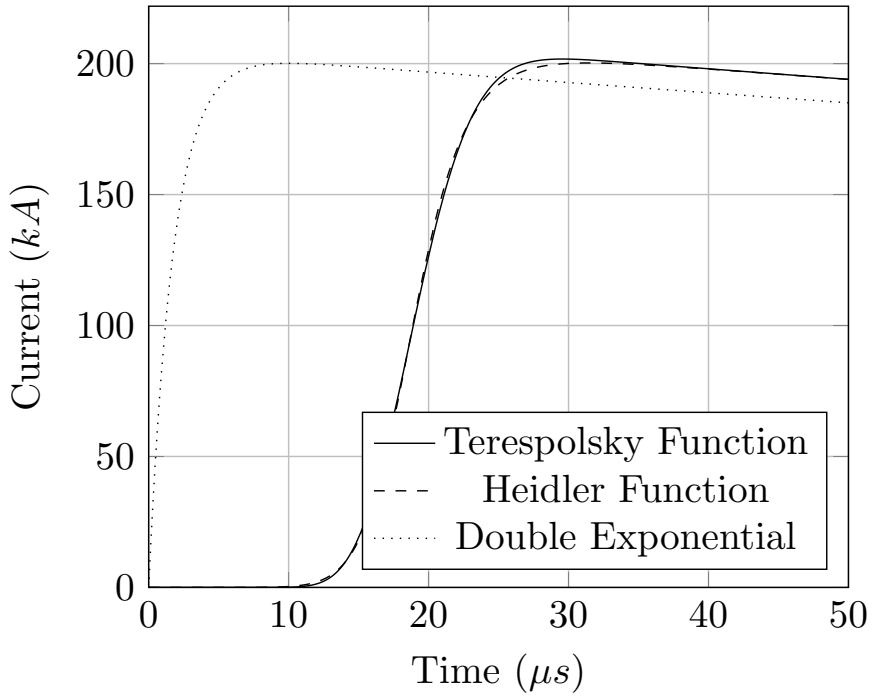


Figure 3.6: DE, Heidler, Terespolsky function plot - Rise Time

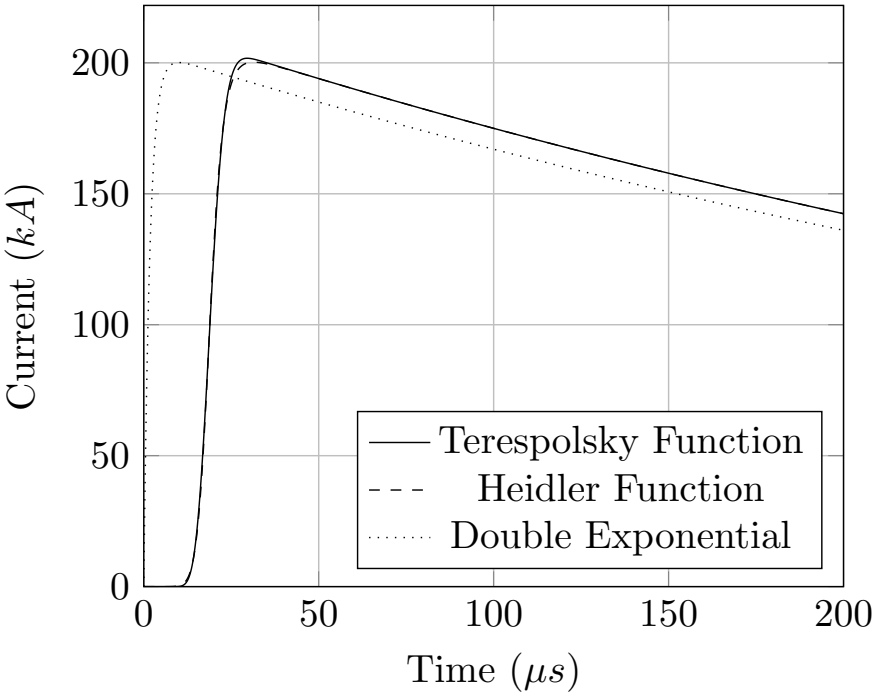


Figure 3.7: DE, Heidler, Terespolsky function plot - Wave Shape

Table 3.1: 200 kA Current Impulse Parameters

Variables	**DE**	**Heidler**	**Terespolsky**
I_p	200 kA	200 kA	200 kA
A	1.025	-	-
η	-	0.93	0.93
α	$2.05 \times 10^{-3}\,\mathrm{s}^{-1}$	-	-
β	$5.64 \times 10^{-5}\,\mathrm{s}^{-1}$	-	-
n	-	10	-
n_a	-	-	33
$\tau 1$	-	19 µs	-
$\tau 2$	-	485 µs	485 µs
ω_0	-	-	$1\,768\,211\,\mathrm{s}^{-1}$

Figures 3.6 and 3.7 show the three impulse models plotted on different time ranges. Figure 3.6 emphasizes the differences in the wavefront rise time characteristics, and clearly shows that the DE model is not physically realisable with an instantaneous rise time [29]. The waveshape shown in the figures is a 200 kA, 10/350 µs impulse. This is the recommended waveshape for a level one lightning protection standard (IEC 62305-1) for a first positive RS [25]. The parameters used in the current impulse models are shown in Table 3.1 [25, 32]. The standard also states that for a lightning protection level one (the highest level), a first negative stroke current impulse should have a peak of 100 kA, and a subsequent stroke should have a peak of 50 kA [25].

Popular Current Impulse Model

A popular current impulse model, commonly used in research papers, was first presented by C.A. Nucci [33]. The model, described by Equation 3.6, is a combination of the Heidler Function (with $n = 2$) and a DE function. The waveshape is intended to represent a subsequent RS with a fast rise time.

$$i(t) = \frac{I_1}{\eta} \cdot \left(\frac{(\frac{t}{\tau_1})^2}{(\frac{t}{\tau_1})^2 + 1} \right) \cdot e^{-\frac{t}{\tau_2}} + I_2(e^{\frac{-t}{\tau_3}} - e^{\frac{-t}{\tau_4}}) \tag{3.6}$$

However due to the Heidler Function component, this equation cannot be analytically integrated. The Terespolsky equivalent function ($n_a = 3$) is described by Equation 3.7.

$$i(t) = \frac{I_3}{\eta} \cdot \left(1 - e^{-\omega_0 t} \left(\sum_{j=0}^{n_a} \frac{\omega_0^j t^j}{j!} \right) \right) \cdot e^{-\frac{t}{\tau_2}} + I_2(e^{\frac{-t}{\tau_3}} - e^{\frac{-t}{\tau_4}}) \tag{3.7}$$

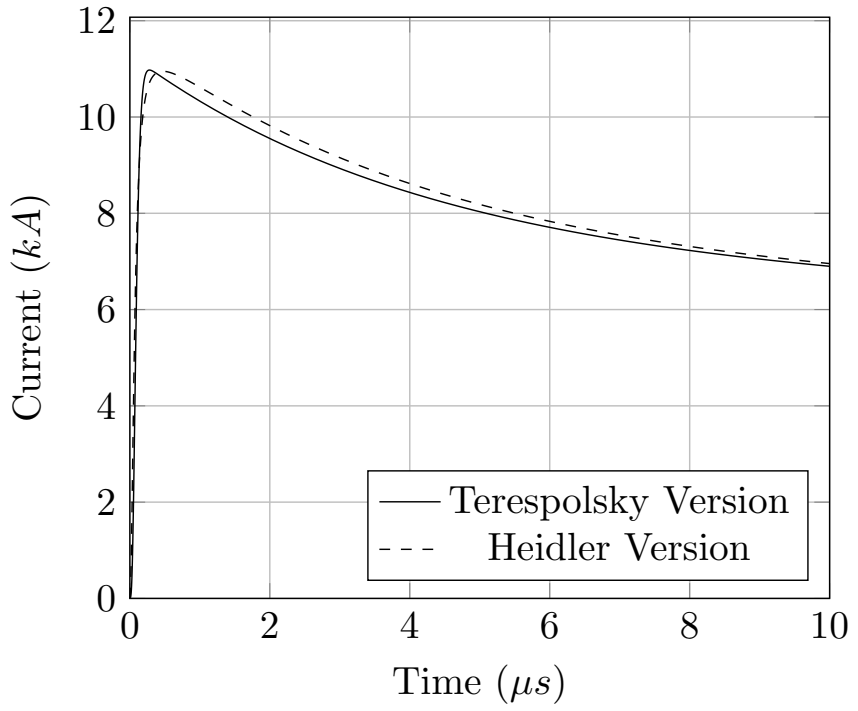


Figure 3.8: Subsequent Stroke Current impulse plot - Function comparison.

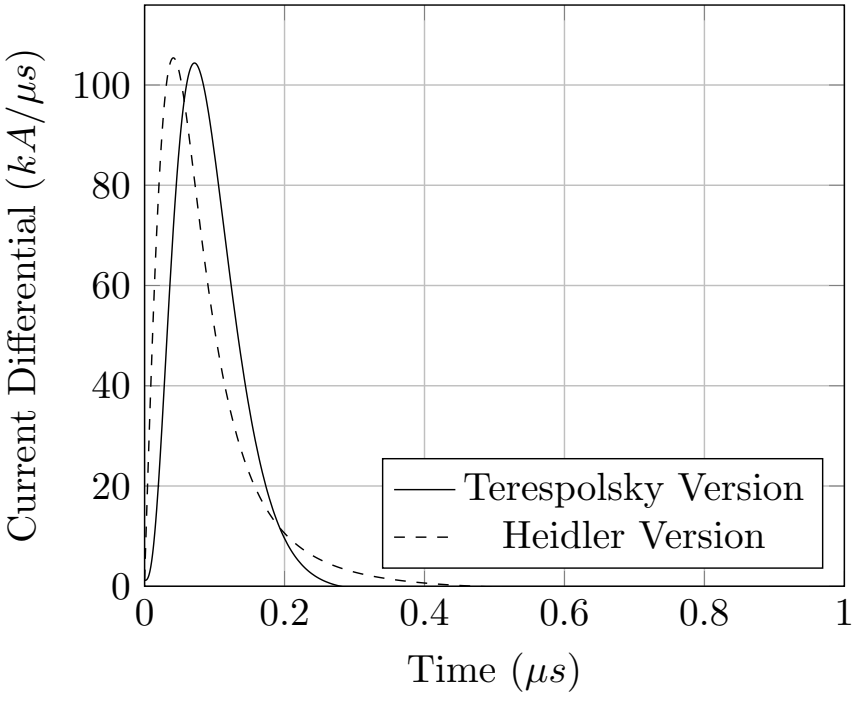


Figure 3.9: Subsequent Stroke Current impulse differential plot - Function comparison.

Table 3.2: Subsequent Stroke Current Impulse Parameters

Variables	**Heidler Version**	**Terespolsky Version**
I_1	9.9 kA	-
I_2	7.5 kA	7.5 kA
I_3	-	9.55 kA
η	0.845	0.845
n	2	-
n_a	-	3
ω_0	-	41.66 /(µs)
$\tau 1$	0.072 µs	-
$\tau 2$	5 µs	5 µs
$\tau 3$	100 µs	100 µs
$\tau 4$	6 µs	6 µs

Figure 3.8 shows the small difference between the current impulse function described by Nucci in [33], and the equivalent Terespolsky function. Figure 3.9 shows a plot of the time differential current. Although this figure shows a negligible time difference between the functions, the important characteristic is the peak differential value. Therefore using the Terespolsky version is acceptable, and the values it produces in later simulations can be compared to other research texts using Equation 3.6.

Table 3.2 shows the parameter values used to create the impulse plots in Figures 3.8 and 3.9. The values used in the Heidler version are taken directly from [33]. Note the difference between the n and n_a values, as well as the τ_1 and ω_0 values used in Table 3.1 and 3.2. When adapting a Heidler Function to a Terespolsky Function, the choice of these values is critical. For a Heidler Function with $n = 10$ (which is recommended in lightning protection standard [25]), the choice of n_a is set to 33.

However when n does not equal 10, such as Equation 3.6, determining the values for n_a and ω_0 is complicated. From author experience the use of Equation 3.8 is recommended. This reduces the problem to one unknown variable. In addition to this n_a can only be a whole number. This equation serves as a rough guide, and will not provide the optimal match.

$$\omega_0 = \frac{n_a}{\tau_1} \tag{3.8}$$

The choice of the waveshape and parameters used in the different current impulse models is left at the readers discretion. The values presented in this section serve only as an example, and not necessarily the best choice for simulations. In addition to this it is recommended that the DE function is not used in electromagnetic simulations. Although this function is simpler to manipulate mathematically, its waveshape does not match the observed characteristics. Therefore the Heidler Function or Terespolsky Function should be used.

3.3 Return Stroke Models

A Return Stroke (RS) model is simply a mathematical expression that describes current along a lightning channel, at different points in time. RS models are needed by electromagnetic field calculations, because the spatial and temporal variation of current is the source of electromagnetic radiation [18]. There are other uses of RS models, however these will not be discussed here.

With reference to Section 2.3 it is important to identify the limits and assumptions of the RS models presented in this section. One assumption used in all the models is that the RS path is perfectly straight and perpendicular to a flat ground plane. From Figure 3.2 it is clear that this assumption does not accurately represent a real lightning channel. A real lightning channel consists of multiple tortuous paths, along with many side branches that do not terminate. Adding these parameters to a model would not contribute significant value to a simulation due to the fact that every lightning stroke is unique, and therefore would not be applicable to any other event. The best decision is to approximate the lightning channel to being perfectly straight and perpendicular to the ground. Other assumptions include a finite channel height, and a constant velocity for the upward travelling current impulse.

Figure 3.10 is a graphical representation of the physical system being modelled by the return stroke model. This figure shows the model assumptions already mentioned, and also shows that the RS current originates at the ground plane attachment point, and travels up the channel with a finite velocity.

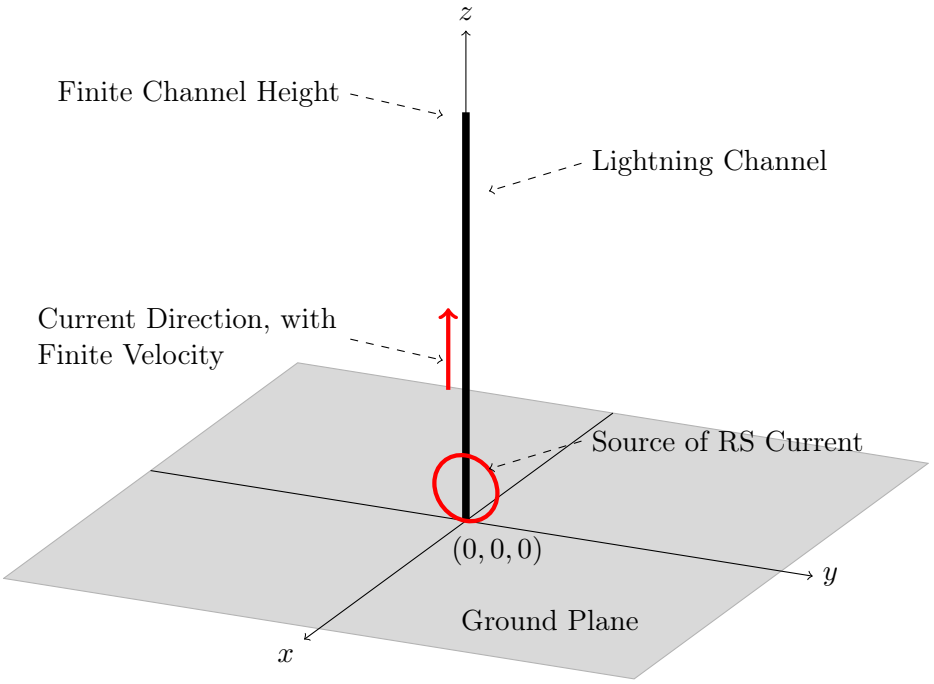


Figure 3.10: Basic RS Physical Model

There are 4 categories of RS models:

Electro-Thermodynamic models Models of this sort aim to describe a number of lightning phenomena, such as current-voltage characteristics, acoustic radiation, plasma development, plasma heating, and plasma channel dimensions [18]. This category is more relevant for researchers in the physics field.

RLC Transmission line models This method models the current path between ground and cloud as a transmission line consisting of Resistive (R), Inductive (L) and Capacitive (C) elements. These RCL per unit components shape and attenuate the current impulse as it travels upward along the channel [18]. This model requires a complicated circuit model to define the current channel.

Electromagnetic models These models rely on the theory of EM radiation through different mediums. By changing the properties of the medium surrounding the current carrying lightning channel, it is possible to shape, attenuate and slow the RS current [15]. This model integrates directly into Finite-Difference Time-Domain (FDTD) simulations [34–37], but it does add additional complexity.

Engineering Models These models are a division of the transmission line models, where the transmission line between cloud and ground is ideal (no loss or components). These models represent the spatial and temporal variation of the current mathematically.

Of these four types only the Engineering Models will be reviewed. The first two categories do not integrate well with the work to follow. Electromagnetic models are a good solution when working with FDTD, as they provide a single model solution to describe RS current as well as the resulting EM fields. The Engineering models used in this section are actually a special case of Electromagnetic models known as the "Phased Current Source Array" or "Type 7" [37], and therefore integrate easily into FDTD simulations. Due to this, only the Engineering models will be discussed, however the reader needs to be aware that the Electromagnetic RS models do exist, and do have additional advantages when using FDTD [15, 34, 36, 37].

Engineering RS Models

Engineering RS models are recognised for relating the current along the channel (in space and time) to the current at the channel base [15, 18, 34]. These models also describe the speed and current attenuation of the impulse as it propagates upward along the channel. There are three subcategories of engineering models: Current Propagation (CP), Current Generation (CG) and Current Dissipation (CD) models [18]. Only the CP models will be presented here, as they produce all the required results, and are simple to implement. Unless specified otherwise, "RS models" will refer to "Engineering Current Propagation Return Stroke models". The channel created by the downward leader is described by CP models as a uniform lossless transmission line (TL). In these models the RS current propagates up the channel with a mathematically described current

attenuation and speed. Equation 3.9 shows a generic format for the RS model.

$$i_{RS}(z,t) = A_{tt}(z) \cdot i_{chan}(z,t) \qquad t > \frac{z}{v_c} \tag{3.9a}$$

$$i_{RS}(z,t) = A_{tt}(z) \cdot i_{chan}(z,t) \cdot U(t - \frac{z}{v_c}) \tag{3.9b}$$

$$i_{RS}(z,t) = A_{tt}(z) \cdot i_{chan}(0, t - \frac{z}{v_c}) \cdot U(t - \frac{z}{v_c}) \tag{3.9c}$$

Equations 3.9a, 3.9b and 3.9c are all equivalent, and show the standard format of the RS model, where $i_{RS}(z,t)$ describes the current at channel height (z) in time. The function $A_{tt}(z)$ controls the attenuation of the channel current (i_{chan}) with height. Equation 3.9a also shows that the function can only exist for values of $t > \frac{z}{v_c}$ (due to the finite travel time of the wavefront). Equation 3.9b includes a unit step function to describe the time value limitation of Equation 3.9a. Equation 3.9c shows that the height component can be considered as a time shift. The importance of this is shown in Equation 3.10, where the channel current at ground level is equivalent to the channel base current.

$$i_{CBC}(t) \equiv i_{chan}(0,t) \tag{3.10}$$

Equation 3.11 shows an alternative notation for the generic return stroke models as described in Equations 3.9. In this function the $i_{CBC}(t)$ can be any acceptable current impulse function, as described in Equations 3.1, 3.2 or 3.3.

$$i_{RS}(z,t) = A_{tt}(z) \cdot i_{CBC}(t - \frac{z}{v_c}) \cdot U(t - \frac{z}{v_c}) \tag{3.11}$$

There are three commonly used RS models which use the same format as Equation 3.11, except they differ in their implementation of the current attenuation function $A_{tt}(z)$.

Transmission Line with no current decay - TL

$$i_{RS}(z,t) = 1 \cdot (i_{CBC}(t - \frac{z}{v_c})) \cdot (U(t - \frac{z}{v_c})) \tag{3.12}$$

Modified Transmission Line with Linear current decay - MTLL

$$i_{RS}(z,t) = (1 - \frac{z}{H}) \cdot (i_{CBC}(t - \frac{z}{v_c})) \cdot (U(t - \frac{z}{v_c})) \tag{3.13}$$

Modified Transmission Line with Exponential current decay - MTLE

$$i_{RS}(z,t) = \exp(-\frac{z}{\lambda_c}) \cdot (i_{CBC}(t - \frac{z}{v_c})) \cdot (U(t - \frac{z}{v_c})) \tag{3.14}$$

Equation 3.12 is the TL model, and it has no current attenuation along the line. Equation 3.13 is the MTLL model, which has a linear current attenuation with increasing height. Equation 3.14 is the MTLE model, which has an exponential current decay with increasing height.

All of these models use a constant velocity v_c to describe the RS wavefront motion over the entire channel. Table 3.3 shows some variations of v_c values as well as paper references. In a physical lightning stroke event the RS wavefront velocity v_c does change with height ($v_c(z)$). Therefore using a constant v_c value is not entirely accurate, however the complexity of adding a height variable velocity to the system model is high, and the overall change in outcome is not significant enough to validate using it. Therefore the velocity is kept constant between $\frac{1}{4}\,c$ and $\frac{1}{2}\,c$.

Table 3.3: Return Stroke Wavefront Velocity

Paper	**Velocity** $\mathrm{m\,s^{-1}}$	**Speed of Light** c
Uman *et al* [38]	80000000	$0.267\,c$
ISHII *et al* [39]	99930819	$\frac{1}{3}\,c$
Napolitano *et al* [13]	149896229	$\frac{1}{2}\,c$
Jiang *et al* [40]	110000000	$0.367\,c$
Azzouz *et al* [41]	100000000	$0.334\,c$
Popov *et al* [42]	120000000	$0.4\,c$

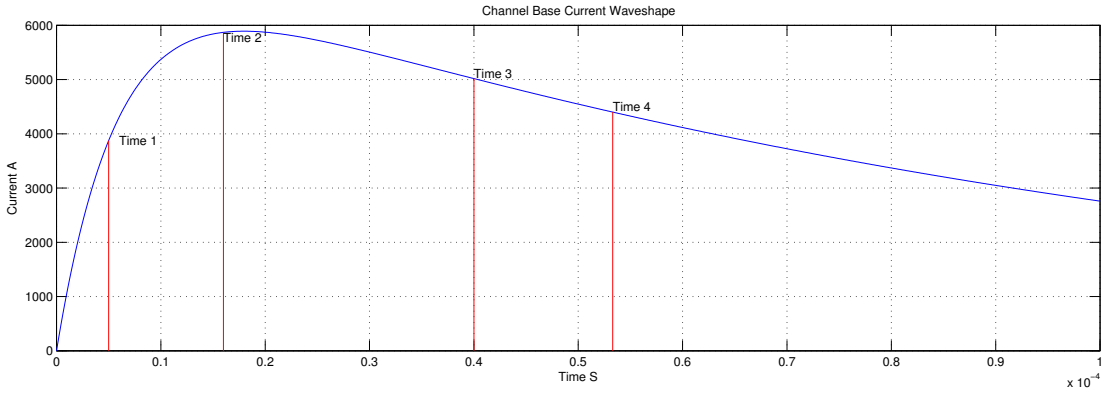
In Equation 3.13 the constant H represents the channel height. Table 3.4 shows a selection of different channel heights used in past research papers. The channel height ranges between 2000 m to 8000 m. The value for λ_c in Equation 3.14 is selected to be 2000 m [18, 24, 33, 43].

Table 3.4: Return Stroke Channel Height

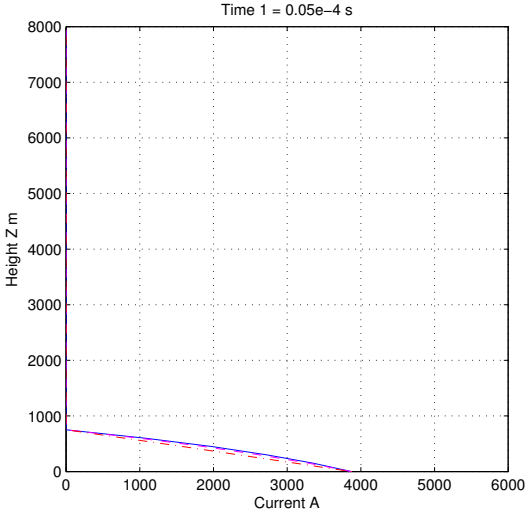
Paper	**Min Height** m	**Height** m	**Max Height** m
V. Cooray [18]	-	7500	-
Baba *et al* [15]	5000	-	8000
Jiang *et al* [40]	-	7500	-
S. Rusck [44]	-	2000	-
Bo *et al* [45]	-	7500	-
Uman *et al* [19]	2700	-	8000
Uman *et al* [38]	-	4000	-

Figure 3.11 shows the difference between the three RS models. Figure 3.11a shows the CBC used for all three functions. The CBC is described by a DE current impulse function (Equation 3.1), where $A = 1$, $I_p = 7500\,\mathrm{A}$, $\tau_3 = 100 \times 10^{-6}\,\mathrm{s^{-1}}$, $\tau_4 = 6 \times 10^{-6}\,\mathrm{s^{-1}}$, $\alpha = 1/\tau_3$ and $\beta = 1/\tau_4$. The DE function, and its parameters do not represent a realistic RS current, and were chosen to demonstrate the impulse waveshape. This figure also shows four randomly selected time points during the CBC. Assuming that the reference time for this example begins at 0 s, then after 5 µs the current passing the channel base is 3800 A. During this time current has moved up the lightning channel, as seen in Figure 3.11b. Figures 3.11b to 3.11e show the current along the lightning channel at the different times of the RS channel base current. The "X" axis plots the current, and the "Y" axis plots the channel height.

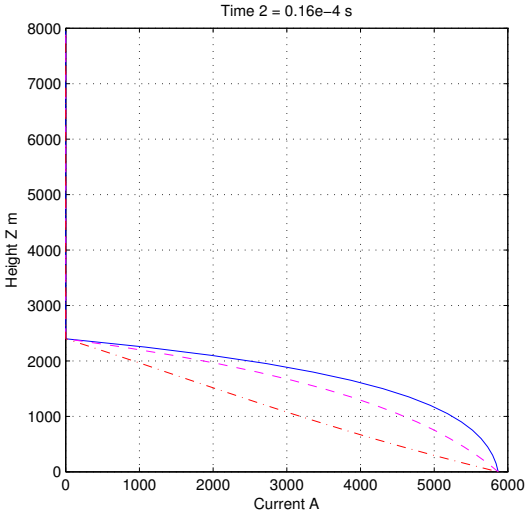
These figures plot the three different RS models (with v_c=0.5 c). The "Solid" (Blue) line is the TL model. The "Dashed" (Purple) line is the MTLL model (with H=8000 m), and the "Dashed and Dotted" (Red) line is the MTLE model. At time step 1, as seen



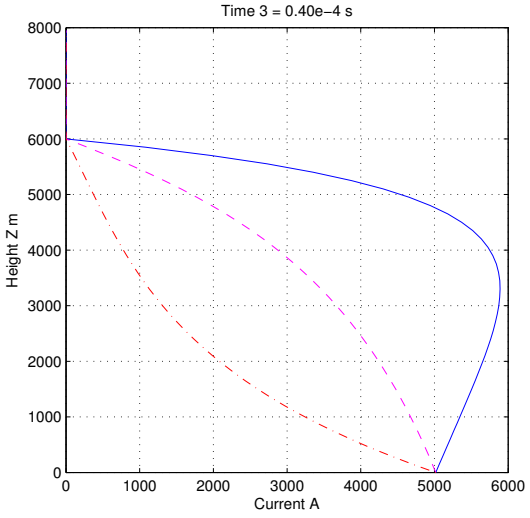
(a) RS Channel Base Current



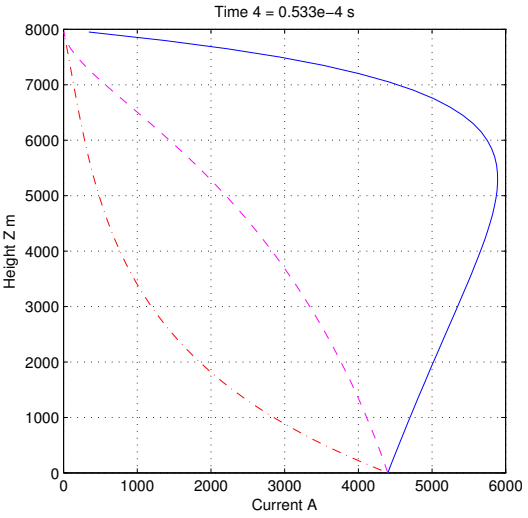
(b) Time Step 1



(c) Time Step 2



(d) Time Step 3



(e) Time Step 4

Figure 3.11: Engineering RS Models for: TL, MTLL and MTLE.

in Figure 3.11b, the wavefront has not had enough time to move far up the channel, and therefore the attenuation functions are not significantly affecting the currents. Progressing in time, Figures 3.11b to 3.11e show that the TL model current is unattenuated as it propagates up the channel, and the waveshape remains the same as the CBC. As expected, the MTLL model decays linearly as the wavefront moves up the channel, and the MTLE (with the lowest current at the relevant heights) decays exponentially with height. These figures show that the RS model chosen has a significant effect in representing the RS current on the channel.

The MTLL model was first proposed by Rakov *et al* in [46], and is typically more accurate when measurements are made near (50 m) to the ground attachment point (rocket triggered lightning) [21]. The MTLE model was first proposed by Nucci *et al* in [24]. According to the theory outlined in Section 3.1, the RS current is observed to attenuate at height as the CBC neutralises the negative charge in the channel. Therefore either of the modified TL models more accurately represent the observed process [12]. However the standard TL method is commonly used in literature, and for convenience in comparing results with other work the TL method will be used for the remainder of this work [16].

3.4 Electromagnetic Theory

The purpose of this section is to introduce the basics regarding electromagnetic theory and Maxwell's equations. It is not intended as a complete review of the theory. For a more in-depth analysis the reader is referred to [47–49]. There are four equations that make up Maxwell's Equations:

Faraday's Law This law essentially states that a time varying magnetic field induces an electric field. This is seen in Equation 3.15, with the integral form on the left hand side (LHS), and the differential form on the right hand side (RHS). The differential form results from applying Stoke's theorem to the integral form. An alternative loose interpretation of the law is that magnetic field lines passing through a surface is equal to the electric field lines enclosing the surface. This is illustrated in Figure 3.12.

$$\oint_c \vec{E} \cdot dl = -\int_s \frac{\partial \vec{B}}{\partial t} \cdot ds \qquad \rightarrow \qquad \nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{3.15}$$

Gauss's Law This law is based on the observations of Gauss who correctly identified that electric charges attract or repel due to a force that is related to the distance between the charges. This force is caused by the electric flux density (D) that emanates out of (or into) an electric charge. The LHS of Equation 3.16 states that the charge density within a volume is equal to the electric flux density on the surface of a conceptual container surrounding the charge.

$$\oint_s \vec{D} \cdot ds = \int_v \widetilde{\rho}\, dv \qquad \rightarrow \qquad \nabla \cdot \vec{D} = \widetilde{\rho} \tag{3.16}$$

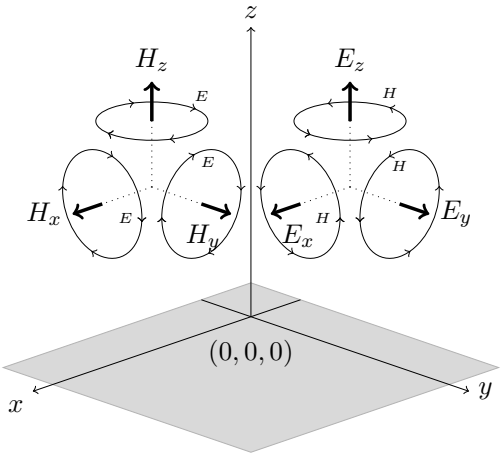


Figure 3.12: 3 Direction EM Field Creation and Propagation

Ampere's Law This is a generalisation of Ampere's Law, which states that the integral of the magnetic field (B) around a conceptual contour must equal the current within the contour (Figure 3.12). This is seen in Equation 3.17, which also shows that a time varying electric field induces a magnetic field. This is closely related to Equation 3.15.

$$\oint_c \vec{H} \cdot dl = \int_s J \cdot ds \; + \; \int_s \frac{\partial \vec{D}}{\partial t} \cdot ds \qquad \rightarrow \qquad \nabla \times \vec{H} = J \; + \; \frac{\partial \vec{D}}{\partial t} \tag{3.17}$$

Magnetic Dipole Law This equation is derived from the fact that magnets (as far as science currently understands) can only exist in dipole pairs, with a positive and negative pole. This is in contrast to electric charges that can either be positive or negative. Given this fact, it makes intuitive sense that the magnetic field lines that leave one magnetic pole, must connect to the other. Therefore any conceptual surface that surrounds a magnetic dipole, must have an equal number of magnetic field lines leaving and entering the surface. This is mathematically described by Equation 3.18.

$$\oint_s \vec{B} \cdot ds = 0 \qquad \rightarrow \qquad \nabla \cdot \vec{B} = 0 \tag{3.18}$$

Other parameters of interest when investigating EM field propagation are the Permeability, Permittivity, and Conductivity of the propagating medium.

Permeability The permeability value (μ) of a material describes the ability of that material to support and maintain a magnetic field. The permeability of a material relates the magnetic field intensity ($\vec{B}$) to the magnetic flux density ($\vec{H}$) through the relationship shown in Equation 3.19.

$$\vec{H} = \frac{1}{\mu}.\vec{B} \tag{3.19}$$

The permeability of free space (μ_0) is by definition $4\pi \times 10^{-7}\,\mathrm{H\,m^{-1}}$. The relative permeability (μ_r) of a material is described by Equation 3.20, which is the ratio of the materials permeability to the permeability of free space.

$$\mu_r = \frac{\mu}{\mu_0} \tag{3.20}$$

Permittivity The permittivity value (ϵ) of a material describes the resistance (ability) of the material to support electric fields. The electric field intensity ($\vec{E}$) and electric flux density ($\vec{D}$) are related by the permittivity as seen in Equation 3.21.

$$\vec{D} = \epsilon.\vec{E} \tag{3.21}$$

The permittivity of free space (ϵ_0) is derived as $8.854\,187\,817\,6 \times 10^{-12}\,\mathrm{F\,m^{-1}}$. The relative permittivity (ϵ_r) of a material is described by Equation 3.22.

$$\epsilon_r = \frac{\epsilon}{\epsilon_0} \tag{3.22}$$

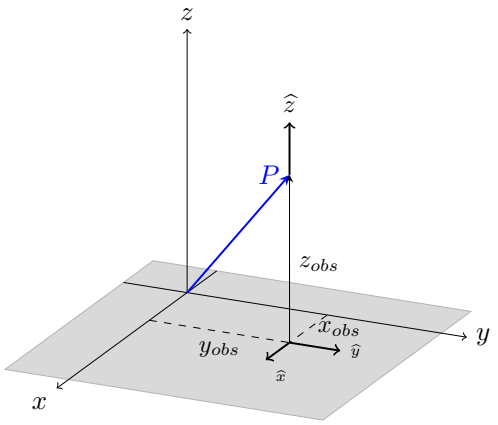
Conductivity The conductivity value (σ) of a material, is the inverse of the materials electrical resistivity. A high resistivity results in a low conductivity, and a low resistivity results in a high conductivity. A materials current density (J) and electric field (E) are related to the materials conductivity by Equation 3.23. From this equation, if an electrical conductor has a current flow, then a line segment would have a high current density. In addition to this, a conductor would have a low resistivity, and therefore a high conductivity. Therefore from Equation 3.23, the resulting electric field would be very low. This is to be expected, as electrical conductors should have an even charge distribution, and therefore no difference in charge to establish an electric field. The connection between Ohm's Law and Equation 3.23 should be easy to make.

$$\sigma = \frac{J}{E} \qquad \rightarrow \qquad E = \frac{J}{\sigma} \tag{3.23}$$

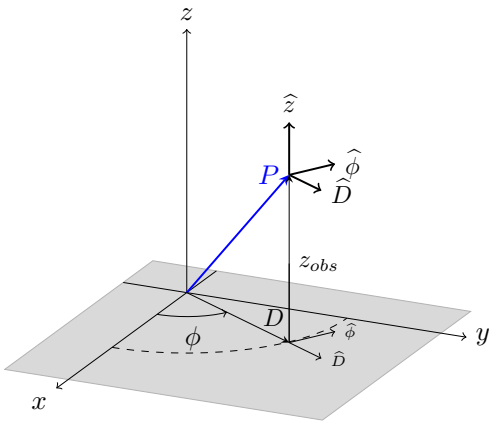
3.5 Curvilinear Co-ordinate Systems

Section 3.4 has outlined the basic theory of Maxwell's Equations, as well as the relevant parameters. As seen in Equations 3.15 to 3.18, Maxwell's equations describe vector fields. There are three different co-ordinate systems that describe vector fields in a 3D plane: Cartesian Co-ordinates, Cylindrical Co-ordinates and Spherical Co-ordinates. These are illustrated in Figures 3.13a to 3.13c. Unit vectors are identified with "$\hat{}$". By convention the ground Distance to observation point "P" in the cylindrical plane is represented by ρ, however in this text the variable "D" is used.

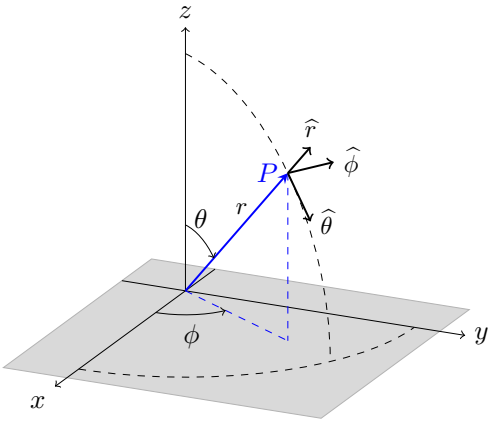
Equations 3.24a to 3.24c show a vector field "F" described by the three different co-ordinate systems of Figures 3.13a to 3.13c respectively.



(a) 3D Cartesian Geometry



(b) 3D Cylindrical Geometry



(c) 3D Spherical Geometry

Figure 3.13: Cartesian, Cylindrical and Spherical Geometry

$$\vec{F} = F_x(x,y,z)\ \widehat{x} + F_y(x,y,z)\ \widehat{y} + F_z(x,y,z)\ \widehat{z} \tag{3.24a}$$

$$\vec{F} = F_D(D,\phi,z)\ \widehat{D} + F_\phi(D,\phi,z)\ \widehat{\phi} + F_z(D,\phi,z)\ \widehat{z} \tag{3.24b}$$

$$\vec{F} = F_r(r,\theta,\phi)\ \widehat{r} + F_\theta(r,\theta,\phi)\ \widehat{\theta} + F_\phi(r,\theta,\phi)\ \widehat{\phi} \tag{3.24c}$$

Equations 3.25a to 3.25c show the nabla operator for the three co-ordinate systems.

$$\nabla = \frac{\partial}{\partial x}\ \widehat{x} + \frac{\partial}{\partial y}\ \widehat{y} + \frac{\partial}{\partial z}\ \widehat{z} \tag{3.25a}$$

$$\nabla = \frac{\partial}{\partial D}\ \widehat{D} + \frac{1}{D}\frac{\partial}{\partial \phi}\ \widehat{\phi} + \frac{\partial}{\partial z}\ \widehat{z} \tag{3.25b}$$

$$\nabla = \frac{\partial}{\partial r}\ \widehat{r} + \frac{1}{r}\frac{\partial}{\partial \theta}\ \widehat{\theta} + \frac{1}{r.\sin(\theta)}\frac{\partial}{\partial \phi}\ \widehat{\phi} \tag{3.25c}$$

Equations 3.26a to 3.26c show the divergence (Dot product) of vector field "F" in the different co-ordinate systems.

$$\nabla \cdot F = \frac{\partial}{\partial x} F_x(x,y,z) + \frac{\partial}{\partial y} F_y(x,y,z) + \frac{\partial}{\partial z} F_z(x,y,z) \tag{3.26a}$$

$$\nabla \cdot F = \frac{1}{D}\frac{\partial}{\partial D} D.F_D(D,\phi,z) + \frac{1}{D}\frac{\partial}{\partial \phi} F_\phi(D,\phi,z) + \frac{\partial}{\partial z} F_z(D,\phi,z) \tag{3.26b}$$

$$\begin{aligned}\nabla \cdot F = &\frac{1}{r^2}\frac{\partial}{\partial r} r^2.F_r(r,\theta,\phi) + \frac{1}{r.\sin\theta}\frac{\partial}{\partial \theta}\sin\theta.F_\theta(r,\theta,\phi) \\ &+ \frac{1}{r.\sin\theta}\frac{\partial}{\partial \phi} F_\phi(r,\theta,\phi)\end{aligned} \tag{3.26c}$$

Equations 3.27a to 3.27c show the gradient (Cross product) of vector field "F" in the different co-ordinate systems.

$$\nabla \times F = \left[\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right] \widehat{x} + \left[\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right] \widehat{y} + \left[\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right] \widehat{z} \tag{3.27a}$$

$$\nabla \times F = \frac{1}{D}\left[\frac{\partial F_z}{\partial \phi} - \frac{\partial F_\phi}{\partial z}\right] \widehat{D} + \left[\frac{\partial F_D}{\partial z} - \frac{\partial F_z}{\partial D}\right] \widehat{\phi} + \frac{1}{D}\left[\frac{\partial F_\phi}{\partial D} - \frac{\partial F_D}{\partial \phi}\right] \widehat{z} \tag{3.27b}$$

$$\begin{aligned}\nabla \times F = &\frac{1}{r.\sin\theta}\left[\frac{\partial (F_\phi.\sin\theta)}{\partial \theta} - \frac{\partial F_\theta}{\partial \phi}\right] \widehat{r} + \frac{1}{r}\left[\frac{1}{\sin\theta}\frac{\partial F_r}{\partial \phi} - \frac{\partial (F_\phi.r)}{\partial r}\right] \widehat{\theta} \\ &+ \frac{1}{r}\left[\frac{\partial (F_\theta.r)}{\partial r} - \frac{\partial F_r}{\partial \theta}\right] \widehat{\phi}\end{aligned} \tag{3.27c}$$

3.6 Chapter Summary

This chapter serves as a background and comprehensive theory for new and experienced lightning researchers. A discussion of the physics and terminology associated with a lightning flash and a lightning stroke has been presented. A section detailing the current impulse (or CBC) models used, as well as a detailed section on return stroke models has also been provided. These two sections make up the EM source model identified in Figure 2.1. From these sections it is recommended that the Heidler Function (Equation 3.2) or the Terespolsky Function (Equation 3.3) be used for the CBC model in conjunction with the MTLE RS model (Equation 3.14). A popular current impulse model (Equation 3.7) has also been presented with all the variables for both the Heidler and Terespolsky Functions.

A section detailing basic electromagnetic theory is included. Maxwell's equations, and associated medium properties are presented in detail. The relevance of these equations are discussed, and explained with figures. A comprehensive review of the three available co-ordinate systems is presented with all the required equations for EM field manipulations presented in later chapters.

The next chapter presents the Finite Antenna Method, with an emphasis on the method theory, equations and implementation considerations. Most importantly the chapter will present examples, and discuss the method strengths.

Chapter 4

Finite Antenna Method

This chapter presents the background and basic theory of the Finite Antenna method, but does not attempt to derive the governing EM field equations. The physical layout of the model, and the related EM equations are presented and discussed in a simple manner. These equations are then applied to a number of test cases to demonstrate the fields that result from a lightning stroke.

4.1 Overview

In the field of lightning electromagnetics this method was first proposed by Uman *et al* [38], where the lightning RS channel is modelled as a finite linear antenna. This work has been improved, and is described in greater detail in [50]. There are three equivalent approaches to evaluating EM fields with this method [50], however the most appropriate for this project is the Lorentz condition (dipole technique).

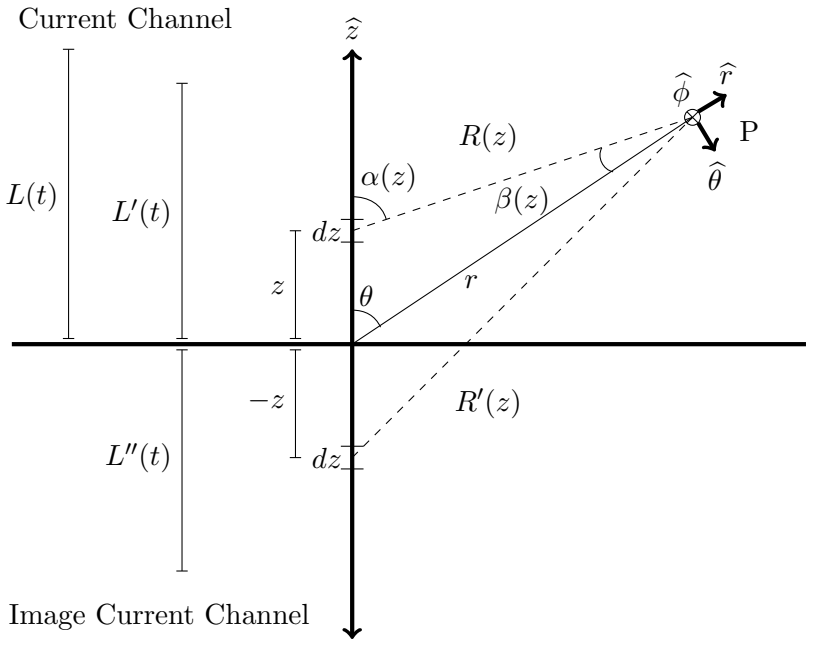
In this method the lightning channel is directed along the $\widehat{z}$ axis, which is then divided into small dz elements. Each dz element is then modelled as an electric dipole. The electric dipole model requires that the dz element length is far less than the distance to the observation point, and that the current in the element is constant across its length [50]. To ensure these assumptions are satisfied, the equation for vector potential at the observation point integrates the channel current over the channel length. The equation for vector potential is then used in conjunction with the Lorentz condition to evaluate the EM fields at the observation point. The Lorentz condition adds an additional equation component that considers the integral of the channel current over time.

For a high level overview, this methods can be thought of as integrating the channel current over time, and over the channel length in order to evaluate the effect that each channel length element has at the observation point.

4.2 Model Development

The geometry describing the EM fields in this section are in Spherical co-ordinates, as seen in Figure 3.13c. From this figure, it is clear that the system is symmetrical around the lightning channel ($\widehat{z}$ axis). Therefore if the distance to the point P is kept constant (r), as well as the angle θ, then the angle ϕ will not affect the EM fields. Therefore the problem can be described in 2D, as seen in Figure 4.1.

This figure shows the current channel above ground, as well as the image channel below ground. The image channel is caused by considering ground as an ideal conductor (resistance = $0\,\Omega$). Therefore the horizontal electric field at ground must be zero, or the



Where		
$L(t)$	=	Maximum Channel Height
$L'(t)$	=	Maximum height "seen" at observation point
z	=	Variable height along the channel
dz	=	Infinitesimal length of height at z
r	=	Length from channel origin to observation point
$R(z)$	=	Length from channel at height z to observation point
θ	=	Angle between vertical channel and r
$\widehat{\theta}$,$\widehat{\phi}$,$\widehat{r}$	=	Unit vectors for spherical co-ordinate system
$\widehat{z}$	=	Unit vector for channel height
$L''(t)$	=	Maximum height of channel image "seen" at observation point
$R'(z)$	=	Length from channel at height $-z$ to observation point
$\alpha(z)$,$\beta(z)$	=	Angles with a height dependence

Figure 4.1: Finite Antenna EM field above ground geometry

field will induce a voltage ($E = V/d$) on the ground and Ohm's Law will no longer hold true [50]. To compensate for this, an image channel is included on the model, where the current in the image channel is equal in magnitude, but flowing in the opposite direction to the current channel. This ensures that the tangential electric fields on the ground plane are zero.

In this figure $L(t)$ represents the height of the lightning channel current in time. The channel above this point has no current, and therefore does not need to be considered in further calculations. However it takes time for an EM field to travel from a point on the line to the observation point ($\frac{R(z)}{c}$). Therefore with reference to the observation point, fields appear to come from a lower point on the channel ($L'(t)$). Stated differently, at the time when fields from the current at $L'(t)$ reach the observation point, the true current height would be at $L(t)$. This is known as the retarded time effect, and is accounted for in Equation 4.1. For the image channel $L''(t)$ use Equation 4.1, but substitute θ with $\pi - \theta$ [50].

$$L'(t) = \frac{r}{1 - (\frac{{v_c}^2}{c^2})} \left(-\frac{{v_c}^2}{c^2}\cos\theta + \frac{v_c t}{r} - \frac{v_c}{c}\sqrt{1 - \frac{{v_c}^2}{c^2} + \frac{{v_c}^2 t^2}{r^2} + \frac{{v_c}^2}{c^2}\cos^2\theta - \frac{2 v_c t}{r}\cos\theta} \right) \quad (4.1)$$

Where

c	=	Speed of light
v_c	=	Velocity of the return stroke.

What follows is the analytical expressions for calculating the electric and magnetic fields at the observation point P that are radiated from the lightning channel ($\widehat{z}$ axis). Equations 4.2 and 4.3 describe the electric fields directed along the $\widehat{r}$ and $\widehat{\theta}$ directions. Equation 4.4 describes the magnetic field directed along the $\widehat{\phi}$ unit vector ("into the page"). The full derivation of these equations are presented by Thottappillil in [50]. From these equations it is clear that the fields depend on the geometric model components, and the return stroke current model ($i_{RS}(z,t)$).

$$\vec{E}(r,\theta,t)\,\widehat{r} = \vec{E}_r(r,\theta,t) =$$

$$-\frac{1}{4\pi\epsilon_0}\int_0^{L'(t)} \frac{\cos\theta - 3\cos\alpha(z)\cos\beta(z)}{R^3(z)} \left\{ \int_{t_b}^{t} i_{RS}(0, \tau - \frac{R(z)}{c} - \frac{z}{v_c}) d\tau \right\} dz\ \widehat{r} \quad (4.2a)$$

$$-\frac{1}{4\pi\epsilon_0}\int_0^{L''(t)} \frac{\cos\theta - 3\cos\alpha(-z)\cos\beta(-z)}{R^3(-z)} \left\{ \int_{t_b}^{t} i_{RS}(0, \tau - \frac{R(-z)}{c} - \frac{z}{v_c}) d\tau \right\} dz\ \widehat{r} \quad (4.2b)$$

$$-\frac{1}{4\pi\epsilon_0}\int_0^{L'(t)} \frac{\cos\theta - 3\cos\alpha(z)\cos\beta(z)}{cR^2(z)} \left\{ i_{RS}(0, t - \frac{R(z)}{c} - \frac{z}{v_c}) \right\} dz\ \widehat{r} \quad (4.2c)$$

$$-\frac{1}{4\pi\epsilon_0}\int_0^{L''(t)} \frac{\cos\theta - 3\cos\alpha(-z)\cos\beta(-z)}{cR^2(-z)} \left\{ i_{RS}(0, t - \frac{R(-z)}{c} - \frac{z}{v_c}) \right\} dz\ \widehat{r} \quad (4.2d)$$

$$-\frac{1}{4\pi\epsilon_0}\int_0^{L'(t)} \frac{\cos\theta - \cos\alpha(z)\cos\beta(z)}{c^2 R(z)} \left\{ \frac{\partial i_{RS}(0, t - \frac{R(z)}{c} - \frac{z}{v_c})}{\partial t} \right\} dz\ \widehat{r} \quad (4.2e)$$

$$-\frac{1}{4\pi\epsilon_0}\int_0^{L''(t)} \frac{\cos\theta - \cos\alpha(-z)\cos\beta(-z)}{c^2 R(-z)} \left\{ \frac{\partial i_{RS}(0, t - \frac{R(-z)}{c} - \frac{z}{v_c})}{\partial t} \right\} dz\ \widehat{r} \quad (4.2f)$$

$$\vec{E}(r,\theta,t)\ \widehat{\theta} = \vec{E}_\theta(r,\theta,t) =$$

$$+\frac{1}{4\pi\epsilon_0}\int_0^{L'(t)} \frac{\sin\theta + 3\cos\alpha(z)\sin\beta(z)}{R^3(z)} \left\{\int_{t_b}^{t} i_{RS}(0,\tau - \frac{R(z)}{c} - \frac{z}{v_c})d\tau\right\} dz\ \widehat{\theta} \quad (4.3a)$$

$$+\frac{1}{4\pi\epsilon_0}\int_0^{L''(t)} \frac{\sin\theta + 3\cos\alpha(-z)\sin\beta(-z)}{R^3(-z)} \left\{\int_{t_b}^{t} i_{RS}(0,\tau - \frac{R(-z)}{c} - \frac{z}{v_c})d\tau\right\} dz\ \widehat{\theta} \quad (4.3b)$$

$$+\frac{1}{4\pi\epsilon_0}\int_0^{L'(t)} \frac{\sin\theta + 3\cos\alpha(z)\sin\beta(z)}{cR^2(z)} \left\{i_{RS}(0,\tau - \frac{R(z)}{c} - \frac{z}{v_c})\right\} dz\ \widehat{\theta} \quad (4.3c)$$

$$+\frac{1}{4\pi\epsilon_0}\int_0^{L''(t)} \frac{\sin\theta + 3\cos\alpha(-z)\sin\beta(-z)}{cR^2(-z)} \left\{i_{RS}(0,\tau - \frac{R(-z)}{c} - \frac{z}{v_c})\right\} dz\ \widehat{\theta} \quad (4.3d)$$

$$+\frac{1}{4\pi\epsilon_0}\int_0^{L'(t)} \frac{\sin\theta + \cos\alpha(z)\sin\beta(z)}{c^2R(z)} \left\{\frac{\partial i_{RS}(0,\tau - \frac{R(z)}{c} - \frac{z}{v_c})}{\partial t}\right\} dz\ \widehat{\theta} \quad (4.3e)$$

$$+\frac{1}{4\pi\epsilon_0}\int_0^{L''(t)} \frac{\sin\theta + \cos\alpha(-z)\sin\beta(-z)}{c^2R(-z)} \left\{\frac{\partial i_{RS}(0,\tau - \frac{R(-z)}{c} - \frac{z}{v_c})}{\partial t}\right\} dz\ \widehat{\theta} \quad (4.3f)$$

$$\vec{B}(r,\theta,t)\ \widehat{\phi} = \vec{B}_\phi(r,\theta,t) =$$

$$+\frac{1}{4\pi\epsilon_0 c^2}\int_0^{L'(t)} \frac{\sin\alpha(z)}{R^2(z)} \left\{i_{RS}(0,t - \frac{R(z)}{c} - \frac{z}{v_c})\right\} dz\ \widehat{\phi} \quad (4.4a)$$

$$+\frac{1}{4\pi\epsilon_0 c^2}\int_0^{L''(t)} \frac{\sin\alpha(-z)}{R^2(-z)} \left\{i_{RS}(0,t - \frac{R(-z)}{c} - \frac{z}{v_c})\right\} dz\ \widehat{\phi} \quad (4.4b)$$

$$+\frac{1}{4\pi\epsilon_0 c^2}\int_0^{L'(t)} \frac{\sin\alpha(z)}{cR(z)} \left\{\frac{\partial i_{RS}(0,t - \frac{R(z)}{c} - \frac{z}{v_c})}{\partial t}\right\} dz\ \widehat{\phi} \quad (4.4c)$$

$$+\frac{1}{4\pi\epsilon_0 c^2}\int_0^{L''(t)} \frac{\sin\alpha(-z)}{cR(-z)} \left\{\frac{\partial i_{RS}(0,t - \frac{R(-z)}{c} - \frac{z}{v_c})}{\partial t}\right\} dz\ \widehat{\phi} \quad (4.4d)$$

Where:

$$\cos\ \alpha(z) = \frac{-(z - r\ \cos\theta)}{R(z)} \quad (4.5)$$

$$\cos\ \beta(z) = \frac{(r - z\cos\theta)}{R(z)} \quad (4.6)$$

$$\sin\ \alpha(z) = \frac{r\ \sin\theta}{R(z)} \quad (4.7)$$

$$\sin\ \beta(z) = \frac{z\sin\theta}{R(z)} \quad (4.8)$$

$$t_b = +\frac{R(z)}{c} + \frac{z}{v_c} \quad (4.9)$$

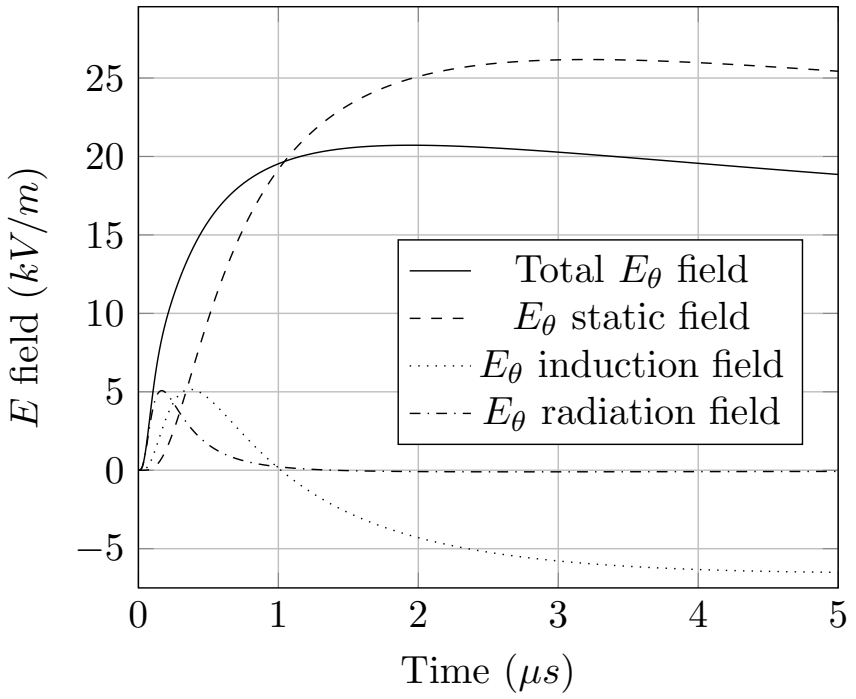


Figure 4.2: Finite Antenna - E_θ Field Components - r = 50 m, $\theta = \pi/2$

Equations 4.5 to 4.8 show the angle variables used in the EM equations. Equation 4.9 shows the lower limit of the time integral. This is the earliest starting time, before which there would be no current in the lightning channel. This is also considered in the RS models with the unit step function.

These equations are composed of unique terms, and each of these terms (field components) contribute to the total field. Traditionally the total field is said to consist of "Electrostatic" (or "Static"), "Induction" and "Radiation" fields [38, 50, 51]. The terms with the time integral of the RS current are referred to as the "Static" fields (Equations 4.2a, 4.2b, 4.3a, 4.3b), and have a $1/R^3$ distance relationship (near field). The terms with the straight RS current are referred to as the "Induction" fields (Equations 4.4a, 4.4b, 4.2c, 4.2d, 4.3c, 4.3d) and have a $1/R^2$ distance relationship. The terms with the time differential of the RS current are referred to as the "Radiation" fields (Equations 4.4c, 4.4d, 4.2e, 4.2f, 4.3e, 4.3f) and have a $1/R$ distance relationship (far field). These field components are not unique, and may have different values when derived through other means [50, 52].

Figure 4.2 shows each of the field component waveshapes as well as the total field. The figure used Equation 3.7 for the CBC model, with the RS model in Equation 3.12 (lossless transmission line). The observation point was located on the ground plane ($\theta = \pi/2$) at a radial distance of 50 m. This example is comparable to the work in [50], which used Equation 3.6 (Heidler Function) for the CBC model.

All the variables in Equations 4.2 to 4.4 have been discussed, and the EM fields can be calculated by choosing a CBC model, a RS model, v_c value, and a observation point in the plane. However inspecting the electric field Equations 4.2 and 4.3, it becomes

clear that solving these equations is not trivial. This is specifically related to the Static field equations with the integral of the RS model. Attempting to numerically solve the integral is difficult due to the fact that the result needs to be a function of z [53]. Therefore there are two options available to simplify the evaluation.

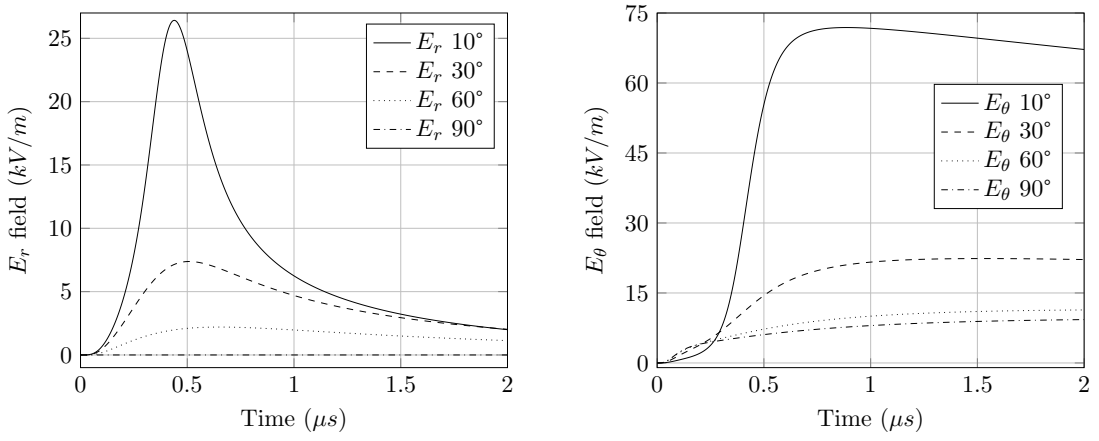
The first is to move the observation point far from the lightning channel. In doing so the $R(z)$ (which is related to r) becomes larger, and the $1/R^3(z)$ distance relationship of the Static field attenuates the field value. Typically for distances greater than a few kilometres only the radiation field components are significant, and the other components can be neglected. However this simplification is not applicable to this work, which requires the distance to be within a range of 50 m to 500 m. And as seen in Figure 4.2, the static field component cannot be neglected within the required range.

The second option is to simplify the time integral. If a Heidler function is used as the CBC model, then the integral cannot be analytically integrated [53]. However this is not a problem with the Terespolsky function. This function was first introduced in 2014 [30], and before this date there was no current impulse model available that had the waveshape of the Heidler function, as well as having an analytical integral solution.

4.3 Model Simulations

This section demonstrates the simulated resulting fields when the above theory is applied to theoretical lightning events. Figures 4.3a and 4.3b show the E_r and E_θ fields located at 100 m from the channel base, at different angles (θ) off the vertical lightning channel. Equation 3.7 is used for the CBC model (Terespolsky version of the popular current impulse model), and the lossless transmission line model is used for the RS model (Equation 3.12). The ground is assumed to be a perfect electrical conductor (PEC), and the RS current velocity is chosen to be $150 \times 10^6\,\mathrm{m\,s^{-1}}$ (Table 3.3).

Figure 4.3a shows $E_r = 0$ when $\theta = 90°$. This conforms to the requirement for the PEC ground plane. This figure also shows that the field value increases as θ decreases, which



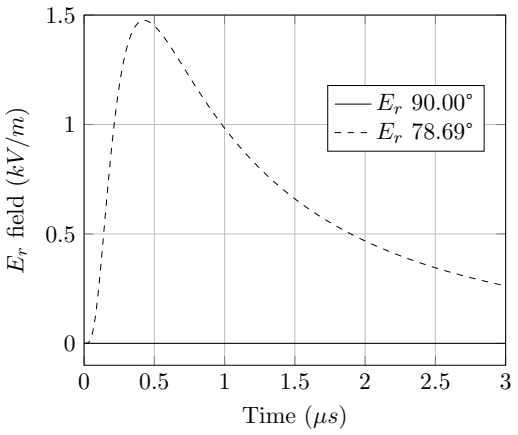
(a) E_r (b) E_θ

Figure 4.3: Finite Antenna - E Fields - r= 100 m - Variable Angle

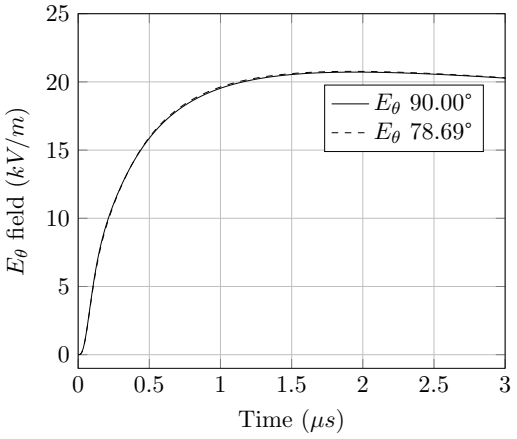
makes intuitive sense as the observation point gets closer to the lightning channel. It is also interesting to compare the field waveshapes in Figure 4.3a, to the current impulse waveshape in Figure 3.8. The current impulse peaks within 0.5 µs, but has a low attenuation time lasting several µs. The E_r field (with $\theta = 10°$) peaks within 0.5 µs, but attenuates rapidly to 10% within 2 µs.

A similar analysis of the fields in Figure 4.3b shows that the peaks all occur after 0.5 µs. Another observation is that at larger angles of θ (further away from the channel, and closer to the PEC ground plane) the fields tend to rise sooner, and peak later. Both Figures 4.3a and 4.3b can be compared to the work in [50].

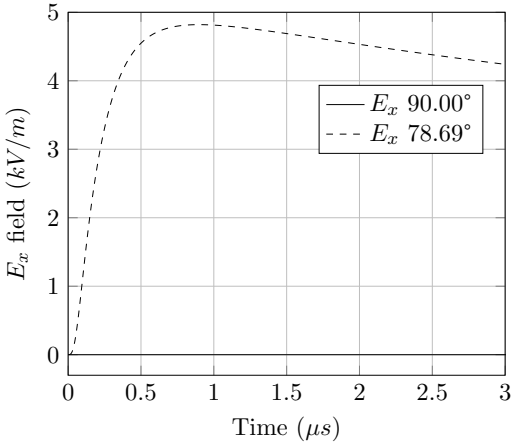
Figures 4.4a to 4.4d show the electric field components (in both spherical and Cartesian co-ordinates) of two observation points situated 50 m away from the lightning channel base along the $\widehat{x}$ axis. The first observation point is 50 m away from the lightning channel (x_{Obs}), and is situated 10 m off the ground (y_{Obs}). In spherical co-ordinates this would be at r = 50.99 m, $\theta = 78.69°$. The second observation point is situated on the



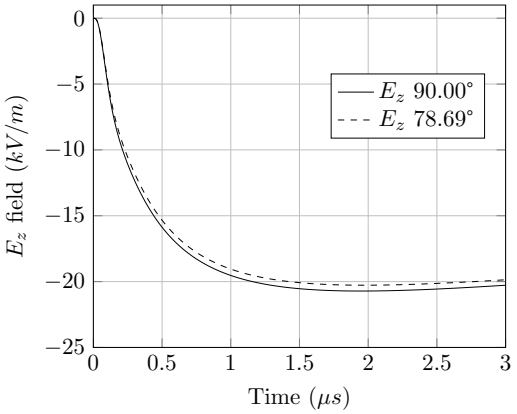
(a) E_r Variable Angle - $x_{Obs} = 50$ m



(b) E_θ Variable Angle - $x_{Obs} = 50$ m



(c) E_x Variable Angle - $x_{Obs} = 50$ m



(d) E_z Variable Angle - $x_{Obs} = 50$ m

Figure 4.4: Finite Antenna - E Fields in Cartesian and Spherical Co-Ordinates

ground plane, and 50 m from the lightning channel (r = 50 , $\theta = 90°$). The CBC and RS models are the same as before.

In terms of LIOVs the first observation point is the worst case position of an overhead distribution line to a lightning stroke channel. Distances closer than this have a higher probability of becoming a direct strike [5], and further than this the fields would attenuate. The second observation point illustrates a potential simplification to the problem. This point is the same distance from the RS channel, but located on the ground plane. Under this situation the field equations are simplified, and the E_r (E_x) field is zero due to the PEC ground.

Figures 4.4a and 4.4b show the fields in spherical co-ordinates. The E_θ fields in Figure 4.4b show a negligible difference (0.5%) between the two observation points. However if the assumption is made that that the fields at both observation points are the same, then the entire E_r field in Figure 4.4a would be lost. This is clearer in Figures 4.4c and 4.4d which show the fields in Cartesian co-ordinates. The co-ordinate shift is a simple transform where E_x and E_z are composed of the vector components E_r and E_θ. The difference between the E_z fields is now more recognisable. More importantly the amplitude of the E_x field is larger than anticipated (when compared with the E_r field). This may not have been an intuitive conclusion when reviewing the fields in spherical co-ordinates. The horizontal electric fields have the most effect on LIOVs, which is why the E_x field value is highlighted. This example shows that the ground point simplification method is not acceptable for LIOV based simulations.

4.4 Chapter Summary

This chapter has presented the basic theory and the required model equations for the Finite Antenna Method. Insights into the practical implementation of the model (including simplifications) have been discussed and demonstrated in example cases. An important outcome of the examples showed that fields located 10 m above the ground plane (typical height of a distribution line) cannot be approximated by the fields on the ground plane. This assumption is generally only applicable to distances further from the lightning channel.

This is a simple model to implement, but suffers from a lack in model variability, such as including a real ground plane, or complex physical environments. The next chapter will introduce the FDTD method in three dimensions, which allows for these advanced simulations.

Chapter 5

3D FDTD Method

This chapter presents the FDTD method for three dimensional (3D) lightning based electromagnetic simulations. The FDTD technique forms a complete field of engineering, and demands the attention of full books. The purpose of this chapter is to present enough information on the method in order to apply it to lightning based electromagnetic simulations, and therefore does not aim to present all aspects of the FDTD method.

A brief overview of the method is discussed, followed by the basic mathematical derivation of the EM update equations. The FDTD EM equations are evaluated at discrete time steps, and are known as the "update equations". These equations are then linked to the "Yee Cell", a fundamental building block used to construct the simulation space (physical environment of interest). The requirements for numerical stability of the simulation are briefly discussed, as well as one possible method for dealing with the boundary conditions (2nd Order Mur) of the simulation space. The process of adding source elements and objects to the simulation space is presented along with options for advanced applications. There is also a section to discuss the practical computational considerations of the method, which is not often recognised or discussed in typical FDTD literature. The final section of this chapter demonstrates a number of example situations in order to show field waveforms, as well as the effects of different simulation space parameters.

5.1 Overview

The Finite-Difference Time-Domain (FDTD) method is a specific version of a Finite Difference Equation (FDE) [54]. FDE's are a mathematical approach used to discretize Partial Differential Equations (PDE's), which are common in nature and can be seen in Maxwell's Equations, thermal equations, and others [48]. The FDTD method, also known as the Yee Algorithm, is specifically focussed on evaluating Maxwell's equations in the time domain [48]. This method is a generic approach, and can be used for most EM simulations. The primary advantage of the method is due to its time domain element, which is well suited to evaluating the transient EM source created by lightning.

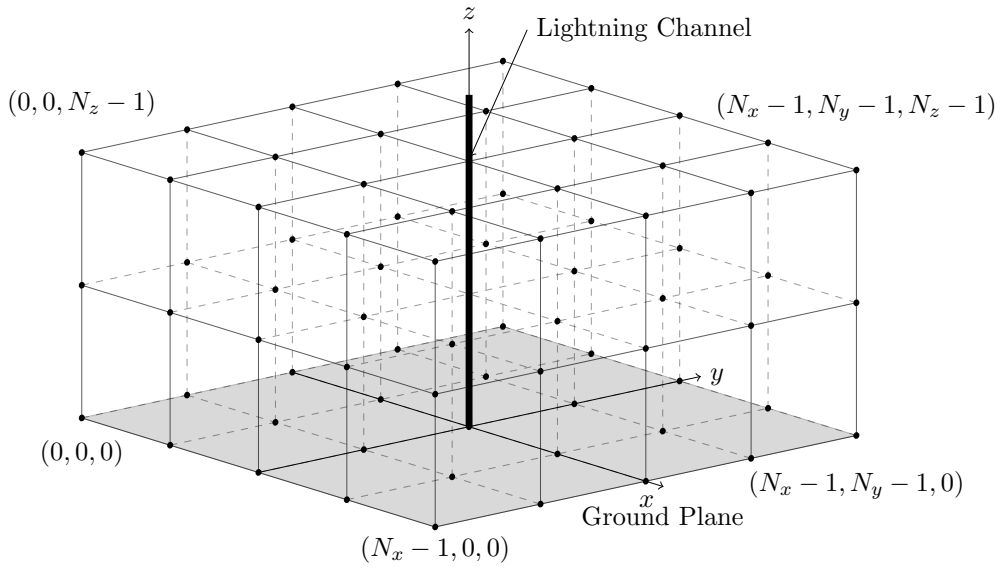


Figure 5.1: 3D Lightning simulation space divided into cells

The technical description of the FDTD method is: A means of representing continuous derivative equations (such as Maxwell's), by second order accurate, two-point centred difference FDE's, that use the "leapfrog" method to update the fields in time [48]. This complicated description is basically directed at the discretization method used for the spatial and time derivatives of Maxwell's equations. For an in-depth discussion on alternative FDE's with higher orders (relating to error), and alternative update equations, the reader is encouraged to read [48]. However for the purposes of this work, the standard FDTD method is sufficient, and better suited to the scope.

The FDTD method is simply a mathematical approach used to discretize Maxwell's equations. This is done by breaking down the propagation medium into smaller areas (cubes), and evaluating the discrete electromagnetic field changes across these cubes (finite difference), at discrete steps in time (time domain). This is illustrated in Figure 5.1, which shows an example simulation environment divided into smaller cubes, where each cube is defined between spatial nodes (black dots) that are located on the corners of each cube [49].

An example of a cube, hereafter referred to as a cell, is shown in Figure 5.2. This is known as the Yee Cell [54]. This figure shows how the electric and magnetic fields are interleaved across the cell with half unit distance, which is a part of the second-order accuracy requirements. The placement of electric fields on integer boundaries (between nodes), and magnetic fields on half integer boundaries (cell sides) is arbitrary, and was chosen so that the electric fields of the cell would align with the boundaries of the simulation space.

The next section will show that the fields in each cell depend on the fields in the cells surrounding it (at a previous step in time). Therefore the FDTD method updates the electric and magnetic fields in every cell of the simulation space, for every step in time; and by doing so the EM fields move through the simulation space. One of the benefits of this method, is that the properties of every cell, such as the medium permittivity,

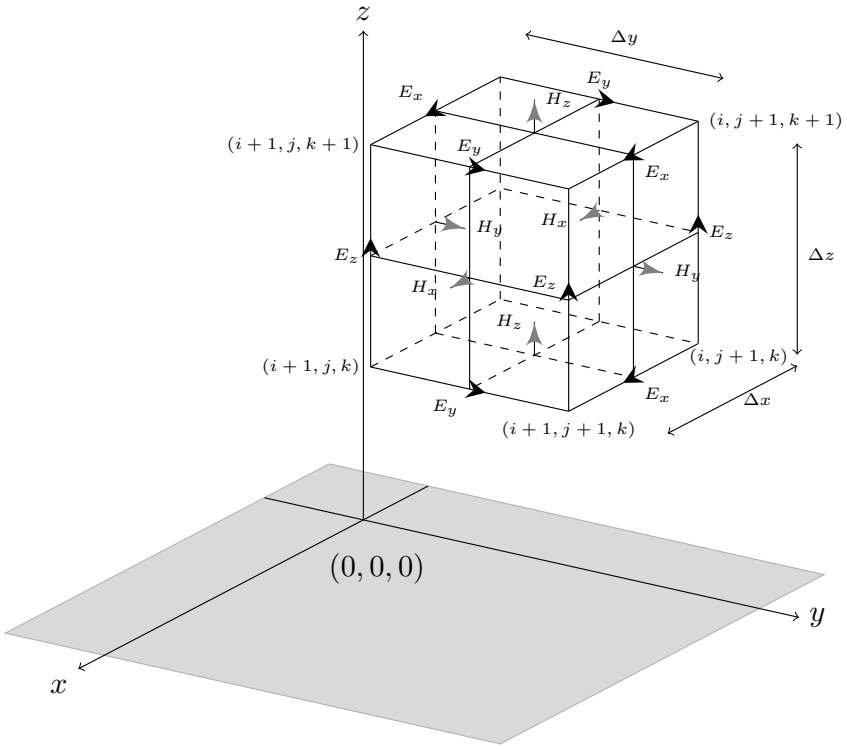


Figure 5.2: 3D Yee Cell - Cartesian Co-ordinates - H Field centred

permeability and conductivity, as well as the boundary conditions, can be individually defined. By changing the properties of these cells it is possible to add objects into the simulation space, and simulate the effects of the objects on the EM fields within the space. In a simple Lightning simulation, one object would be the lightning channel (current source), and another would be the ground plane. In more complex simulations it would be possible to simulate buildings, transmission lines, multilayer grounds, buried conductors and almost any other object of interest.

What follows this section is an explanation of how the discrete equations for the EM fields in a Yee Cell are derived in Cartesian co-ordinates. These equations are discrete and easily applied to numerical techniques in software [9]. Therefore FDTD is part of the Computational Electromagnetics (CEM) family. This method inherently has a number of numerical errors that need to be designed for, and understood when interpreting the results. The two most important are the simulation stability, and the simulation space boundary conditions.

5.2 Model Development

FDTD in the 3D Cartesian space is the most intuitive method for explaining the FDTD method, and allows for complex simulation environments to be constructed. The consequence of this system is that it requires high computer memory for evaluating the three dimensional simulation space.

5.2.1 3D FDTD Equations

Equations 3.15 (Faraday's Law) and 3.17 (Ampere's Law) are the two Maxwell's equations used in deriving FDTD equations. Consider Equation 3.17, which relates the magnetic field change in space to the current density and the electric field change in time. The current density can be written as having two components as seen in Equation 5.1. The $\vec{J_i}$ is an externally imposed current in the cell. This is the source component that adds to the fields in the evaluation space, and without this component the FDTD method would only describe how a pre-existing EM field propagates in the evaluation space [48, 49]. The $\vec{J_c}$ component is conduction current that flows in electrically conducting media (σ^e) when there is an electric field present in the material. This component is the loss term associated with a material.

$$\begin{aligned} \vec{J} &= \vec{J_i} + \vec{J_c} \\ &= \vec{J_i} + \sigma^e \vec{E} \end{aligned} \tag{5.1}$$

These components are reflected in Equation 5.2. Equation 5.3 shows the same methodology applied to the magnetic fields in Faraday's law, where M_i is a source of magnetic fields, and σ^m is the magnetic conductivity of a material (loss term). Both of these terms are theoretical, and hold no real world properties (with our current understanding of electromagnetics). For the remainder of this text these magnetic terms will remain zero, however they are available for EM field manipulation.

$$\nabla \times \vec{H} = \vec{J_i} + \sigma^e \vec{E} + \epsilon \frac{\partial \vec{E}}{\partial t} \tag{5.2}$$

$$\nabla \times \vec{E} = -\vec{M_i} - \sigma^m \vec{H} - \mu \frac{\partial \vec{H}}{\partial t} \tag{5.3}$$

The Permittivity (ϵ), Permeability (μ) and Conductivity (σ^e) in Equations 5.2 and 5.3 can be uniquely defined at all points in space, and therefore have a spatial dependence (x, y and z). These variables describe the properties of the material in which the EM fields propagate, and as such are critical in adding objects to the simulation space. These variables can have complex non-linear dependencies, and even a directional dependence (μ_x, μ_y and μ_z), however this work assumes that ϵ, μ and σ^e are isotropic simple constants in a homogeneous medium.

$$\begin{aligned} \nabla \times \vec{E} &= \begin{bmatrix} \widehat{x} & \widehat{y} & \widehat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ E_x & E_y & E_z \end{bmatrix} = -\mu \frac{\partial \vec{H}}{\partial t} \\ &= \widehat{x} \left(\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} \right) - \widehat{y} \left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} \right) + \widehat{z} \left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} \right) \end{aligned} \tag{5.4}$$

$$\begin{aligned} \nabla \times \vec{H} &= \begin{bmatrix} \widehat{x} & \widehat{y} & \widehat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ H_x & H_y & H_z \end{bmatrix} = \epsilon \frac{\partial \vec{E}}{\partial t} + \sigma^e \vec{E} + \vec{J_i} \\ &= \widehat{x} \left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \right) - \widehat{y} \left(\frac{\partial H_z}{\partial x} - \frac{\partial H_x}{\partial z} \right) + \widehat{z} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \right) \end{aligned} \tag{5.5}$$

Equations 5.4 and 5.5 show the expansion of Equations 5.2 and 5.3 using the nabla operator for Cartesian co-ordinates (Equation 3.25a). By grouping the unit vector components, the following six unique identities can be created.

$$\begin{aligned}
\frac{\partial E_x}{\partial t} &= \frac{1}{\epsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - \sigma_x^e E_x - J_{ix}\right) & \frac{\partial H_x}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right) \\
\frac{\partial E_y}{\partial t} &= \frac{1}{\epsilon}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} - \sigma_y^e E_y - J_{iy}\right) & \frac{\partial H_y}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right) \\
\frac{\partial \mathbf{E_z}}{\partial \mathbf{t}} &= \frac{\mathbf{1}}{\epsilon}\left(\frac{\partial \mathbf{H_y}}{\partial \mathbf{x}} - \frac{\partial \mathbf{H_x}}{\partial \mathbf{y}} - \sigma_\mathbf{z}^\mathbf{e}\mathbf{E_z} - \mathbf{J_{iz}}\right) & \frac{\partial H_z}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right)
\end{aligned} \tag{5.6}$$

To discretize the identities in Equations 5.6, consider for example the identity containing the time derivative of E_z, which is expanded in Equations 5.7. The LHS of Equations 5.7 shows the continuous form, and the RHS shows the discrete form of the E_z example case terms [54]. This discretization process is used throughout the remaining chapters.

$$\epsilon\frac{\partial E_z}{\partial t} = \epsilon_{(i,j,k+\frac{1}{2})}\frac{E_z\Big|_{(i,j,k+\frac{1}{2})}^{n+1} - E_z\Big|_{(i,j,k+\frac{1}{2})}^{n}}{\Delta t} \tag{5.7a}$$

$$\frac{\partial H_y}{\partial x} = \frac{H_y\Big|_{(i+\frac{1}{2},j,k+\frac{1}{2})}^{n+\frac{1}{2}} - H_y\Big|_{(i-\frac{1}{2},j,k+\frac{1}{2})}^{n+\frac{1}{2}}}{\Delta x} \tag{5.7b}$$

$$\frac{\partial H_x}{\partial y} = \frac{H_x\Big|_{(i,j+\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} - H_x\Big|_{(i,j-\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}}}{\Delta y} \tag{5.7c}$$

$$\sigma^e E_z = \sigma^e_{(i,j,k+\frac{1}{2})}E_z^{n+\frac{1}{2}} = \sigma^e_{(i,j,k+\frac{1}{2})}\frac{E_z\Big|_{(i,j,k+\frac{1}{2})}^{n+1} + E_z\Big|_{(i,j,k+\frac{1}{2})}^{n}}{2} \tag{5.7d}$$

$$\vec{J}_{iz} = J_{iz}\Big|_{(i,j,k+\frac{1}{2})}^{n+\frac{1}{2}} \tag{5.7e}$$

The indices for the spatial locations have been taken from the cell in Figure 5.2, which are necessary for the FDTD second-order characteristic. The notation used in Equations 5.7, as well as the rest of this document are summarised in Equation 5.8 [48], where F is an arbitrary field that has a spatial and time dependence.

$$F\Big|_{(i,j,k)}^{n} = F(i,j,k,n) \rightarrow F(i.\Delta x, j.\Delta y, k.\Delta z, n.\Delta t) = F(x,y,z,t) \tag{5.8}$$

The integer indices i, j and k are used to define the spatial location of nodes throughout the evaluation space, where each node is a corner of a Yee Cell. The n integer index defines the discrete steps in the evaluation time. Originally the FDTD equations were used to derive the Yee Cell, but it is equally valuable to use the Yee Cell to understand the FDTD equations. In Figure 5.2 the electric fields of the cell are defined at the spatial locations between the nodes, which is why E_z is located at $(i,j,k+\frac{1}{2})$ for all integer increments of the indices (throughout the evaluation space).

Figure 5.2 also shows that the magnetic field intensity vectors are located on the cell sides. These locations are situated between four nodes, and therefore have two half

integer locations. This is why H_z fields are located at $(i+\frac{1}{2}, j+\frac{1}{2}, k)$ locations throughout the evaluation space. The same is true for the other EM vector fields.

After studying the cell in Figure 5.2, a pattern emerges where the electric fields have the half integer increment added to the index with the same direction as the field ($z \rightarrow k$). Similarly the magnetic fields have the half integer increment added to the indices that do not share the same direction ($z \rightarrow i, k$).

The time indices of Equations 5.7 are not clearly represented in Figure 5.2, and is a function of the differencing schemes used in the FDTD equations [48]. These equations require electric fields to exist at integer time steps (n), and magnetic fields to exist at half integer time steps ($n + \frac{1}{2}$). An example of this is seen by the requirement for the center time difference ($n + \frac{1}{2}$) of Equation 5.7a needing to match the time index of the spatial difference in Equations 5.7b and 5.7c. This time requirement of the difference equations is further emphasised in Equation 5.7d, where the electric field loss term is discretely averaged (not differenced) in order for the unit time step to be maintained.

To simplify the notation used in Equations 5.7, it is noted that the ϵ, σ^e and J components (as well as μ for $H^{n+\frac{1}{2}}$ field equations) are all situated at the same location as the field under review (LHS of Equations 5.7).

Using the above notation and assumptions, the Equation components 5.7a to 5.7e can be combined and simplified. Equations 5.9 shows the step by step process of finding the discrete update equation for E_z at a point on the Yee Cell. Equation 5.9b is known as the FDTD update equation for the E_z field, where the term "Update" refers to the field at the next point in time.

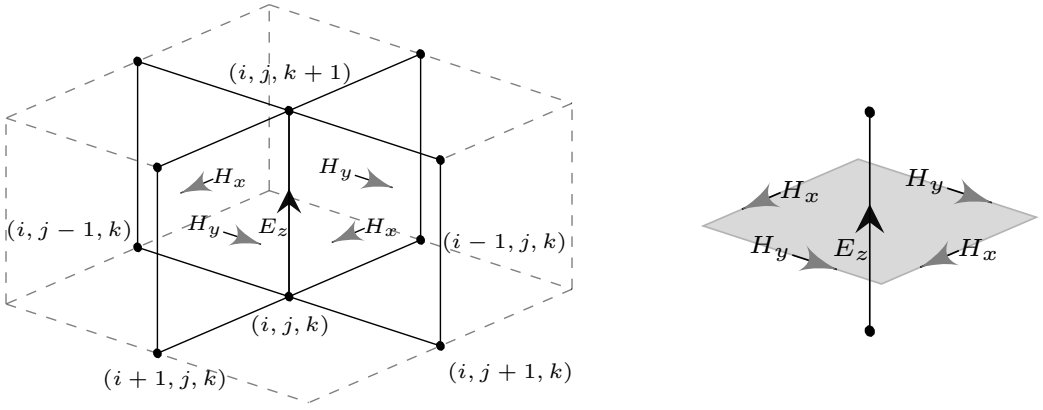
$$\epsilon \left(\frac{E_z\Big|_{(i,j,k+\frac{1}{2})}^{n+1} - E_z\Big|_{(i,j,k+\frac{1}{2})}^{n}}{\Delta t} \right) + \sigma^e \left(\frac{E_z\Big|_{(i,j,k+\frac{1}{2})}^{n+1} + E_z\Big|_{(i,j,k+\frac{1}{2})}^{n}}{2} \right) =$$

$$\left(\frac{2\epsilon + \sigma^e \Delta t}{2.\Delta t} \right) E_z\Big|_{(i,j,k+\frac{1}{2})}^{n+1} - \left(\frac{2\epsilon - \sigma^e \Delta t}{2.\Delta t} \right) E_z\Big|_{(i,j,k+\frac{1}{2})}^{n} = -J_{iz}$$

$$+ \left(\frac{H_y\Big|_{(i+\frac{1}{2},j,k+\frac{1}{2})}^{n+\frac{1}{2}} - H_y\Big|_{(i-\frac{1}{2},j,k+\frac{1}{2})}^{n+\frac{1}{2}}}{\Delta x} - \frac{H_x\Big|_{(i,j+\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} - H_x\Big|_{(i,j-\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}}}{\Delta y} \right) \tag{5.9a}$$

$$E_z\Big|_{(i,j,k+\frac{1}{2})}^{n+1} = \left(\frac{2\epsilon - \sigma^e \Delta t}{2\epsilon + \sigma^e \Delta t} \right) E_z\Big|_{(i,j,k+\frac{1}{2})}^{n} - \left(\frac{2\Delta t}{2\epsilon + \sigma^e \Delta t} \right) J_{iz}$$

$$+ \left(\frac{2\Delta t}{2\epsilon + \sigma^e \Delta t} \right) \left(\frac{H_y\Big|_{(i+\frac{1}{2},j,k+\frac{1}{2})}^{n+\frac{1}{2}} - H_y\Big|_{(i-\frac{1}{2},j,k+\frac{1}{2})}^{n+\frac{1}{2}}}{\Delta x} - \frac{H_x\Big|_{(i,j+\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} - H_x\Big|_{(i,j-\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}}}{\Delta y} \right) \tag{5.9b}$$



(a) E_z field components (b) Ampere's Law

Figure 5.3: 3D FDTD - E_z Yee Cell Field Mathematics

The LHS of Equation 5.9b shows that the E_z value at the updated time step, is equal to the E_z value at its previous point in time (same location), plus the magnetic field values surrounding the E_z location at a half time step, and minus the source current density. This is known as the "leapfrog" update equation, because new values are evaluated by previous time step values, as well as previous half time step values [48].

Figure 5.3 illustrates the components of Equation 5.9b assuming that $J = 0$. Figure 5.3a shows the E_z field located on an edge between four Yee Cells. This figure also shows the magnetic field vectors on the cell walls surrounding the E_z field, which are the primary components of Equation 5.9b. Figure 5.3b shows how the electric field update equation is linked to Ampere's Law. The electric field passing through a surface is related to the magnetic field surrounding the contour of the surface.

The update equations for the remaining electric and magnetic field components are found by applying the same process seen in Equations 5.7 and 5.9 to the Equations in 5.6. The discretized versions of these electric and magnetic fields are shown in Equations 5.10 to 5.15.

$$E_x\Big|^{n+1}_{(i+\frac{1}{2},j,k)} = C_{ee}E_x\Big|^{n}_{(i+\frac{1}{2},j,k)} + C_eJ_{ix} + C_e\left(\frac{H_z\Big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} - H_z\Big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j-\frac{1}{2},k)}}{\Delta y} - \frac{H_y\Big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j,k+\frac{1}{2})} - H_y\Big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j,k-\frac{1}{2})}}{\Delta z}\right) \tag{5.10}$$

$$E_y\Big|^{n+1}_{(i,j+\frac{1}{2},k)} = C_{ee}E_y\Big|^{n}_{(i,j+\frac{1}{2},k)} + C_eJ_{iy} + C_e\left(\frac{H_x\Big|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} - H_x\Big|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k-\frac{1}{2})}}{\Delta z} - \frac{H_z\Big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} - H_z\Big|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2},k)}}{\Delta x}\right) \tag{5.11}$$

$$E_z\Big|^{n+1}_{(i,j,k+\frac{1}{2})} = C_{ee}E_z\Big|^{n}_{(i,j,k+\frac{1}{2})} + C_eJ_{iz} + C_e\left(\frac{H_y\Big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j,k+\frac{1}{2})} - H_y\Big|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j,k+\frac{1}{2})}}{\Delta x} - \frac{H_x\Big|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} - H_x\Big|^{n+\frac{1}{2}}_{(i,j-\frac{1}{2},k+\frac{1}{2})}}{\Delta y}\right) \tag{5.12}$$

$$H_x\Big|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} = H_x\Big|^{n-\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} + C_h\left(\frac{E_y\Big|^{n}_{(i,j+\frac{1}{2},k+1)} - E_y\Big|^{n}_{(i,j+\frac{1}{2},k)}}{\Delta z} - \frac{E_z\Big|^{n}_{(i,j+1,k+\frac{1}{2})} - E_z\Big|^{n}_{(i,j,k+\frac{1}{2})}}{\Delta y}\right) \tag{5.13}$$

$$H_y\Big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j,k+\frac{1}{2})} = H_y\Big|^{n-\frac{1}{2}}_{(i+\frac{1}{2},j,k+\frac{1}{2})} + C_h\left(\frac{E_z\Big|^{n}_{(i+1,j,k+\frac{1}{2})} - E_z\Big|^{n}_{(i,j,k+\frac{1}{2})}}{\Delta x} - \frac{E_x\Big|^{n}_{(i+\frac{1}{2},j,k+1)} - E_x\Big|^{n}_{(i+\frac{1}{2},j,k)}}{\Delta z}\right) \tag{5.14}$$

$$H_z\Big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} = H_z\Big|^{n-\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} + C_h\left(\frac{E_x\Big|^{n}_{(i+\frac{1}{2},j+1,k)} - E_x\Big|^{n}_{(i+\frac{1}{2},j,k)}}{\Delta y} - \frac{E_y\Big|^{n}_{(i+1,j+\frac{1}{2},k)} - E_y\Big|^{n}_{(i,j+\frac{1}{2},k)}}{\Delta x}\right) \tag{5.15}$$

Where:

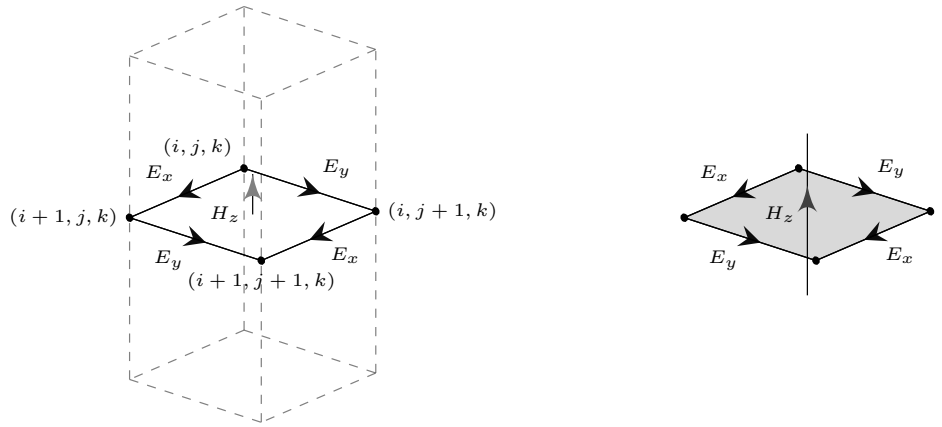
$$C_{ee} = \frac{2\epsilon - \sigma^e\Delta t}{2\epsilon + \sigma^e\Delta t} \tag{5.16a}$$

$$C_e = \frac{2\Delta t}{2\epsilon + \sigma^e\Delta t} \tag{5.16b}$$

$$C_h = \frac{\Delta t}{\mu} \tag{5.16c}$$

These six equations are the foundation for implementing the FDTD algorithm in software, as discussed later in Section 5.2.5. Equations 5.16 are the simulation space constants that define the properties of every Yee Cell in the simulation space. In these equations the ϵ, σ^e and μ components are defined at the location of the update field.

Figure 5.4 illustrates how the H field update equations (5.13 to 5.15) are linked to Faraday's law. Figure 5.4a shows the H_z field defined between two Yee Cells, as compared to four in Figure 5.3a, and requires the electric fields located on the edges of the Yee Cell's face. Figure 5.4b shows Faraday's law, which links the magnetic field lines passing through a surface to the electric fields around the contour of the surface.



(a) H_z field components (b) Faraday's Law

Figure 5.4: 3D FDTD - H_z Yee Cell Field Mathematics

5.2.2 Stability and Dispersion

In electromagnetics there are a number of natural effects that hinder the propagation of EM fields, and often the purpose of simulating EM field propagation is to study these effects. A consequence of FDE's and FDTD methods is that they introduce numerical issues, due to the discretization process, that can be misinterpreted as a natural effect. This process can also cause instability in wave propagation due to a misrepresentation of the natural system.

Some of these numerical issues include dispersion (variation of phase velocity "v_p" with frequency), dissipation (attenuation of fields), anisotropy (variation of phase velocity "v_p" with propagation direction) and instability (uncontrolled growth of a field). Each of these discretization errors exhibit different effects depending on the FDE used.

The FDTD algorithm has a number of benefits due to its two point centred difference scheme coupled with the "leapfrog" time update property. For FDTD all four of these effects are minimised, or resolved by adhering to the Courant-Friedrichs-Lewy (CFL) stability criteria [48]. There are a number of rigorous proofs of the CFL law for FDE's and FDTD, however the most intuitive proof is to consider the physical laws of the environment, such as velocity (Equation 5.17). This equation holds true for the continuous time and space domain, and should equally hold true in the discrete time and space domain of the FDTD environment.

$$\text{Velocity} = \frac{\text{Displacement}}{\text{Time}} \tag{5.17}$$

Equation 5.18 shows Equation 5.17 represented in discrete time and discrete space. The velocity "v_c" represents the true physical constant propagation velocity of the system. The velocity "v_{dis}" represents the effective velocity "seen" in the discrete domain. Ideally these velocities should equate, however this may not be numerically realisable due to limited numerical step values. Without considering FDE's or FDTD, this equation should make practical and physical sense for any objects motion. Equation 5.18 is an

exact solution to the CFL for one dimensional motion in the $\widehat{x}$ direction.

$$v_c \rightarrow \frac{\Delta x}{\Delta t} = v_{dis}; \tag{5.18}$$

From Equation 5.18, it is clear that if these velocities do not match, then there will be a discrepancy between the real system velocity, and the discrete numerical velocity representing the system. Under this situation, if the numerical velocity is greater than the real velocity, then propagating wavefronts moving through a discrete spatial domain will move further during each time step, and overlap (sum) with lagging wavefronts. This results in an unstable system. Therefore the discrete velocity should be kept equal to or less than the constant physical velocity "v_c" of the system. The CFL stability criteria in one dimension is now more accurately defined in Equation 5.19. This allows the selection of the spatial grid to determine the simulation time steps.

$$\Delta t \leq \frac{\Delta x}{v_c} \tag{5.19}$$

Equation 5.20 shows three dimensional equation for the CFL stability criteria in the Cartesian co-ordinate system. If $\Delta x = \Delta y = \Delta z$, then Equation 5.21 is used, where "Dimension" is replaced by the number of dimensions under review. The derivation for these higher order displacement components do not relate to the "corner to corner" distance of a cell, but rather the displacement of the wavefront through the cells [48].

$$\Delta t \leq \frac{1}{v_c\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}}} \tag{5.20}$$

$$\Delta t \leq \frac{\Delta x}{v_c\sqrt{\text{Dimension}}} \tag{5.21}$$

The CFL stability criteria in the above equations is critical in FDTD. If it is not met, even for small discrepancies, then the system becomes unstable. The CFL condition also controls the numerical dispersion, dissipation and anisotropy. To minimize dispersion effects the Δt value should be kept as close to the "magic time step" (optimal time step) as possible. This is achieved by equating the CFL condition, and results in the FDTD algorithm having no numerical dispersion. With regards to dissipation the Δt value may be less than the "magic time step". Other FDE's experience numerical field attenuation due to Δt being set too small, however FDTD remains unaffected.

With regards to anisotropy the CFL condition must also be met, but is mostly affected by the number of Yee cells used to represent the wavelength of the EM source frequency. A recommended minimum is 10 cells per wavelength, which causes a maximum anisotropy error (v_p/c) of 1.5% [48]. However these effects cannot be fully eliminated due to the finite cell size [48].

The CFL stability criteria creates a practical limitation on simulations. Typically a fine spatial resolution is required by advanced simulations. This spatial resolution (cell size), in conjunction with the CFL, force a small time step. This results in not only a high number of Yee cells in the evaluation space, but also a high number of computational steps to evaluate the simulation.

5.2.3 Boundary Conditions

Figures 5.3a and 5.4a show the spatial cell arrangement needed for update Equations 5.12 and 5.15 to be solvable. Similar cell/field requirements exist for the remaining field update equations. Using these figures as a reference it becomes clear that there are problems in solving the electric field, but not the magnetic field, update equations that exist on the simulation space boundaries.

Figure 5.9 shows an example simulation space of Yee Cells. This figure shows that H_y, E_y and E_z fields exist on the "X-max" boundary (Y-Z plane at the maximum X node). Figure 5.3a shows that the E_z fields on the boundary cannot be solved, because the cells containing the $H_z(i_{max}+\frac{1}{2}, *j, *k)$ do not exist. The same is true for all the electric fields that fall on the boundaries of the simulation space. Therefore the standard update equations cannot be used for the electric fields on the boundaries. As seen in Figure 5.4, the same is not true for the magnetic fields that fall on the boundaries (provided the electric fields are known).

If the electric fields on the boundaries are not solved then these fields will remain fixed at $0\,\mathrm{V\,m^{-1}}$. This property makes the boundary resemble a perfect electrical conductor (PEC), as discussed in Chapter 4. In certain circumstances this property is useful, such as trying to model a PEC ground plane. However for most simulations this simply results in the internal EM fields reflecting off the boundaries. One solution is to make the simulation space sufficiently big that reflections do not interfere with locations of interest within the simulation space. However this method requires a large computer memory, and unnecessarily long processing times (Section 5.2.5). This work used Absorbing Boundary Conditions (ABC's) to solve the boundary fields.

An absorbing boundary condition is a mathematical expression that approximates the fields on the boundary by using the fields surrounding the boundary point at previous steps in time. Equations 5.22 and 5.23 show the 2nd order Mur ABC for the E_z field at the Lower X boundary (Y-Z plane at X minimum), and Upper X boundary (Y-Z plane at X maximum) [48].

$$
\begin{aligned}
E_z\Big|^{n+1}_{(i_{min},j,k+\frac{1}{2})} = \\
&- E_z\Big|^{n-1}_{(i_{min}+1,j,k+\frac{1}{2})} - \left(\frac{\Delta x - v_p\Delta t}{\Delta x + v_p\Delta t}\right)\left(E_z\Big|^{n+1}_{(i_{min}+1,j,k+\frac{1}{2})} + E_z\Big|^{n-1}_{(i_{min},j,k+\frac{1}{2})}\right) \\
&+ \left(\frac{2\Delta x}{\Delta x + v_p\Delta t}\right)\left(E_z\Big|^{n}_{(i_{min},j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{min}+1,j,k+\frac{1}{2})}\right) \\
&+ \left(\frac{\Delta x(v_p\Delta t)^2}{2(\Delta y)^2(\Delta x + v_p\Delta t)}\right)\left(E_z\Big|^{n}_{(i_{min},j+1,k+\frac{1}{2})} - 2E_z\Big|^{n}_{(i_{min},j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{min},j-1,k+\frac{1}{2})}\right. \\
&\qquad\left. + E_z\Big|^{n}_{(i_{min}+1,j+1,k+\frac{1}{2})} - 2E_z\Big|^{n}_{(i_{min}+1,j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{min}+1,j-1,k+\frac{1}{2})}\right) \\
&+ \left(\frac{\Delta x(v_p\Delta t)^2}{2(\Delta z)^2(\Delta x + v_p\Delta t)}\right)\left(E_z\Big|^{n}_{(i_{min},j,k+\frac{3}{2})} - 2E_z\Big|^{n}_{(i_{min},j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{min},j,k-\frac{1}{2})}\right. \\
&\qquad\left. + E_z\Big|^{n}_{(i_{min}+1,j,k+\frac{3}{2})} - 2E_z\Big|^{n}_{(i_{min}+1,j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{min}+1,j,k-\frac{1}{2})}\right)
\end{aligned}
\tag{5.22}
$$

$$
\begin{aligned}
E_z\Big|^{n+1}_{(i_{max},j,k+\frac{1}{2})} = \\
&- E_z\Big|^{n-1}_{(i_{max}-1,j,k+\frac{1}{2})} - \left(\frac{\Delta x - v_p\Delta t}{\Delta x + v_p\Delta t}\right)\left(E_z\Big|^{n+1}_{(i_{max}-1,j,k+\frac{1}{2})} + E_z\Big|^{n-1}_{(i_{max},j,k+\frac{1}{2})}\right) \\
&+ \left(\frac{2\Delta x}{\Delta x + v_p\Delta t}\right)\left(E_z\Big|^{n}_{(i_{max},j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{max}-1,j,k+\frac{1}{2})}\right) \\
&+ \left(\frac{\Delta x(v_p\Delta t)^2}{2(\Delta y)^2(\Delta x + v_p\Delta t)}\right)\left(E_z\Big|^{n}_{(i_{max},j+1,k+\frac{1}{2})} - 2E_z\Big|^{n}_{(i_{max},j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{max},j-1,k+\frac{1}{2})}\right. \\
&\qquad\left. + E_z\Big|^{n}_{(i_{max}-1,j+1,k+\frac{1}{2})} - 2E_z\Big|^{n}_{(i_{max}-1,j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{max}-1,j-1,k+\frac{1}{2})}\right) \\
&+ \left(\frac{\Delta x(v_p\Delta t)^2}{2(\Delta z)^2(\Delta x + v_p\Delta t)}\right)\left(E_z\Big|^{n}_{(i_{max},j,k+\frac{3}{2})} - 2E_z\Big|^{n}_{(i_{max},j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{max},j,k-\frac{1}{2})}\right. \\
&\qquad\left. + E_z\Big|^{n}_{(i_{max}-1,j,k+\frac{3}{2})} - 2E_z\Big|^{n}_{(i_{max}-1,j,k+\frac{1}{2})} + E_z\Big|^{n}_{(i_{max}-1,j,k-\frac{1}{2})}\right)
\end{aligned}
\tag{5.23}
$$

The derivation for these equations can be found in [48]. The ABC equations for E_y on the X boundaries, E_z and E_x on the Y boundaries, and E_x and E_y on the Z boundaries can be found from Equations 5.22 and 5.23 through simple index manipulation.

Figure 5.5 is a graphical representation of the fields and cells required for Equation 5.23 (without the time dependence). This figure shows that when evaluating E_z on the boundary (bold field in figure), the fields surrounding it need to be known. This again causes a problem, as the fields on the boundary plane edges and corners cannot be evaluated using Equations 5.22 and 5.23.

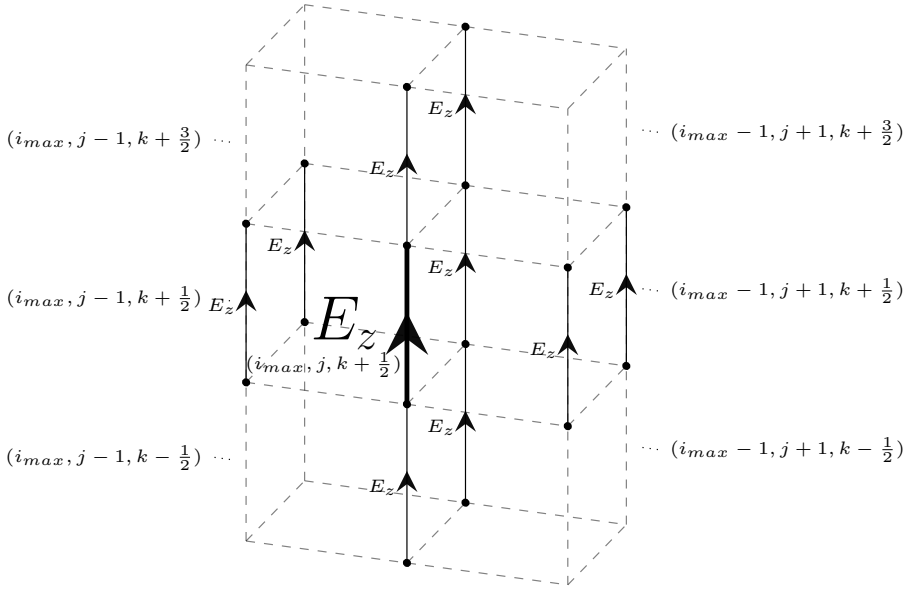


Figure 5.5: Cell Requirements for E_z at X Upper Boundary (2nd Order Mur)

Equations 5.24 and 5.25 are a rudimentary implementation of a 1st order Mur boundary condition for the plane edges and corners respectively [55]. This is by no means the best implementation for these boundary conditions, and will be responsible for higher reflections than those resulting from Equations 5.22 and 5.23.

$$E_z\Big|^{n+1}_{(i_{min},j_{min},k+\frac{1}{2})} = E_z\Big|^{n}_{(i_{min}+1,j_{min}+1,k+\frac{1}{2})} \tag{5.24a}$$

$$E_z\Big|^{n+1}_{(i_{max},j_{min},k+\frac{1}{2})} = E_z\Big|^{n}_{(i_{max}-1,j_{min}+1,k+\frac{1}{2})} \tag{5.24b}$$

$$E_z\Big|^{n+1}_{(i_{min},j_{max},k+\frac{1}{2})} = E_z\Big|^{n}_{(i_{min}+1,j_{max}-1,k+\frac{1}{2})} \tag{5.24c}$$

$$E_z\Big|^{n+1}_{(i_{max},j_{max},k+\frac{1}{2})} = E_z\Big|^{n}_{(i_{max}-1,j_{max}-1,k+\frac{1}{2})} \tag{5.24d}$$

$$E_z\Big|^{n+1}_{(i_{min},j_{min},k_{max}-\frac{1}{2})} = E_z\Big|^{n}_{(i_{min}+1,j_{min}+1,k_{max}-\frac{3}{2})} \tag{5.25a}$$

$$E_z\Big|^{n+1}_{(i_{max},j_{min},k_{max}-\frac{1}{2})} = E_z\Big|^{n}_{(i_{max}-1,j_{min}+1,k_{max}-\frac{3}{2})} \tag{5.25b}$$

$$E_z\Big|^{n+1}_{(i_{min},j_{max},k_{max}-\frac{1}{2})} = E_z\Big|^{n}_{(i_{min}+1,j_{max}-1,k_{max}-\frac{3}{2})} \tag{5.25c}$$

$$E_z\Big|^{n+1}_{(i_{max},j_{max},k_{max}-\frac{1}{2})} = E_z\Big|^{n}_{(i_{max}-1,j_{max}-1,k_{max}-\frac{3}{2})} \tag{5.25d}$$

Figure 5.6 shows the graphical representation of Equations 5.24, which are applied to every X-Y layer of the simulation space. Figure 5.7 shows the graphical representation of Equations 5.25. A full implementation of all these boundary conditions can be found in the example code of Appendix D.

These equations and figures are only applicable to the E_z field boundary conditions. These equations need to be adapted to the remaining electric fields, and applied to the relevant edges and corners.

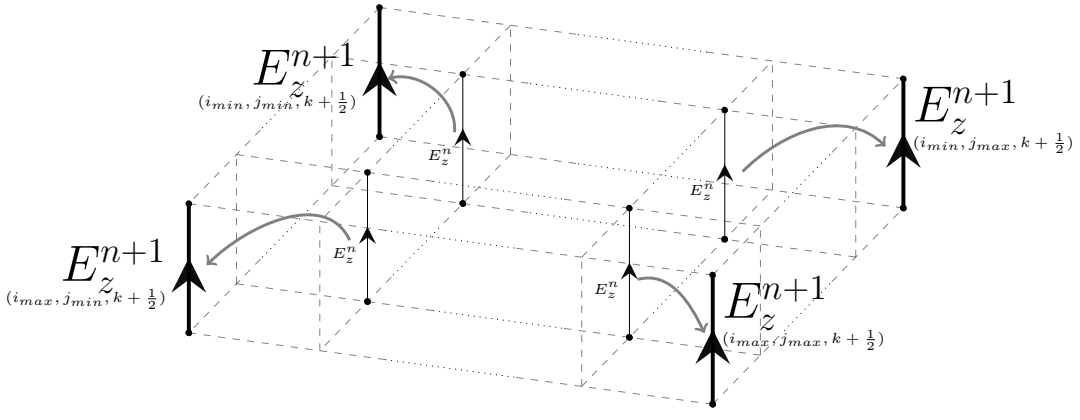


Figure 5.6: Cell Requirements for E_z at Boundary edges (1st Order Mur)

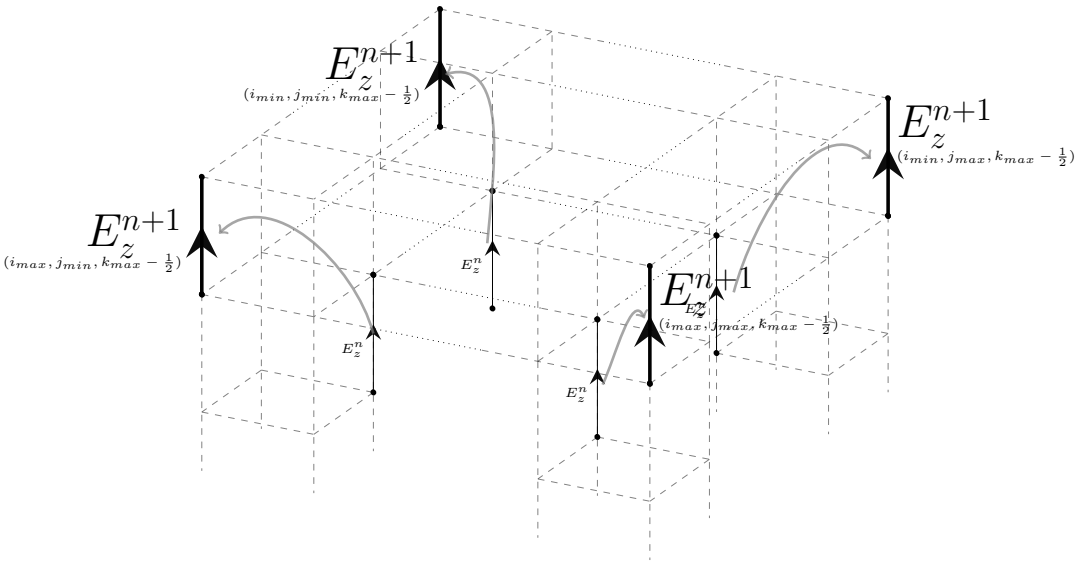


Figure 5.7: Cell Requirements for E_z at Corners (1st Order Mur)

There are two recommendations for improving the boundary conditions. The first is to implement higher order Mur conditions for the edges and corners of the evaluation space. The second is to use the Perfectly Matched Layer (PML) method. The PML method involves creating a padded layer of fictitious material between the simulation space and the boundaries. The properties of the cells in these layers are designed to minimise reflections of waves entering the material, and then attenuate the waves sufficiently as they move through the material. In doing so any waves that do reflect back into the simulation space are negligible, and create the effect that the EM fields propagated past the boundaries.

5.2.4 Simulation space objects

Objects are added to a simulation space in order to simulate the effects that these objects have on the EM fields. In terms of lightning these objects could be trees, buildings, people, distribution lines, ground planes, complex terrain, etc. For a simple lightning FDTD simulation space, as seen in Figure 5.1, only two objects are needed. The first is a ground plane, and the second is the lightning channel current (EM source).

Adding physically modelled objects to the simulation space is done by modifying the Yee Cells (permittivity (ϵ), permeability (μ) and conductivity (σ^e)) at the location of the object in the simulation space [49]. In doing this the ϵ, μ and σ^e terms become spatially dependent, which means that the multiplication constants in Equations 5.16 also become spatially dependant.

If ϵ, μ and σ^e of a ground plane are known then these parameters can be used to create a multilayer ground inside the simulation space. The same is true for any other object of interest. The limitation is that these objects can only be made out of cells, which is a problem for spherical objects, or objects with curved surfaces. Methods for dealing with these problems are beyond the scope of this work, but are discussed in [49].

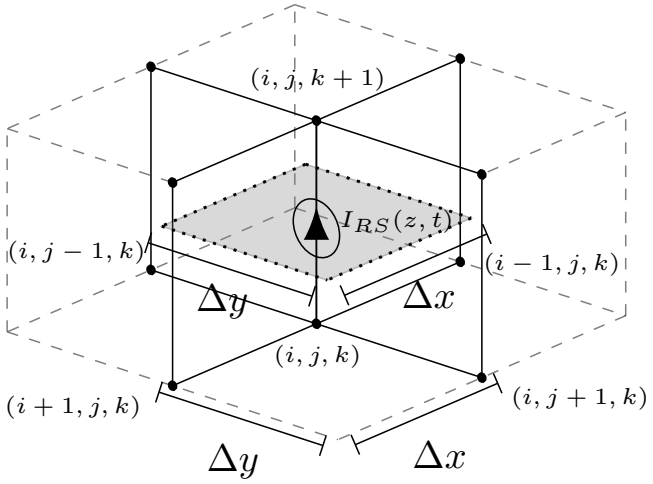


Figure 5.8: 3D FDTD Lightning Current Source Component - J_{iz} Current Density

Adding the lightning channel current to the simulation is equally trivial. Figure 5.8 shows how a current element is added between two nodes, at the same location as the electric field. The current at this location is defined by the return stroke model (I_{RS}), where its height (z) and time (t) are defined by $(k + \frac{1}{2}) * \Delta z$ and $n * \Delta t$ respectively. This is done for all the nodes that make up the lightning channel.

The current density between these nodes is then defined by Equation 5.26. The area used for this current density is the cell X-Y plane area, and not the cross sectional area of the lightning channel, as seen in Figure 5.8. The current density J_{iz} is then used in Equation 5.12 for all the locations that make up the lightning channel. For the simulations in this work the lightning channel is placed at the center of the X-Y plane, along the entire Z direction (J_{iz}). No other sources are used, and therefore J_{ix} and J_{iy} are zero.

$$J_{iz}(z,t) = \frac{I_{RS}(z,t)}{\Delta x.\Delta y} \tag{5.26}$$

It is also possible to model resistors, capacitors, inductors, diodes and voltage sources between nodes. This would allow for full distribution lines to be modelled within the simulation space, as well as voltage and current measurements to be simulated on the distribution line in the presence of the EM field. This is a significant advantage of the method with regards to LIOV based work. The traditional approach would be to use one model to describe the lightning EM fields at multiple points along a distribution line, and a separate model to describe the lightning induced voltage [11]. By using the 3D FDTD model it is possible to extract the same simulation results from a single model.

5.2.5 Computational Considerations

Figure 5.9 shows an example FDTD simulation space which has $3 * 4 * 2$ Yee Cells in the X, Y, and Z directions respectively. This same simulation space can also be described by the nodes, where there are $4 * 5 * 3$ nodes in the X,Y and Z directions. The variables N_x, N_y and N_z are the number of nodes in each direction. Another important consideration is the number scheme used in this work. The programming language used

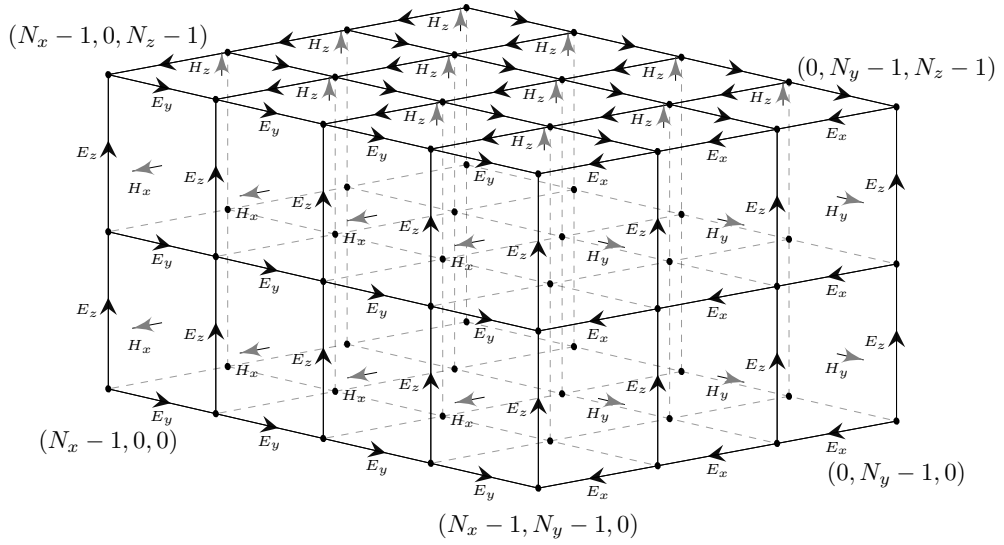


Figure 5.9: EM Fields assigned to FDTD Simulation Space

for the simulations is $C++$, which uses "0" to address the first variable in an array [55], rather than "1" which is used in MATLAB® [49]. This indexing arrangement can be the cause of numerous "one off errors" in code, which is why understanding the indexing and numbering arrangement is important.

Similar "one off" issues occur due to the Yee Cell arrangement. Consider the E_z fields in Figure 5.9. In the X direction there are N_x nodes, and $N_x \times E_z$ fields. The same is true for the Y direction, however in the Z direction there are N_z nodes, and $(N_z-1)\times E_z$ fields. Therefore the E_z memory matrix has "$N_x.N_y.(N_z-1)$" elements. Similar equations exist for the other two electric fields. The inverse is true for the magnetic fields, where H_z has N_z field components in the Z direction, (N_x-1) in the X direction and (N_y-1) in the Y direction. Therefore the H_z memory matrix has "$(N_x-1).(N_y-1).N_z$" elements.

The primary concern regarding the field matrix sizes, is the computer memory required for the simulation. Equations 5.27 show the exact and approximate number of memory elements.

$$\begin{aligned} N_{mem} = {} & (N_x-1).N_y.N_z + N_x.(N_y-1).N_z + N_x.N_y.(N_z-1) \\ & + N_x.(N_y-1).(N_z-1) + (N_x-1).N_y.(N_z-1) + (N_x-1).(N_y-1).N_z \end{aligned} \tag{5.27a}$$

$$N_{mem} \simeq 6.N_x.N_y.N_z \tag{5.27b}$$

For example, consider a simulation space that describes a lightning channel 8000 m high, located at the center of an X-Y plane that extends 200 m in both positive and negative X, Y directions. Assuming a Yee Cell that is $1\,\mathrm{m}^3$ ($\Delta x = \Delta y = \Delta z = 1\,\mathrm{m}$), there would be approximately $7.68 * 10^9$ elements. Assuming a "double" variable type (8 bytes) is used, the total memory required to store the EM field data during a simulation would be 57 GB. If the simulation space consists of a complex environment, then Equations 5.16 would become a matrix (rather than a constant) and double the memory requirement.

Figure 5.10 shows the generic flow diagram of an FDTD code implementation [49]

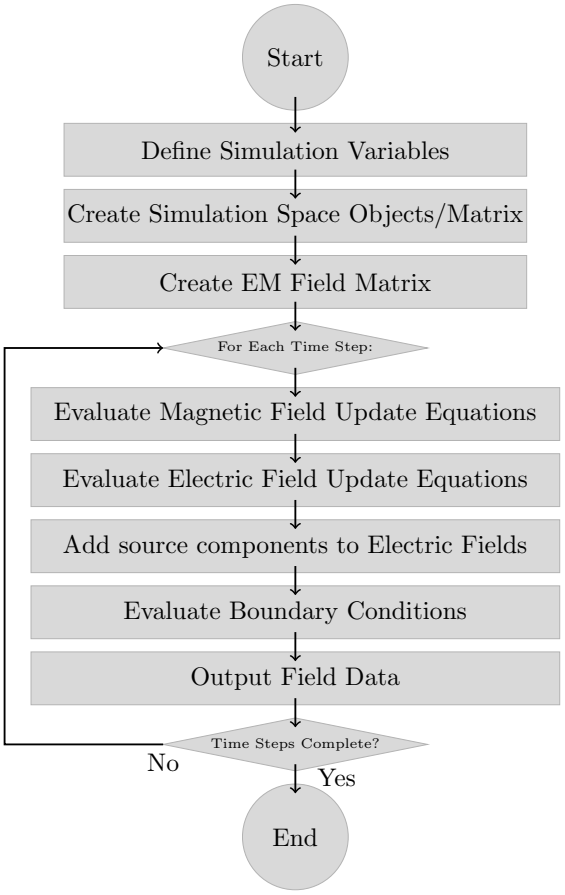


Figure 5.10: FDTD Software Flow Diagram

(example code in Appendix D). The simulation variables define the simulation space size, grid size, lightning stroke parameters, and various other parameters. The EM field matrices are reserved in computer memory for the duration of the simulation. Given the high memory requirements of this method it may be possible to distribute the field matrices into a SWAP partition, however the speed of the simulation is drastically reduced. Therefore it is preferable to keep all simulation matrices in memory, rather than in storage media. For the average computer user this memory requirement poses a problem. A simple solution is to decrease simulation space size, decrease Yee Cell size, or decrease simulation time. The effects of these options are discussed in Section 5.3.

5.3 Model Simulations

This section uses the above mentioned theory to produce some example fields. These fields are then compared to the fields produced by the Finite Antenna method to demonstrate the discrete effects of the FDTD method.

Figure 5.11 shows an FDTD model environment for a basic lightning simulation. The simulation space is $(2.X_{dis}) * (2.Y_{dis}) * Z_{dis}$ m^3 in size. The lightning channel is placed in the center of the X-Y plane, and runs perpendicular for the full height (Z_{dis}). The

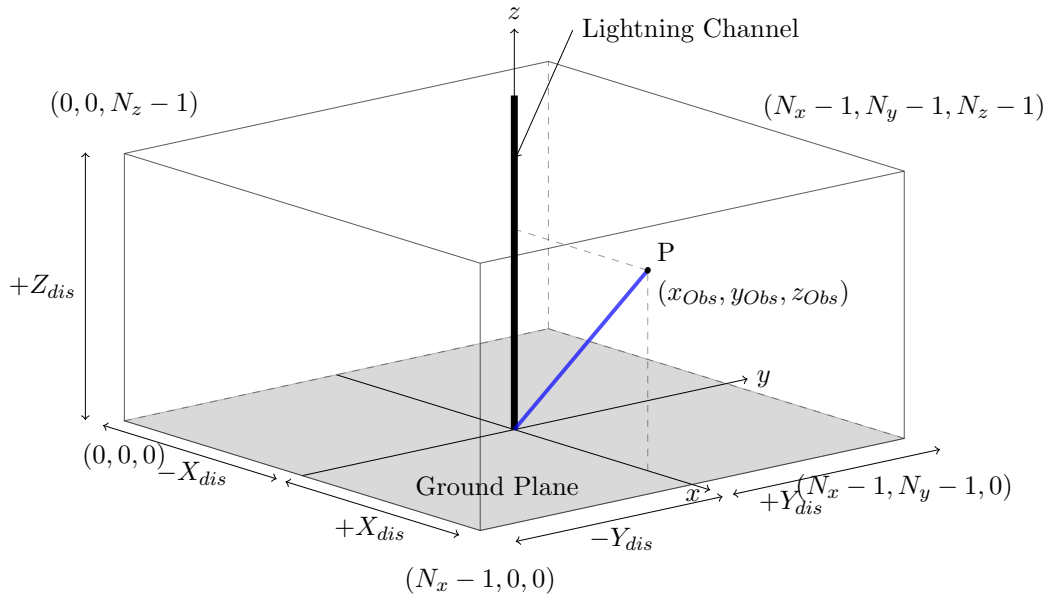


Figure 5.11: Basic FDTD Model Structure

lightning channel current is described by the transmission line (TL) return stroke model (Equation 3.12) with the current impulse described by Equation 3.7 (Terespolsky version of the popular current impulse), and a return stroke speed of $0.5c$. The observation point of interest is defined at location P. For simplicity in field co-ordinate conversions the observation point in the following examples is kept on the X-Z plane where $y_{Obs} = 0$.

The ground plane is assumed to be a perfect electrical conductor (PEC). Rather than add ground layers with the relevant conductance values ($\sigma^e = 1$), it is possible to take advantage of the PEC property of the simulation space boundary conditions. Therefore no boundary conditions will be solved for the $Z = 0$ plane. This simplifies the process of adding ground plane objects, and additional matrices for the FDTD algorithm.

The area surrounding the lightning channel is assumed to be free space, where $\mu_r = \mu_0$, $\epsilon_r = \epsilon_0$, $\sigma^e = 0$, and propagation velocity $V_p = c$ throughout the simulation space. The only variables that still need to be defined are the simulation space dimensions, Yee Cell dimensions and the simulation run time.

Figure 5.12 shows the shortest paths of the EM fields that reflect off the X, Y and Z boundaries. If the simulation time (T_{sim}) is known, as well as the observation point location, it is possible to calculate the dimensions of the simulation space such that the reflections only reach the observation point at the end of the simulation. Equations 5.28 show the calculations for the minimum simulation space dimensions with zero effects from boundary reflections. The simulation space can be made larger, but this will have no effect on the observed field, and will only increase memory requirements of the simulation. The $-X_{min}$ distance will be less than the $+X_{min}$ distance, however for symmetrical convenience they are kept equal.

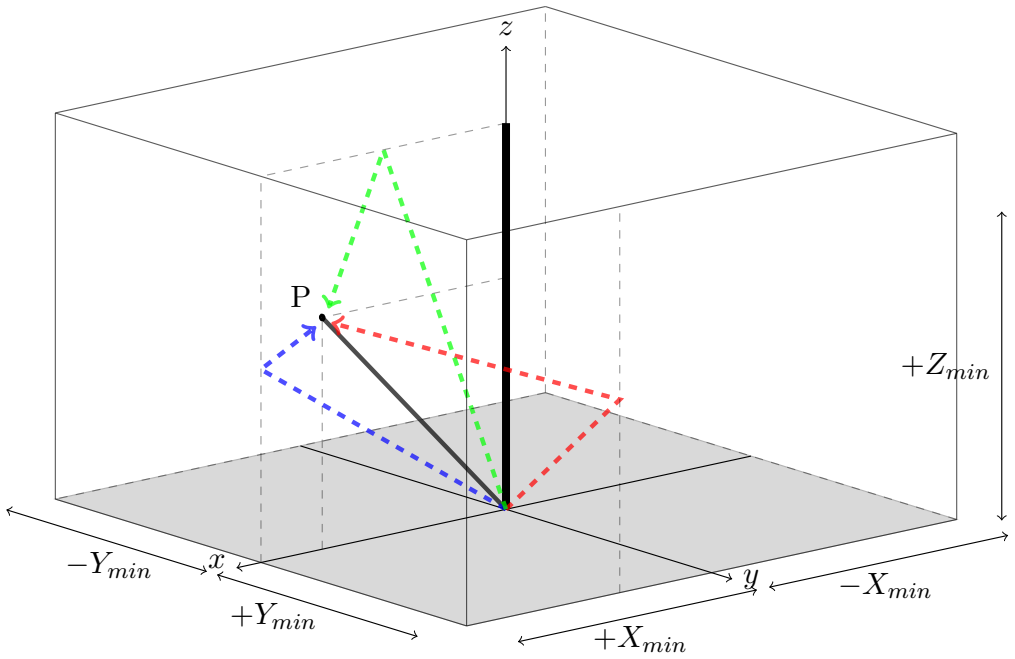


Figure 5.12: EM Reflections on Simulation Space Boundaries

$$+X_{min} = x_{Obs} + \frac{\sqrt{(T_{sim}.V_p)^2 - z_{Obs}^2} - x_{Obs}}{2} \tag{5.28a}$$

$$+Y_{min} = \frac{\sqrt{(T_{sim}.V_p)^2 - x_{Obs}^2 - z_{Obs}^2}}{2} \tag{5.28b}$$

$$+Z_{min} = z_{Obs} + \frac{\sqrt{(T_{sim}.V_p)^2 - x_{Obs}^2} - z_{Obs}}{2} \tag{5.28c}$$

The simulation time T_{sim} is a combination of the time taken to reach the observation point, and the time required to observe the field at point P. The following examples will demonstrate the effects of varying the simulation space, as well as the Yee Cell dimensions.

Example Case

The observation point is set to 50 m from the lightning channel, and 10 m off the ground. This follows the example given in Section 4.3 (Figure 4.4), which aims to evaluate the worst case position of a distribution line with respect to a lightning RS channel. The field review time is chosen to be 3 µs, which means the simulation time is 3170 ns.

Using the simulation time and observation point, the simulation space has X_{dis}, Y_{dis} and Z_{dis} set to 501 m, 474 m and 479 m respectively. These are the minimum dimensions for the simulation space where reflections from the boundaries will not affect the observation point fields.

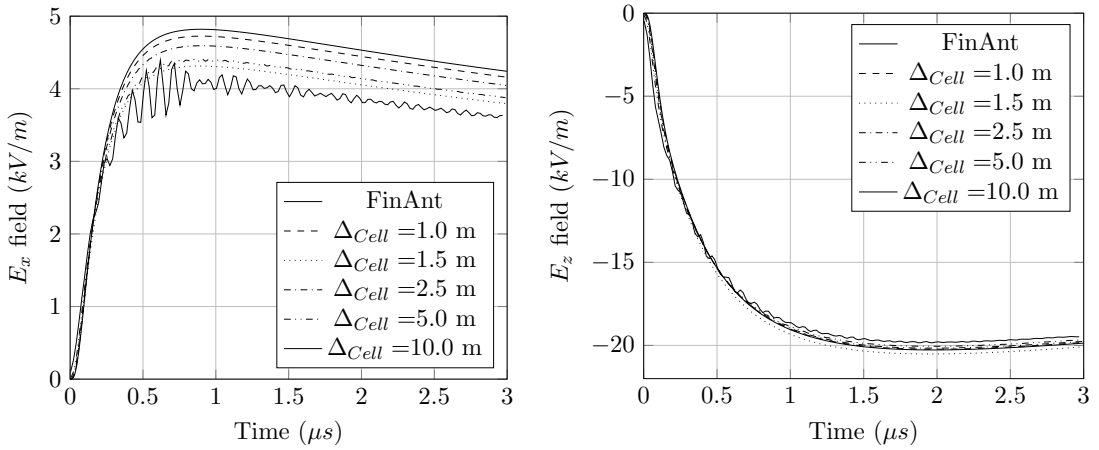


Figure 5.13: 3D FDTD Fields - $x_{Obs} = 50\,\text{m}$, $z_{Obs} = 10\,\text{m}$ - Variable Grid Size

The Yee Cells are chosen to have a uniform dimension where $\Delta x = \Delta y = \Delta z = \Delta_{cell}$. Figure 5.13 shows the E_x and E_y fields at the observation point produced by the 3D FDTD method with various Yee Cell sizes. This figure also shows the fields produced by the Finite Antenna method at the same observation point.

The E_z fields in Figure 5.13 all show good agreement, however the E_x fields do not. This figure has the equivalent fields produced by the Finite Antenna method to indicate what the fields should be at this location. Reviewing the E_x fields exposes a number of properties of the FDTD method. The first observation is that using a Yee Cell with $\Delta_{Cell} = 10\,\text{m}$ produces a field that is significantly different (almost 20% error) from the expected field produced by the Finite Antenna method. In addition to this the field appears to be unstable. In this situation the jagged curve is not a result of instability, but rather a coarse time step. Due to the size of the cell being relatively large, the time step derived by the CFL Equation 5.21 is also large. This time step could be made smaller, however the code that implemented this simulation (Appendix D) used the maximum allowable time step to reduce computation time. This time step also enhanced the effects of the source start up transient, which is caused by the discrete changes in the source. This is why the curve becomes more stable as time progresses, rather than growing exponentially, which would indicate numerical instability. The other FDTD fields will also contain the effects of the start up transient, however these are damped sooner due to the finer cell size, and smaller time steps.

The second observation is that the FDTD fields tend towards the Finite Antenna values when the cell size is made smaller. This makes intuitive sense given that the finer resolution would indicate a higher level of accuracy, however this is not entirely true. The E_x field in the figure shows that the $\Delta_{Cell} = 1.5\,\text{m}$ field is worse than the $\Delta_{Cell} = 5\,\text{m}$ field and all the smaller cells. The reason for this difference is the discrete placement of Yee Cell fields and nodes on the FDTD space, as illustrated in Figure 5.14.

This figure shows a single Yee Cell, and the fields associated with the corner node situated at the observation point (P). One of the fundamental features the FDTD method is that all six fields are located at different points in space, as compared to the Finite Antenna method (3 fields due to axial symmetry) that has all its fields located

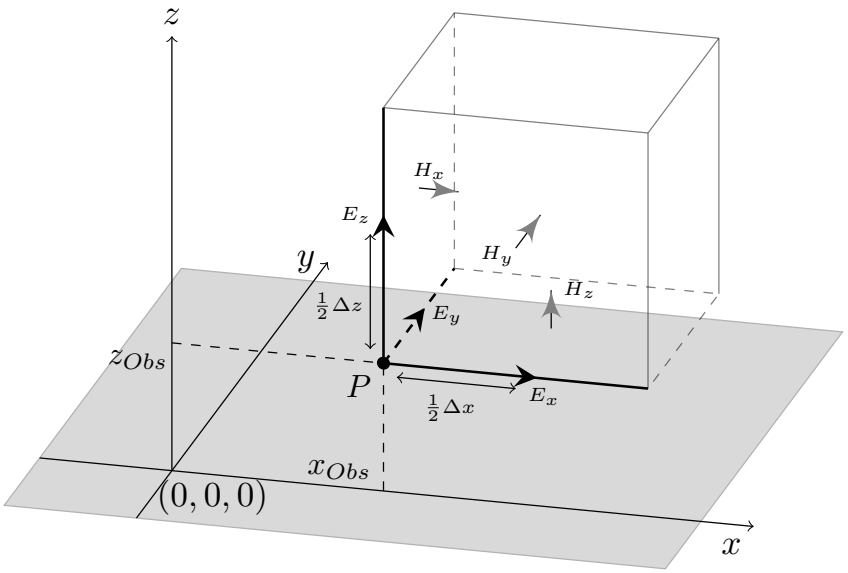


Figure 5.14: Yee Cell - Discrete Location of Nodes and Fields

at the same point in space (P). Table 5.1 shows the X-Y co-ordinate locations for the effective observation point node, the E_x field location and the E_z field locations for each of the Yee Cells in Figure 5.13.

This table shows that the $\Delta_{Cell} = 1.5\,\text{m}$ field is centred around a different effective observation point. This difference in observation point is caused by the discretization of the FDTD simulation space into the Yee Cells, where the cell corner nodes must be located at integer multiples of cell dimensions. The table also shows that the E_x and E_z fields are located at half integer cell dimensions from the effective observation point. This explains why the $\Delta_{Cell} = 1.5\,\text{m}$ field had a significantly different field and trend from the other fields.

Table 5.1 also helps to show that the differences in the fields of Figure 5.13 are not technically errors, but rather incorrectly referenced to the observation point location. Figures 5.15 show the E_x and E_z fields produced by the FDTD method using an Yee Cell with $\Delta_{Cell} = 5\,\text{m}$, as well as the fields produced by the Finite Antenna method at the original and shifted observation points.

Figures 5.15 show that the fields of the Finite Antenna method at the shifted locations

Table 5.1: Example Case: Yee Cell Field Locations (m)

Δ_{cell}	(x_{Obs},z_{Obs}) (m)	E_x Field Location	E_z Field Location
1	(50.0 , 10.0)	(50.50 , 10.0)	(50.0 , 10.50)
1.5	(49.5 , 9.0)	(50.25 , 9.0)	(49.5 , 9.75)
2.5	(50.0 , 10.0)	(51.25 , 10.0)	(50.0 , 11.25)
5	(50.0 , 10.0)	(52.50 , 10.0)	(50.0 , 12.50)
10	(50.0 , 10.0)	(55.00 , 10.0)	(50.0 , 15.00)

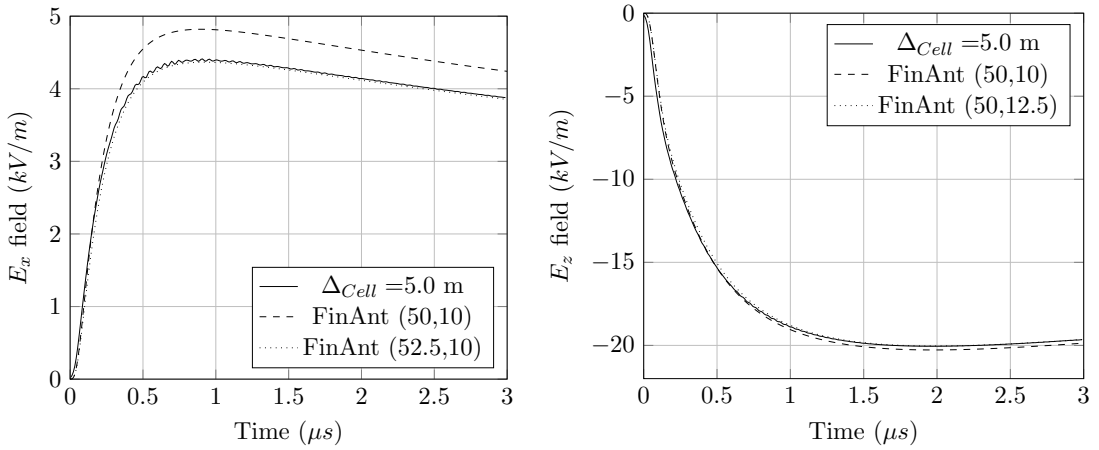


Figure 5.15: 3D FDTD Fields - $x_{Obs} = 50\,\text{m}$, $z_{Obs} = 10\,\text{m}$ - $\frac{1}{2}\Delta_{Cell}$ field shift.

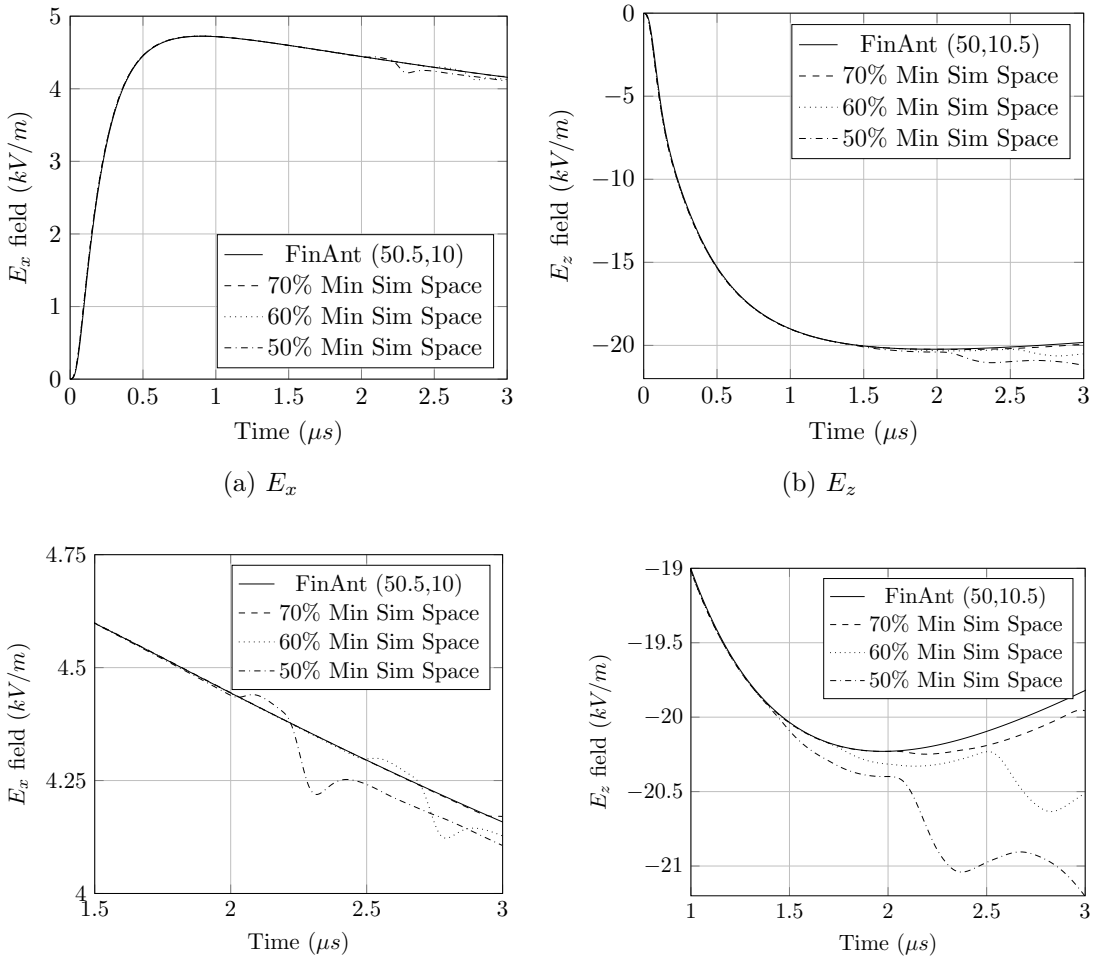
on the Yee Cell compare well with the fields of the FDTD method. This highlights a limitation of the FDTD method. It is possible to achieve acceptably accurate results with FDTD, even when using large Yee Cells (spatial discretization), however the tradeoff is that the observation points are limited to the discrete locations.

The FDTD E_x field of Figure 5.15 also shows the jagged effects of the start up transient, which in comparison to Figure 5.13, have been significantly reduced. These effects can be further reduced by using smaller Yee Cells to define the simulation space. Another reason to use smaller Yee Cells is to reduce the problem of the shifted fields described above. The problem with discretizing the simulation space into smaller Yee Cells is that the simulation would require larger amounts of computer memory to store the field values.

Table 5.2 shows the memory requirements for different Yee Cell dimensions (using the simulation space of the example). From this table it becomes clear that establishing a fine scale spatial resolution in the simulation space can be practically difficult. In addition to this, smaller Yee Cells require smaller time steps, which also means that the simulations will take longer to complete. Therefore there is a trade-off between producing smooth waveforms with fine spatial resolution, and the computational limitations. One method to aid this limitation is to decrease the simulation space.

Table 5.2: Example Case: Cell Computer Memory Requirements

Δ_{cell} (m)	**Effective** x_{Obs} (m)	**Effective** z_{Obs} (m)	**Mem** (GB)
0.25	50	10	1302
0.5	50	10	162.7
1	50	10	20.34
1.5	49.5	9	6.03
2.5	50	10	1.3
5	50	10	0.163
10	50	10	0.02



(a) E_x (b) E_z

(c) E_x Magnified (d) E_z Magnified

Figure 5.16: 3D FDTD Fields - $x_{Obs} = 50\,\text{m}$, $z_{Obs} = 10\,\text{m}$ - Variable Simulation Space.

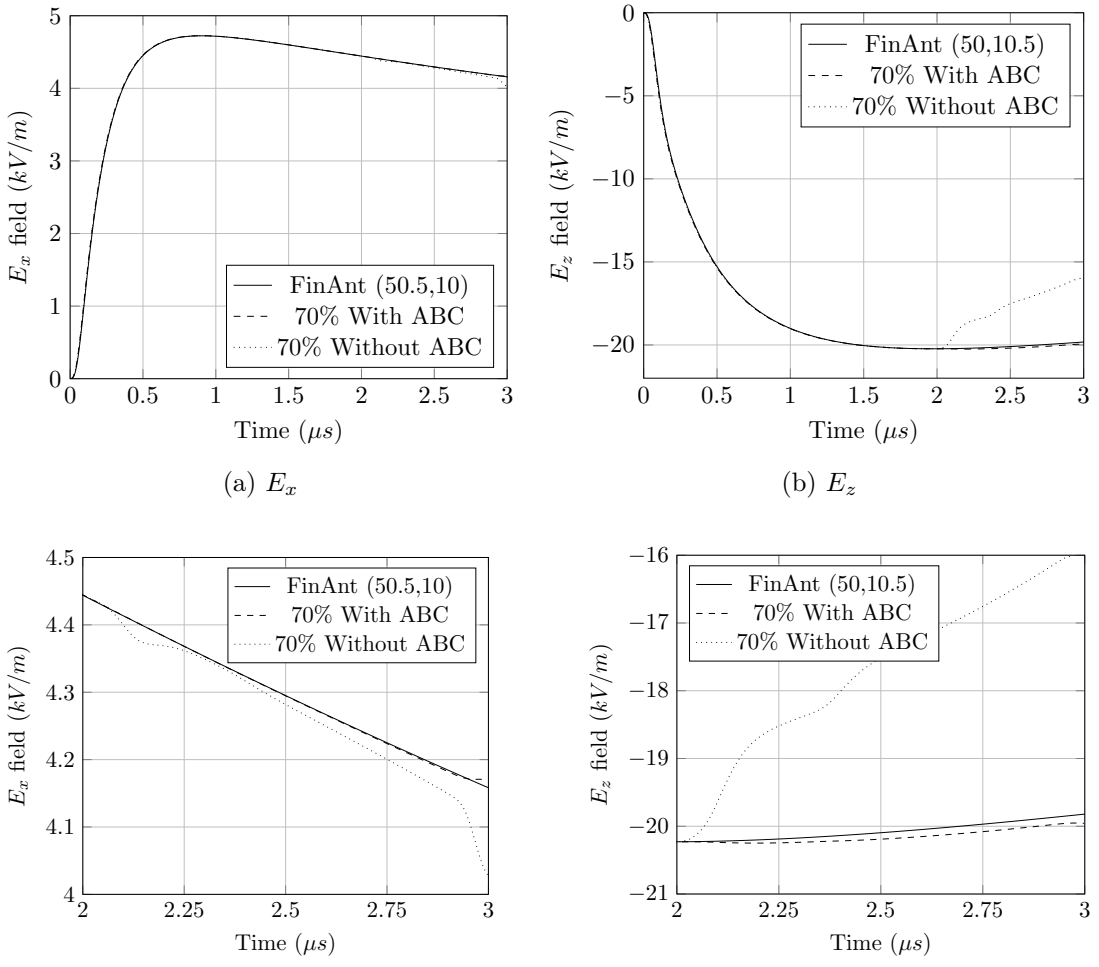
All the FDTD example fields presented so far have used the minimum dimensions required for no boundary reflections to reach the observation point. Equations 5.28 define these minimum boundaries, and it is clear that these equations are time dependant. A long simulation time allows EM fields to travel further, which means that the boundaries of the simulation space need to be further away from the observation point. Therefore it is preferable to make the simulation time as short as possible. However it is also possible to reduce the simulation space and rely on the absorbing boundary conditions (ABC) to manage the reflections. Figures 5.16 show the electric fields at the observation point using different percentages of the minimum simulation space size, and a $\Delta_{Cell} = 1\,\text{m}$.

The fields in Figure 5.16 show that as the simulation space is made smaller, the boundary reflected fields have a greater effect on the simulated fields. It is important to understand that this effect is a numerical consequence of the boundary planes, and not a modelled physical phenomena. From the figures it is clear that using 70% of the minimum simulation space (for each dimension) produces an acceptable field value. The benefit of this spatial reduction is outlined in Table 5.3, which shows that the computer memory requirements have been reduced to 34% of the original memory (0.7*0.7*0.7).

Table 5.3: Example Case: Computer Memory for different Simulation Space Dimensions

% of Min Boundary	X_{dis}	Y_{dis}	Z_{dis}	**Mem** (GB)
100%	501	474	479	20.3401
90%	450.9	426.6	431.1	14.8279
80%	400.8	379.2	383.2	10.4141
70%	350.7	331.8	335.3	6.97665
60%	300.6	284.4	287.4	4.39346
50%	250.5	237	239.5	2.54251

All the fields shown in this section were produced with a simulation space using the ABC's discussed in Section 5.2.3. Figure 5.17 shows the electric fields produced with a 70% dimension simulation space, with and without ABC's implemented on the boundaries. This shows that using ABC's can significantly reduce the simulation space without significantly affecting the fields at the observation point. As discussed in Section 5.2.3, by using higher order ABC's it is possible to reduce the simulation space further.



(a) E_x (b) E_z

(c) E_x Magnified (d) E_z Magnified

Figure 5.17: 3D FDTD Fields - $x_{Obs} = 50\,\text{m}$, $z_{Obs} = 10\,\text{m}$ - With and Without ABC.

5.4 Chapter Summary

This chapter has presented a comprehensive development of the 3D FDTD method for lightning based simulations. It has discussed the theory of the FDTD method, as well as the essential mathematics. In addition to this the various numerical effects have been discussed along with the methods for reducing the effects. The process of developing a lightning simulation space was dealt with in great detail, as well as the methodology for expanding the model for complex environments. A section also discussed the practical computational limitations of the FDTD method, which affects the manner in which a simulation is implemented.

Section 5.3 showed the resulting electric fields from a 3D FDTD lightning simulation, as well as how critical FDTD variables affect these simulated fields. The modelled environment for this example case was deliberately kept simple so that the resulting fields from the Finite Antenna method could be used for comparison, however it is important to highlight the fact that this is a special case. The 3D FDTD method offers far more simulation variability that the Finite Antenna method cannot. The 3D FDTD method allows for complex ground planes, complex structures, as well as electrical component simulations. This is important because it would allow for distribution line elements to be included in the simulation space, which would then allow for LIOV simulations to be completed with the same EM simulation. The Finite Antenna method cannot include such complex simulation environments, and would require a separate model and simulation to gain LIOV simulation results.

The one major drawback to this method is that it is best suited for observation points relatively close to the lightning channel. Points that are further away (in the km scale) require large amounts of computer memory, and although this may not be a problem for institutions with large computing facilities, it does pose a problem for private PC owners. The next section will present the 2D FDTD method, which can produce EM fields at a finer spatial resolution than the 3D FDTD method, and use less computer memory. The drawback to this method is that it can only consider relatively simple simulation environments, but still consider complex ground planes.

Chapter 6

2D FDTD Method

The previous chapter introduced the FDTD method in 3D Cartesian co-ordinates. The advantage of this method is that it allows for highly complex and variable model environments to be simulated. The disadvantage is that the method requires high computer memory to achieve simulations with a fine spatial resolution. If the model environment is chosen to be axially symmetric around the lightning channel, then it is possible to reduce a 3D environment into a 2D environment and still achieve the same field information. The advantage of reducing the dimensions is a significant reduction in memory requirements, however the model is limited to axially symmetric simulation spaces. This chapter presents the theory and basic mathematical steps involved in creating a 2D FDTD model of lightning electromagnetics. The FDTD theory presented in Chapter 5 is also applicable in this chapter, and therefore will not will be discussed in detail.

6.1 Overview

This method requires the lightning RS channel to be kept straight and perpendicular to the ground plane. If this is true then the fields at a fixed radial distance from the channel will remain constant circling the channel. This is known as axial symmetry, and using Cylindrical co-ordinates (Figure 3.13b) allows for this symmetry to be advantageous.

Figure 6.1 shows an example simulation space divided into Yee Cells with nodes defined in cylindrical co-ordinates. The (i, j, k) co-ordinate indices are different to those in the previous chapter, and are defined by $\widehat{D}$, $\widehat{\phi}$ and $\widehat{z}$ respectively from Figure 3.13b. In comparison to Figure 5.1 (Cartesian grid), this figure reveals a number of interesting properties. The first is that the cell nodes are defined at discrete angular increments. In addition to this any node defined at a 0° is the same as 360° ($N_\phi - 1$). Therefore $360°/\Delta\phi$ must be an integer. This also requires a unique FDTD update equation for fields on the 0° axis.

Additionally the Yee Cells in Figure 6.1 are not uniform throughout the simulation space, as they are in Figure 5.1. Figure 6.2 shows a Yee Cell defined in Cylindrical co-ordinates, and the equivalent Cartesian cell is shown in Figure 5.2. This figure shows that the Yee Cell has a fixed Δz and ΔD throughout the simulation space, however

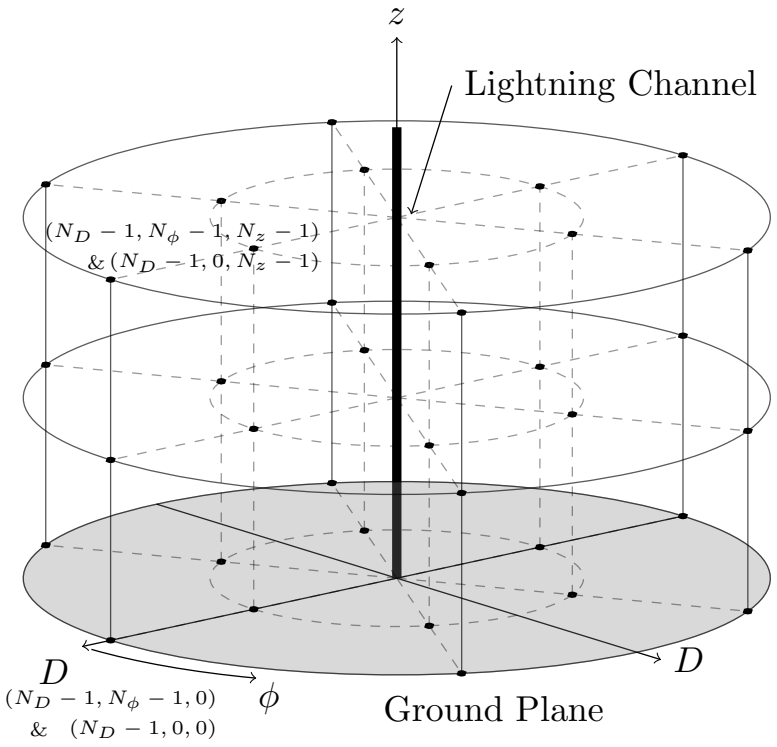


Figure 6.1: 3D Simulation space divided into cells - Cylindrical

the arc distance on the cells are dependant on the radial distance (D). There are two consequences of this. The first is that for large values of D (observation point far from the channel), the Yee Cells become large. This also means that the spatial resolution throughout the simulation space changes, and gets worse as the distance D increases.

The second consequence is seen at observation points near, or on the channel. As the observation point gets closer to the channel the nodes defined at (i, j, k) and $(i, j+1, k)$ converge to the same point ($D = 0$ therefore $D.\Delta\phi = 0$). Therefore there is no angular dependence at points on the channel. These points also require a unique FDTD update equation, which is different to the rest of the simulation space.

It is possible to implement a full 3D Cylindrical FDTD simulation space, however for complex environments it would be better to use the 3D Cartesian FDTD method described in Chapter 5. Therefore if the simulation space is kept axially symmetric around the lightning channel it is possible to evaluate the 3D Cylindrical simulation space in 2 dimensions defined by $\widehat{D}$ and $\widehat{z}$.

This is intuitive when considering a fixed (D_{Obs} and z_{Obs} constant) observation point. The fields at this point would not change as ϕ_{Obs} is varied between 0° and 360°. An analogy to this is the ripples formed when dropping a pebble into a pond. The ripples radiate away from the center with a constant amplitude. Mathematically this can be represented by Equation 6.1, which describes fields that have no angular difference.

$$\frac{d}{d\phi}.field = 0 \tag{6.1}$$

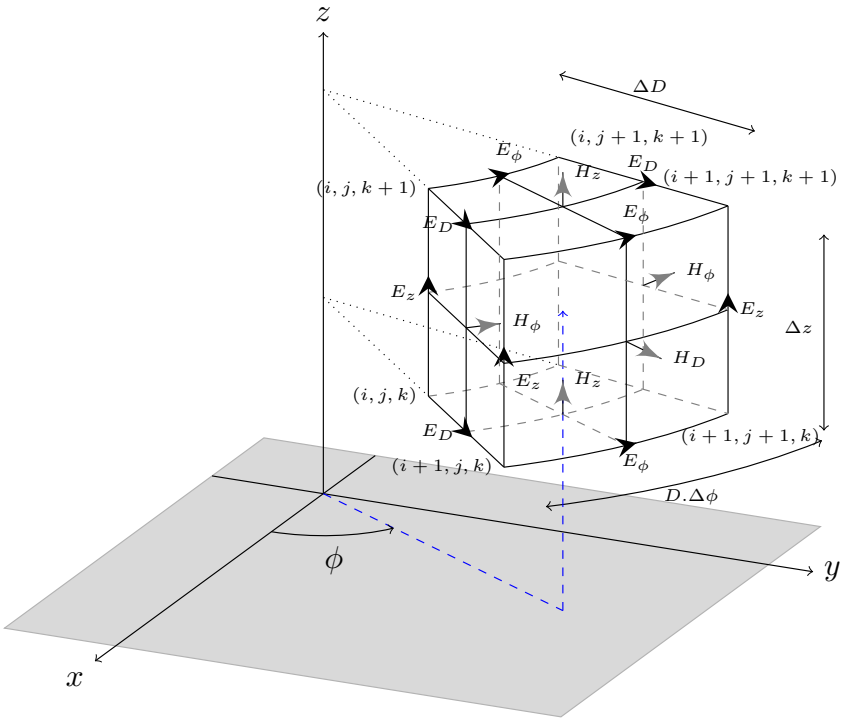


Figure 6.2: 3D Yee Cell - Cylindrical Co-ordinates - H Field centred

Because of this angular independence an "D-Z" plane would be the same at any angle ϕ. Therefore a 2D plane is able to provide the same field information as a 3D plane. Figure 6.3 shows the 2D plane that will be evaluated.

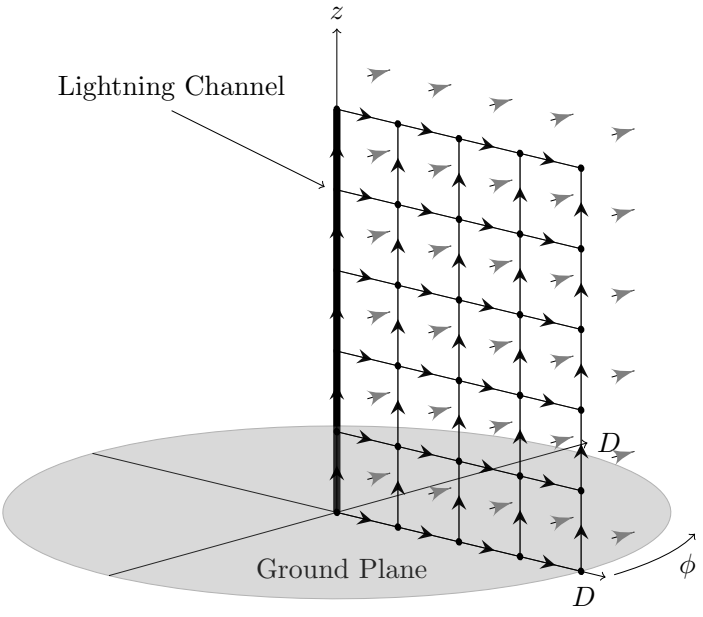


Figure 6.3: 2D Simulation space divided into cells - Showing 3D Cylindrical area. Electric field in black, Magnetic field in gray.

6.2 Model Development

The development follows the same process as that in Chapter 5. The primary difference is that the Cylindrical co-ordinate system is used in developing the FDTD equations.

6.2.1 2D FDTD Equations

Equations 3.15 (Faraday's Law) and 3.17 (Ampere's Law), in combination with Equation 3.27b (Section 3.5) are used to define the required Maxwell's equations in Cylindrical co-ordinates. Equations 6.2 and 6.3 show the expanded differential form of these equations.

$$\begin{aligned}\nabla \times \vec{E} &= \begin{bmatrix} \widehat{D} & \widehat{\phi} & \widehat{z} \\ \frac{\partial}{\partial D} & \frac{1}{D}\frac{\partial}{\partial \phi} & \frac{\partial}{\partial z} \\ E_D & E_\phi & E_z \end{bmatrix} = \frac{1}{D}\begin{bmatrix} \widehat{D} & D\widehat{\phi} & \widehat{z} \\ \frac{\partial}{\partial D} & \frac{\partial}{\partial \phi} & \frac{\partial}{\partial z} \\ E_D & D.E_\phi & E_z \end{bmatrix} = -\mu\frac{\partial \vec{H}}{\partial t} \\ &= \widehat{D}\ \frac{1}{D}\left(\frac{\partial E_z}{\partial \phi} - \frac{\partial (D.E_\phi)}{\partial z}\right) - \widehat{\phi}\ \left(\frac{\partial E_z}{\partial D} - \frac{\partial E_D}{\partial z}\right) + \widehat{z}\ \frac{1}{D}\left(\frac{\partial (D.E_\phi)}{\partial D} - \frac{\partial E_D}{\partial \phi}\right)\end{aligned} \tag{6.2}$$

$$\begin{aligned}\nabla \times \vec{H} &= \begin{bmatrix} \widehat{D} & \widehat{\phi} & \widehat{z} \\ \frac{\partial}{\partial D} & \frac{1}{D}\frac{\partial}{\partial \phi} & \frac{\partial}{\partial z} \\ H_D & H_\phi & H_z \end{bmatrix} = \frac{1}{D}\begin{bmatrix} \widehat{D} & D\widehat{\phi} & \widehat{z} \\ \frac{\partial}{\partial D} & \frac{\partial}{\partial \phi} & \frac{\partial}{\partial z} \\ H_D & D.H_\phi & H_z \end{bmatrix} = +\epsilon\frac{\partial \vec{E}}{\partial t} + \sigma\vec{E} + \vec{J_i} \\ &= \widehat{D}\ \frac{1}{D}\left(\frac{\partial H_z}{\partial \phi} - \frac{\partial (D.H_\phi)}{\partial z}\right) - \widehat{\phi}\ \left(\frac{\partial H_z}{\partial D} - \frac{\partial H_D}{\partial z}\right) + \widehat{z}\ \frac{1}{D}\left(\frac{\partial (D.H_\phi)}{\partial D} - \frac{\partial H_D}{\partial \phi}\right)\end{aligned} \tag{6.3}$$

Grouping the common vector components of Equations 6.2 and 6.3 produces the 6 identities shown in Equation 6.4.

$$\begin{aligned}
\epsilon\frac{\partial E_D}{\partial t} &= \left(\frac{1}{D}\frac{\partial H_z}{\partial \phi} - \frac{\partial H_\phi}{\partial z}\right) - \sigma_D^e E_D - J_{iD} & \frac{\partial H_D}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_\phi}{\partial z} - \frac{1}{D}\frac{\partial E_z}{\partial \phi}\right) \\
\epsilon\frac{\partial E_\phi}{\partial t} &= \left(\frac{\partial H_D}{\partial z} - \frac{\partial H_z}{\partial D}\right) - \sigma_\phi^e E_\phi - J_{i\phi} & \frac{\partial H_\phi}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_z}{\partial D} - \frac{\partial E_D}{\partial z}\right) \\
\epsilon\frac{\partial E_z}{\partial t} &= \frac{1}{D}\left(\frac{\partial (D.H_\phi)}{\partial D} - \frac{\partial H_D}{\partial \phi}\right) - \sigma_z^e E_z - J_{iz} & \frac{\partial H_z}{\partial t} &= \frac{1}{D.\mu}\left(\frac{\partial E_D}{\partial \phi} - \frac{\partial (D.E_\phi)}{\partial D}\right)
\end{aligned} \tag{6.4}$$

These 6 identities are simplified by assuming that the area of interest (simulation space) will consist of free space, and a PEC ground plane. Therefore all σ terms will equal zero. A further simplification is made by assuming that the source (lightning current) is only directed along the $\widehat{z}$ direction. Therefore J_{iD} and $J_{i\phi}$ are also zero. The last simplification is achieved using Equation 6.1 which states that field differentials in the ϕ direction are zero. After applying these simplifications the 6 identities are grouped into two mutually independent sets of equations shown in Equations 6.5 and 6.6.

$$\frac{\partial E_D}{\partial t} = \frac{1}{\epsilon}\left(-\frac{\partial H_\phi}{\partial z}\right) \tag{6.5a}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{D.\epsilon}\left(\frac{\partial (D.H_\phi)}{\partial D}\right) - \frac{1}{\epsilon}J_{iz} \tag{6.5b}$$

$$\frac{\partial H_\phi}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_z}{\partial D} - \frac{\partial E_D}{\partial z}\right) \tag{6.5c}$$

$$\frac{\partial H_D}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_\phi}{\partial z}\right) \tag{6.6a}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{D.\mu}\left(-\frac{\partial (D.E_\phi)}{\partial D}\right) \tag{6.6b}$$

$$\frac{\partial E_\phi}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_D}{\partial z} - \frac{\partial H_z}{\partial D}\right) \tag{6.6c}$$

Equations 6.5 are defined as the Transverse Electric (TE) mode equations because the electric fields propagate across the plane [48]. The same is true for Equations 6.6 which are defined as being Transverse Magnetic (TM) mode equations. It is important to re-emphasize that the TE and TM mode equations are completely uncoupled, and do not share any common fields. Therefore these two equation sets can be evaluated separately, as compared to the 3D Cartesian implementation where all the field equations are linked. Another observation is that the TM mode equations have no source components "J". Making the assumption that all field components start with a zero value, the TM mode field will not change without a source. Therefore the TM mode equations will remain zero, and only the TE mode equations need to be evaluated to describe the fields of the Cylindrical system.

Equations 6.7 to 6.9 show the discrete versions of the TE mode difference Equations 6.5. The process of converting from differential to difference equations is the same as seen in Equations 5.7 of Section 5.2.1. These are the update equations of the system, and form the foundation for implementing the FDTD algorithm. Equations 6.7 to 6.9 are more intuitively understood when studied in conjunction with the Yee cell in Figure 6.2 and the example simulation space is shown in Figure 6.5. The update equations follow a similar pattern, where the new field value is equal to the old field value plus the fields surrounding it.

$$E_D\Big|_{(i+\frac{1}{2},k)}^{n+1} = E_D\Big|_{(i+\frac{1}{2},k)}^{n} + \frac{\Delta t}{\epsilon}\left(-1.\frac{H_\phi\Big|_{(i+\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} - H_\phi\Big|_{(i+\frac{1}{2},k-\frac{1}{2})}^{n+\frac{1}{2}}}{\Delta z}\right) \tag{6.7}$$

$$E_z\Big|_{(i,k+\frac{1}{2})}^{n+1} = E_z\Big|_{(i,k+\frac{1}{2})}^{n} + \frac{\Delta t}{\epsilon}\left(\frac{D_{(i+\frac{1}{2})}.H_\phi\Big|_{(i+\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} - D_{(i-\frac{1}{2})}.H_\phi\Big|_{(i-\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}}}{D_{(i)}.\Delta D}\right) - \frac{\Delta t}{\epsilon}J_{iz} \tag{6.8}$$

$$\begin{aligned} H_\phi\Big|_{(i+\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} =& H_\phi\Big|_{(i+\frac{1}{2},k+\frac{1}{2})}^{n-\frac{1}{2}} \\ &+ \frac{\Delta t}{\mu}\left(\frac{E_z\Big|_{(i+1,k+\frac{1}{2})}^{n} - E_z\Big|_{(i,k+\frac{1}{2})}^{n}}{\Delta D} - \frac{E_D\Big|_{(i+\frac{1}{2},k+1)}^{n} - E_D\Big|_{(i+\frac{1}{2},k)}^{n}}{\Delta z}\right) \end{aligned} \tag{6.9}$$

The notation used in these equations are explained in Equation 6.10. The $\widehat{\phi}$ component has been completely removed from the update equations. The μ and ϵ terms are spatially located at the same position as the fields on the LHS of the update equations, however in a homogeneous isotropic medium these terms remain constant throughout the plane.

$$F\Big|_{(i,k)}^{n} = F(i,k,n) \rightarrow F(i.\Delta D, k.\Delta z, n.\Delta t) = F(D,z,t) \tag{6.10}$$

The only unique change in these update equations is the inclusion of the $D_{(i)}$ terms in Equation 6.8. The D (radial distance) term is part of the differential form, and therefore needs to be discretized onto the discrete spatial grid of the FDTD plane where:

$$D_{(i)} = i.\Delta D \tag{6.11}$$

The update equations can be evaluated throughout the plane with exception to the lightning channel boundary (where $i = 0$). As seen in Equation 6.8, when $i = 0$ there is a division by 0, and the H_ϕ fields cannot be defined on the 2D plane (as seen in Figure 6.5). Therefore a separate update equation is needed for the E_z fields on the lightning channel.

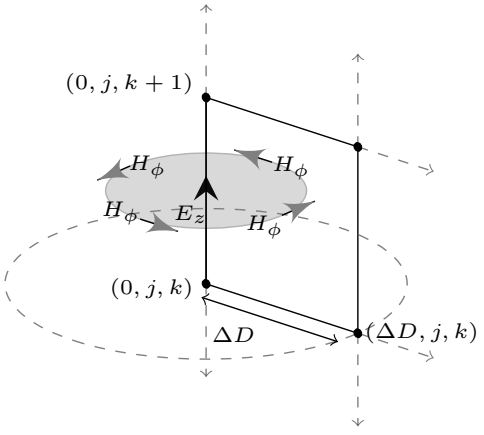


Figure 6.4: E_z field on Lightning Channel

Figure 6.4 shows a single 2D Yee Cell (from Figure 6.5) with a E_z field located on the lightning channel. Due to the axial symmetry the H_ϕ field, located at $\frac{\Delta D}{2}$ radial distance, is assumed to be constant along the contour of the of the grey surface. A further assumption is that the E_z field is constant across the surface of the grey area. Equations 6.12 show how these assumptions are applied to Ampere's Law (Equation 3.17) without the current density component.

$$\oint_c \vec{H} \cdot dl = \int_s \frac{\partial \vec{D}}{\partial t} \cdot ds \tag{6.12a}$$

$$\oint_c H_\phi\Big|_{(\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} \cdot dl = \int_s \epsilon \frac{\partial E_z}{\partial t}\Big|_{(0,k+\frac{1}{2})}^{n+\frac{1}{2}} \cdot ds \tag{6.12b}$$

$$H_\phi\Big|_{(\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} \cdot \oint_c 1dl = \epsilon \frac{\partial E_z}{\partial t}\Big|_{(0,k+\frac{1}{2})}^{n+\frac{1}{2}} \cdot \int_s 1ds \tag{6.12c}$$

$$H_\phi\Big|_{(\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} \cdot 2\pi(\frac{\Delta D}{2}) = \epsilon \frac{\partial E_z}{\partial t}\Big|_{(0,k+\frac{1}{2})}^{n+\frac{1}{2}} \cdot \pi(\frac{\Delta D}{2})^2 \tag{6.12d}$$

After discretizing the time differential in Equation 6.12d, and rearranging terms, the update equation for the lightning channel is defined in Equation 6.13. This equation still needs the addition of the lightning current density term "$\frac{\Delta t}{\epsilon}J$", which can be removed from Equation 6.8 for the remaining simulation space.

$$E_z\Big|_{(0,k+\frac{1}{2})}^{n+1} = E_z\Big|_{(0,k+\frac{1}{2})}^{n} + \frac{4\Delta t}{\epsilon \Delta D} H_\phi\Big|_{(\frac{1}{2},k+\frac{1}{2})}^{n+\frac{1}{2}} \tag{6.13}$$

6.2.2 Stability and Dispersion

The CFL stability criteria for the three dimensional Cylindrical Yee cell is seen in Equation 6.14.

$$\Delta t \leq \frac{1}{v_c \sqrt{\frac{1}{\Delta D^2} + \frac{1}{(D.\Delta\phi)^2} + \frac{1}{\Delta z^2}}} \tag{6.14}$$

For the two dimensional cylindrical Yee cells shown in Figure 6.5, the CFL stability criteria is achieved by removing the $\Delta\phi$ term, as seen in Equation 6.15.

$$\Delta t \leq \frac{1}{v_c \sqrt{\frac{1}{\Delta D^2} + \frac{1}{\Delta z^2}}} \tag{6.15}$$

The optimal time step (Δt_{opt}) is achieved by setting Δt equal to the right hand side of Equation 6.15, as was done in Chapter 5. However from testing it was found that Δt_{opt} caused instability in the plane, and therefore the time step was set to 90% of Δt_{opt}. The reason for this difference is unclear, however it is not the purpose of this work to investigate the numerical effects of the FDTD method, but rather to present its implementation with respect to lightning simulations.

In selecting a smaller Δt the system remained stable (property of the FDTD algorithm), however may suffer from higher dispersion effects. This is unfortunately a necessary trade off, and as seen in the Simulations section the effects are negligible.

6.2.3 Boundary Conditions

Figure 6.5 shows an example 2D FDTD simulation space with the lightning channel located on the LHS of the plane, and a PEC ground plane on the bottom of the plane. As with the 3D Cartesian FDTD implementation, the Yee Cells were chosen such that the Electric fields are located in the Lightning channel and the ground plane. In doing so it is simple to add the lightning RS current density to the FDTD update equations, as well as force a zero electric field on the ground plane (PEC properties).

This implementation simplifies the four boundary conditions of the plane. For the lightning channel edge, the E_z field is updated using Equation 6.13. Assuming the ground plane is a PEC, then it is a requirement that E_D field remains zero. Therefore E_D does not require an update equation.

The remaining boundary walls are the RHS and top edges of the space. It is possible to implement unique boundary conditions (update equations) for the E_z and E_D fields, as was done in Chapter 5. In this example the H_ϕ fields are extended by an additional cell dimension such that the H_ϕ field forms the boundary wall. This was done so that the top and right boundaries would share a common format, as well as to simplify the corner conditions.

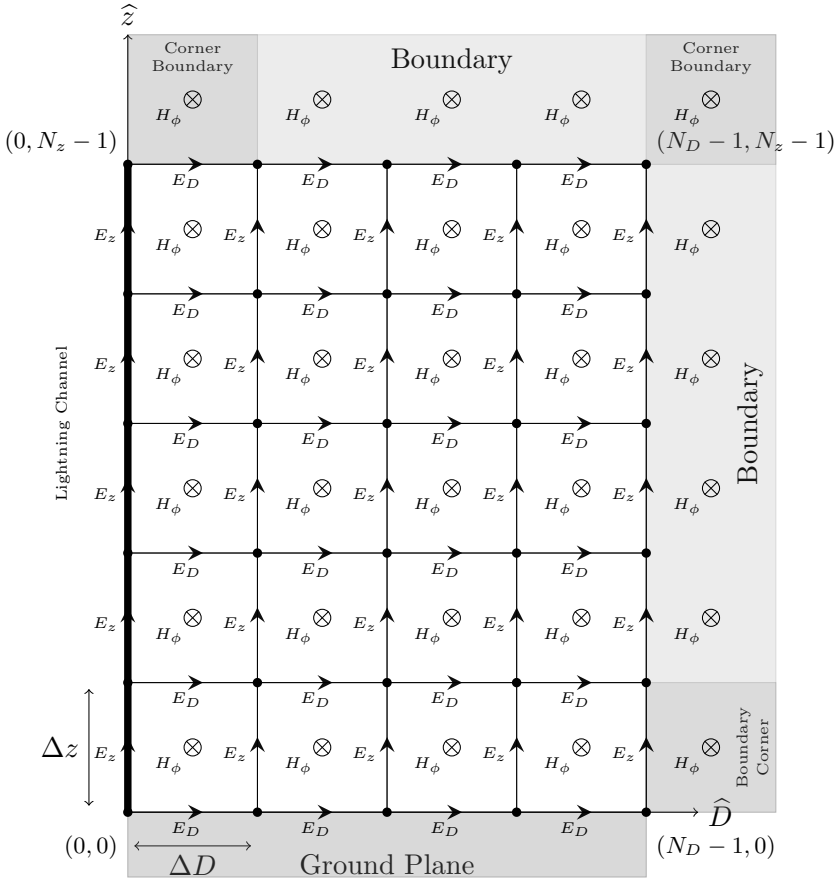


Figure 6.5: EM Fields assigned to 2D FDTD Simulation Space

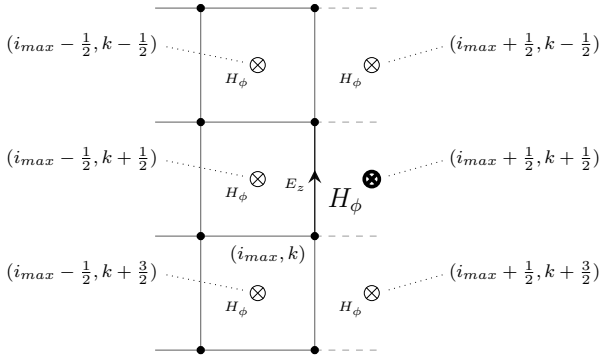


Figure 6.6: Cell Requirements for H_ϕ at $\widehat{D}$ Upper Boundary (2nd Order Mur)

Equations 6.16 and 6.17 show the 2nd order Mur update equations for the length of the right and upper H_ϕ boundary conditions. Figure 6.6 shows an arbitrary H_ϕ field (in bold) located on the right hand boundary with the surrounding H_ϕ fields required for the update Equation 6.16. The fields along the top boundary have similar cell requirements for the upper boundary condition.

$$
\begin{aligned}
H_\phi\Big|^{n+1}_{(i_{max}+\frac{1}{2},k+\frac{1}{2})} = \\
&- H_\phi\Big|^{n-1}_{(i_{max}-\frac{1}{2},k+\frac{1}{2})} - \left(\frac{\Delta D - v_p\Delta t}{\Delta D + v_p\Delta t}\right)\left(H_\phi\Big|^{n+1}_{(i_{max}-\frac{1}{2},k+\frac{1}{2})} + H_\phi\Big|^{n-1}_{(i_{max}+\frac{1}{2},k+\frac{1}{2})}\right) \\
&+ \left(\frac{2\Delta D}{\Delta D + v_p\Delta t}\right)\left(H_\phi\Big|^{n}_{(i_{max}+\frac{1}{2},k+\frac{1}{2})} + H_\phi\Big|^{n}_{(i_{max}-\frac{1}{2},k+\frac{1}{2})}\right) \\
&+ \left(\frac{\Delta D(v_p\Delta t)^2}{2(\Delta z)^2(\Delta D + v_p\Delta t)}\right)\left(H_\phi\Big|^{n}_{(i_{max}+\frac{1}{2},k+\frac{3}{2})} - 2H_\phi\Big|^{n}_{(i_{max}+\frac{1}{2},k+\frac{1}{2})} + H_\phi\Big|^{n}_{(i_{max}+\frac{1}{2},k-\frac{1}{2})}\right. \\
&\qquad \left. + H_\phi\Big|^{n}_{(i_{max}-\frac{1}{2},k+\frac{3}{2})} - 2H_\phi\Big|^{n}_{(i_{max}-\frac{1}{2},k+\frac{1}{2})} + H_\phi\Big|^{n}_{(i_{max}-\frac{1}{2},k-\frac{1}{2})}\right)
\end{aligned}
\tag{6.16}
$$

$$
\begin{aligned}
H_\phi\Big|^{n+1}_{(i+\frac{1}{2},k_{max}+\frac{1}{2})} = \\
&- H_\phi\Big|^{n-1}_{(i+\frac{1}{2},k_{max}-\frac{1}{2})} - \left(\frac{\Delta z - v_p\Delta t}{\Delta z + v_p\Delta t}\right)\left(H_\phi\Big|^{n+1}_{(i+\frac{1}{2},k_{max}-\frac{1}{2})} + H_\phi\Big|^{n-1}_{(i+\frac{1}{2},k_{max}+\frac{1}{2})}\right) \\
&+ \left(\frac{2\Delta z}{\Delta z + v_p\Delta t}\right)\left(H_\phi\Big|^{n}_{(i+\frac{1}{2},k_{max}+\frac{1}{2})} + H_\phi\Big|^{n}_{(i+\frac{1}{2},k_{max}-\frac{1}{2})}\right) \\
&+ \left(\frac{\Delta z(v_p\Delta t)^2}{2(\Delta D)^2(\Delta z + v_p\Delta t)}\right)\left(H_\phi\Big|^{n}_{(i+\frac{3}{2},k_{max}+\frac{1}{2})} - 2H_\phi\Big|^{n}_{(i+\frac{1}{2},k_{max}+\frac{1}{2})} + H_\phi\Big|^{n}_{(i-\frac{1}{2},k_{max}+\frac{1}{2})}\right. \\
&\qquad \left. + H_\phi\Big|^{n}_{(i+\frac{3}{2},k_{max}-\frac{1}{2})} - 2H_\phi\Big|^{n}_{(i+\frac{1}{2},k_{max}-\frac{1}{2})} + H_\phi\Big|^{n}_{(i-\frac{1}{2},k_{max}-\frac{1}{2})}\right)
\end{aligned}
\tag{6.17}
$$

Figure 6.6 also shows that the update equation for an arbitary H_ϕ field requires knowledge of the fields above and below the field. The same is true for the upper boundary, and therefore Equations 6.16 and 6.17 cannot be used for the corners of the boundaries.

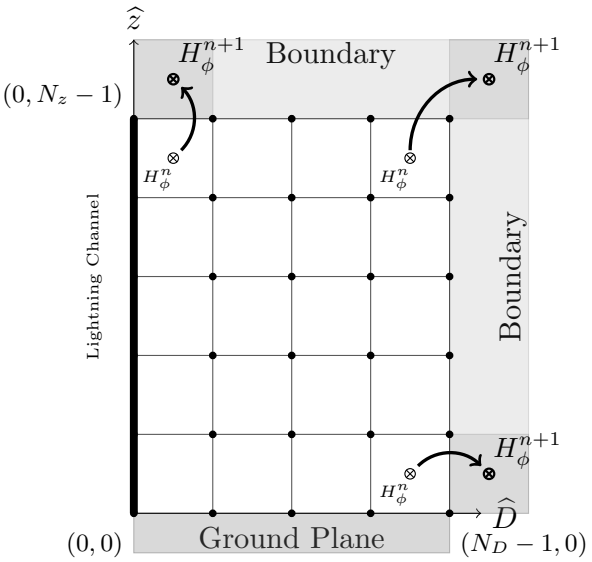


Figure 6.7: Cell Requirements for H_ϕ at Corner Boundary Cells (2nd Order Mur)

$$H_\phi\Big|^{n+1}_{(\frac{1}{2},k_{max}+\frac{1}{2})} = H_\phi\Big|^{n}_{(\frac{1}{2},k_{max}-\frac{1}{2})} \tag{6.18a}$$

$$H_\phi\Big|^{n+1}_{(i_{max}+\frac{1}{2},k_{max}+\frac{1}{2})} = H_\phi\Big|^{n}_{(i_{max}-\frac{1}{2},k_{max}-\frac{1}{2})} \tag{6.18b}$$

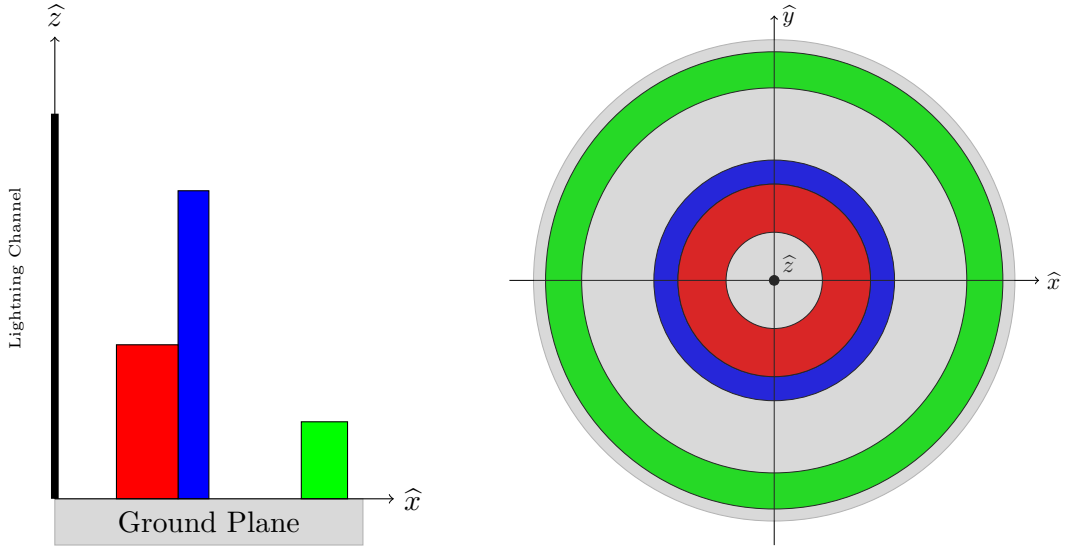
$$H_\phi\Big|^{n+1}_{(i_{max}+\frac{1}{2},\frac{1}{2})} = H_\phi\Big|^{n}_{(i_{max}-\frac{1}{2},\frac{1}{2})} \tag{6.18c}$$

Equations 6.18 show the 1st order Mur update equations for the corner boundary conditions. Figure 6.7 illustrates how these equations operate in the plane. These update equations are not ideal, and would benefit from a higher order Mur implementation, however is sufficient for this work. The 1st order implementation is best suited for fields that impact a boundary directly (perpendicular to the plane), which is why the top left and bottom right cells are aligned with the lightning RS base.

6.2.4 Simulation space objects

The discussion so far has assumed a simple simulation space consisting of a PEC ground plane, and free of objects. This assumption allows the Yee Cell permeability (μ), permittivity (ϵ) and conductivity (σ^e) to be constant throughout the simulation space, which ultimately simplifies the update equations and implementation of the model.

However as was done in the 3D FDTD implementation it is possible to define the properties of each individual Yee Cell in the 2D FDTD plane. In doing so the effects of objects on the EM field propagation can be simulated. The most important object to include is a conductive ground plane [56], but other objects are more difficult to add due to the axial symmetry, as seen in Figures 6.8a and 6.8b. These figures show a ground plane, with three different objects and dimensions. Due to the cylindrical co-ordinate



(a) Side View (b) Top View

Figure 6.8: 2D Cylindrical Simulation Space Objects

system, when an object is added to the 2D plane it is modelling a 3D object that surrounds the lightning channel. Adding objects that model real world environments may be difficult due to the circular requirement of the co-ordinate system. It is however simple to add conductive flat ground planes that model real ground planes.

Another important object to add to the simulation space is the lightning current source. This is done by adding the lightning current density component to the channel update Equation 6.13. Equation 6.19 defines the current density in terms of the lightning return stroke current, and the cell area it is passing through. As seen in Figure 6.4, the cell area is now defined by a circle with a $\frac{\Delta D}{2}$ radius.

$$J_{iz}(z,t) = \frac{I_{RS}(z,t)}{\pi(\frac{\Delta D}{2})^2} \tag{6.19}$$

6.2.5 Computational Considerations

Figure 6.5 shows an example simulation space made up of a 4*5 Yee Cell matrix. This simulation space is also defined by the 5*6 cell nodes in the $\widehat{D}$ and $\widehat{Z}$ directions, where N_D and N_z are the number of nodes. As mentioned before the programming language used in this work is C++, and therefore the array index used begins with "0" and not "1".

Figure 6.5 shows that in the $\widehat{Z}$ direction there are $N_z \times E_D$ fields, but only $(N_z - 1) \times E_z$ fields. The same is true for the $\widehat{D}$ direction with $N_D \times E_z$ fields, but only $(N_D - 1) \times E_D$ fields. This follows the same concepts discussed in the 3D FDTD section, however due to the expansion of the magnetic fields to form the boundary, there are $N_D \times N_z$ magnetic field components in the $\widehat{D}$ and $\widehat{Z}$ directions respectively.

These field matrix dimensions are directly linked to the amount of computer memory required for an FDTD simulation. Equation 6.20 shows the exact and approximate number of memory elements required for all fields in the simulation space.

$$\begin{aligned} N_{mem} &= (NumE_z) + (NumE_D) + (NumH_\phi) \\ N_{mem} &= N_D.(N_z - 1) + (N_D - 1).N_z + N_D.N_z \\ N_{mem} &\simeq 3.N_x.N_y.N_z \end{aligned} \tag{6.20a}$$

Section 5.2.5 discussed a simple example simulation space that is 8000 m high, with a 200 m radial distance away from the lightning channel. The Yee Cell dimensions (in Cartesian co-ordinates) were each set to 1 m, and assuming that a "double" variable type (8 bytes) is used to store each field component, then the simulation would need 57 GB to store the field memory. This memory requirement could possibly double if the 3D simulation space consisted of complex variable objects in the simulated space. However in the 2D FDTD method, a simulation space with the same dimensions and spatial resolution would only need 37 MB. This is 0.06% of the memory required for the equivalent 3D FDTD method, which demonstrates the primary advantage of the 2D FDTD method.

The lower memory requirement allows for a finer spatial resolution in the plane, or a larger simulation space, both of which have important applications for lightning EM simulations. For example consider a problem requiring knowledge of EM fields located at 100 km, with a 8000 m high lightning channel and a 1 m spatial resolution. The approximate memory requirement would be 18 GB, which is achievable with modern computers, however the equivalent model in the 3D FDTD method would be unrealistic.

Example code for this 2D FDTD method has been included in Appendix E, and it follows the same implementation process presented in Figure 5.10.

6.3 Model Simulations

This section will produce some example field waveforms using the method presented above. The example case is the same as those presented in the Finite Antenna method and 3D FDTD method, so that comparisons can be made.

Figure 6.9 shows the 2D FDTD model environment for a basic lightning simulation space [57]. The simulation space is $Z_{dis} * D_{dis}$ m^2 in size. The lightning channel is placed along the left hand boundary, which represents the center of the cylindrical co-ordinate space, and runs the full height of Z_{dis}. The lightning channel current distribution is described by the transmission line (TL) return stroke model (Equation 3.12) with the current impulse described by Equation 3.7 (Terespolsky version of the popular current impulse), and a return stroke speed of $0.5\,c$. The field observation point is defined at point P, and is placed on the lower left hand node of a Yee Cell in the simulation space.

There is a similarity between the 2D FDTD model environment in Figure 6.9, with the 3D FDTD environment in Figure 5.14. In the 3D example case the observation point is placed on the X-Z plane where $y_{Obs} = 0$. This placement simplified the variables in the 3D example, and equated its field vectors with those in the 2D FDTD example.

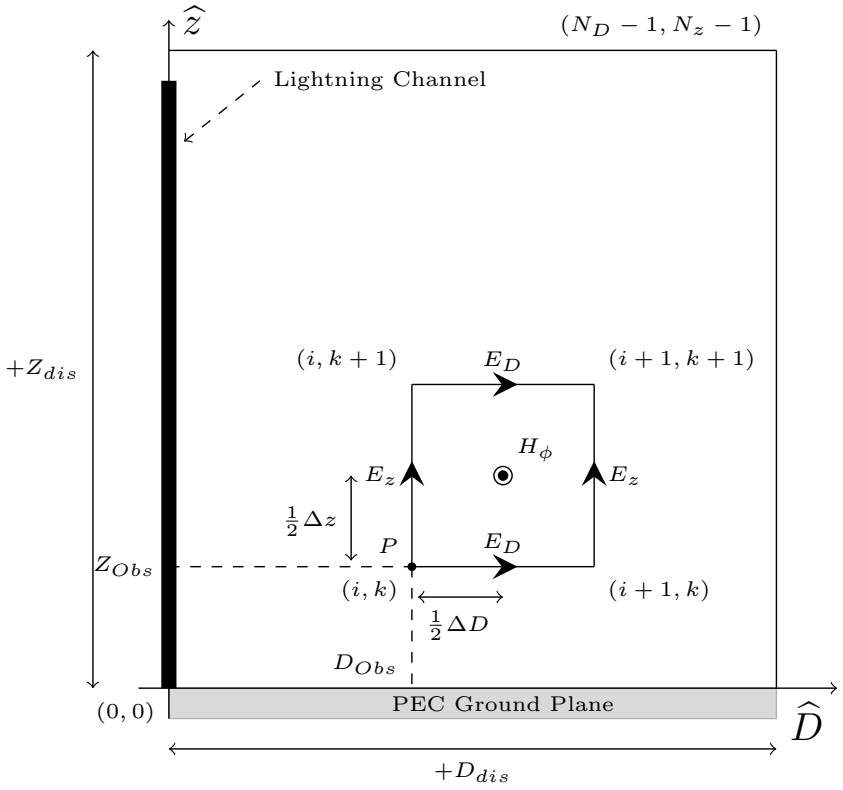


Figure 6.9: Basic 2D Model Space - With Discrete Location Yee Cell Nodes and Fields

Therefore this section will demonstrate that the $\widehat{x}$ and $\widehat{z}$ fields in the 3D example case are equivalent to the $\widehat{D}$ and $\widehat{z}$ fields in the 2D example case.

The ground plane is assumed to be a perfect electrical conductor (PEC). It is possible to add a number of cell layers with $\sigma = 1$, but the same can be achieved by ignoring the boundary conditions along the lower edge ($Z = 0$). This simplifies the process of adding a ground plane object, as well as adding additional matrices to the FDTD algorithm.

The area surrounding the lightning channel is free space where $\mu_r = \mu_0$, $\epsilon_r = \epsilon_0$, $\sigma^e = 0$ and the propagation velocity $V_p = c$ throughout the simulation space. The above assumptions have defined all the simulation variables except the simulation space size, Yee Cell dimensions, and simulation run time. The simulation time is simply the time taken for the lightning EM field to reach the observation point, plus the time period needed to review the field waveform at the observation point.

One method to define the simulation space dimensions is to consider the effects of the boundary plane reflections, which exist even when ABC's are used. With these considerations in mind the simulation space dimensions are defined by the minimum $\widehat{D}$ distance ($+D_{min}$), and minimum $\widehat{z}$ distance ($+Z_{min}$) such that reflections from these boundaries only reach the observation point at the end of the simulation time. This is illustrated in Figure 5.12. Equations 5.28a and 5.28c define the $+D_{min}$ and $+Z_{min}$ dimensions where the $\widehat{x}$ unit vectors are replaced with the equivalent $\widehat{D}$ unit vectors.

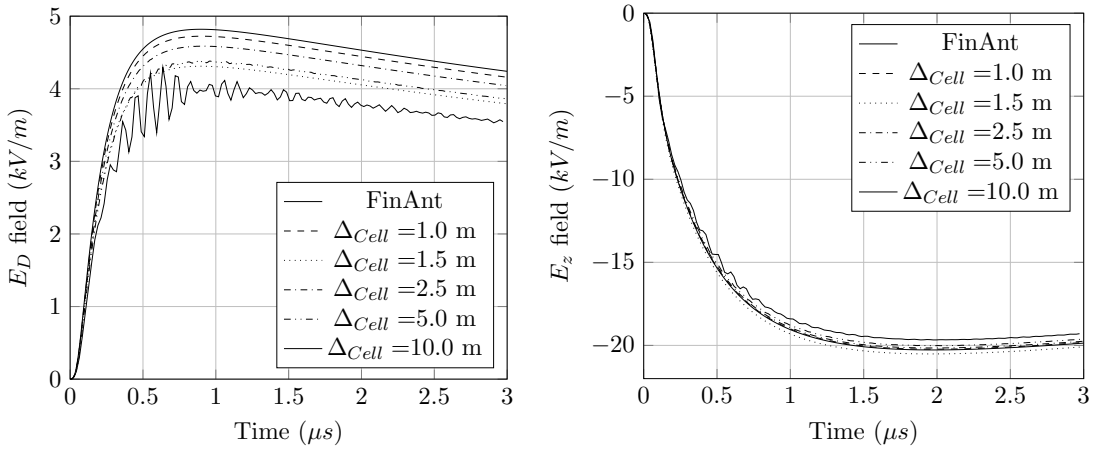


Figure 6.10: 2D FDTD Fields - $D_{Obs} = 50\,\text{m}$, $z_{Obs} = 10\,\text{m}$ - Variable Grid Size

Example Case

The observation point is set to 50 m from the lightning channel, and 10 m off the ground. This is the same observation point that was used in the Finite Antenna and 3D FDTD examples, and aims to evaluate the worst case position of a distribution line with respect to a lightning return stroke channel. The field review time is chosen to be 3 µs, and therefore the simulation time is 3170 ns. Using this observation point and simulation time the D_{min} and Z_{min} are set to 501 m and 479 m respectively.

The Yee Cells are chosen to have a uniform dimension where $\Delta D = \Delta z = \Delta_{cell}$. Figure 6.10 shows the electric fields at the observation point produced by the 2D FDTD method using different Yee Cell dimensions, as well as the electric fields produced by the Finite Antenna method (for comparison). A comparison between the 2D FDTD fields in Figure 6.10 and the 3D FDTD fields in Figure 5.13 shows that the methods are equivalent for this basic simulation.

The fields produced in these figures are discussed in great detain in Section 5.3, however there are two primary concepts that need to be highlighted. The first is that as the Yee Cells are made smaller, there is a finer spatial resolution as well as time step due to the CFL condition (stability). The time step and spatial resolution resulting from the $\Delta_{cell} = 10\,\text{m}$ are too large, and the source startup transient causes numerical noise in the field waveform. The second observation is that the observation point P as well as the fields can only be placed at discrete locations in the simulation space. This is defined by $D_{Obs} = i.\Delta D$ and $z_{Obs} = k.\Delta z$ where i and k are integers. In addition to this it is clear from Figure 6.9 that none of the fields are defined at the same location. This is why the fields converge towards the Finite Antenna fields (located at point P) when the Yee Cell is made smaller. The locations of the fields and observation points in Figure 6.10 are defined in Table 5.1. This explains why the fields from $\Delta_{cell} = 1.5\,\text{m}$ appear to be worse than fields from larger cells.

Both these observations show the importance of keeping the cell dimensions as small as possible. In the 3D FDTD method the simulation time, location of observation point, simulation space dimensions and Yee Cell dimensions are all limited by the amount

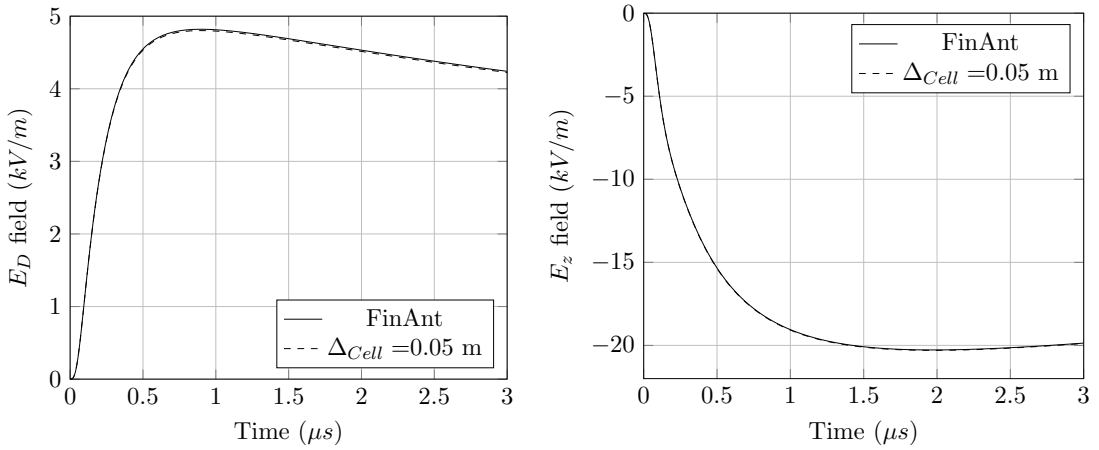


Figure 6.11: 2D FDTD Fields - $D_{Obs} = 50\,\text{m}$, $z_{Obs} = 10\,\text{m}$ - Fine Scale Yee Cells

of computer memory available for the simulation. The 2D FDTD method requires significantly less computer memory, and therefore allows for longer simulation times, further observation points, larger simulation spaces and smaller Yee Cells.

Table 6.1 shows the memory requirements (using a "double" variable type) for the 2D FDTD and 3D FDTD methods with different Yee Cell sizes (Δ_{cell}). This assumes that the observation point, simulation time and simulation space remain unchanged from the example above. This table shows that a simulation with a 5 cm spatial resolution is simple to implement, where as it would be unrealistic in the 3D FDTD method.

Figure 6.11 shows the electric fields at the observation point using the 2D FDTD method and fine scale Yee Cell dimensions, as well as the fields from the Finite Antenna method. This figure shows that although the E_D, E_z and Finite Antenna fields each have separate locations, the difference between the fields at these locations has become negligible. This demonstrates one advantage of the 2D FDTD method over the 3D FDTD method.

Another advantage is that the simulation time can be extended, as seen in Figure 6.12. This figure shows the electric fields for the example case with a simulation time of 10.17 µs, and $\Delta_{Cell} = 0.5\,\text{m}$. This figure also shows the effect of decreasing the simulation space boundary. The percentage value indicates the percentage of D_{min} and Z_{min} used for the simulation space, and the Finite Antenna field at the shifted

Table 6.1: Example Case: Cell Computer Memory Requirements 2D VS 3D

Δ_{cell} (m)	**Effective** (D_{Obs}, z_{Obs})	**2D FDTD Mem**	**3D FDTD Mem**
0.015	(49.995,9.99)	23.8 GB	6.03×10^6 GB
0.05	(50,10)	2.15 GB	163×10^3 GB
0.1	(50,10)	550 MB	20.34×10^3 GB
0.5	(50,10)	22 MB	163 GB
1	(50,10)	5.5 MB	20.43 GB
1.5	(49.5,9)	2.44 MB	6.03 GB
5	(50,10)	0.22 MB	167 MB

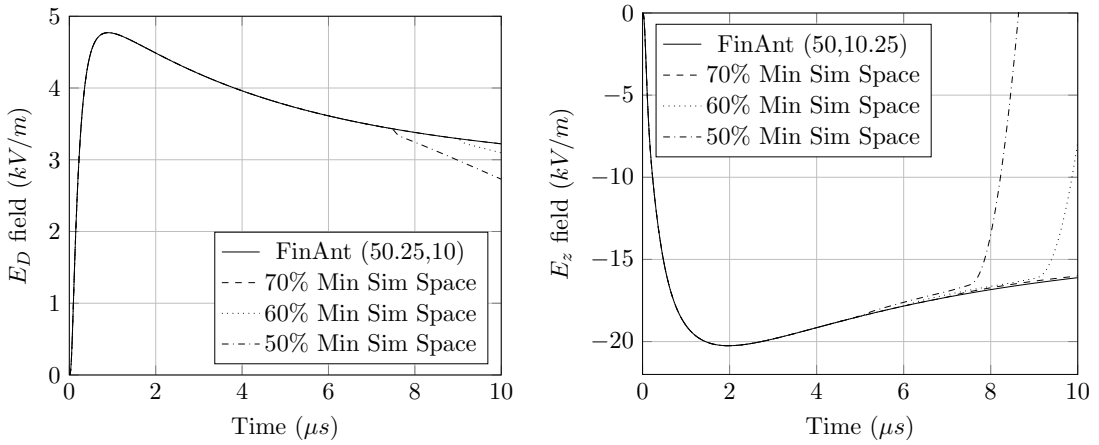


Figure 6.12: 2D FDTD Fields - $D_{Obs} = 50\,\text{m}$, $z_{Obs} = 10\,\text{m}$ - Variable Simulation Space

location shows what the waveform should be. As expected when the simulation space is made smaller the effects of the boundary reflections increase. From Figure 6.12 it is recommended that the simulation space only be reduced to 70% in each dimension. This has the effect of reducing the computer memory requirements by 50% ($0.7 * 0.7$), which allows for improved spatial resolution, as well as a reduced computation time.

This method is also well suited for simulations with observation points far (greater than 1000 m) from the lightning channel, however the approach taken in implementing these simulations is significantly different to the method discussed in this chapter. At far observation points the use of boundary conditions becomes less significant as the space surrounding the observation point (defined by Equations 5.28a to 5.28c) is negligible in respect to the size of the simulation space (consider a point 100 km from the lightning channel). This can also be thought of as the ratio of field observation time (T_{obs}) VS time taken to reach the observation point (T_{travel}). When $T_{Obs} > T_{travel}$ then the methodology of this chapter is applicable.

Another consequence of a far observation point is a long simulation time. Small numerical artefacts that may be negligible in short simulations, tend to grow during long simulations. These problems can be rectified, however as outlined in the introduction, this work is primarily concerned with fields near (50 m to 500 m) the lightning channel. As such the methodology for far observation points will not be discussed in detail.

6.4 Chapter Summary

This chapter has presented a comprehensive approach to applying the 2D FDTD method to lightning EM simulations. This is done by discussing the theory and assumptions required in changing 3D Cylindrical co-ordinates into 2D Cylindrical co-ordinates by taking advantage of axial symmetry around the lightning channel. This axial symmetry around a perpendicular lightning channel is also the limiting factor for the 2D FDTD method, but under simple simulation environments it does not cause problems.

The full mathematical development of the method is presented to aid in creating unique variations of the 2D FDTD method. In addition to this, example code has been included in the appendix to assist readers in understanding how the FDTD update equations, and boundary conditions are implemented in code. This chapter also discusses the importance of simulation space objects, and computational restrictions with regards to lightning based simulations.

An example case has been used to demonstrate how the method is implemented, as well as the effects of various numerical artefacts such as the Yee Cell Size and boundary conditions. The fields produced in these examples show that the 2D FDTD method produces the same results as the 3D FDTD method when a simple simulation space is used. The primary advantages of this method is its ease of implementation, and significantly lower computational requirements. This allows for a finer resolution in the simulation space and longer simulation times. This method is also well suited to simulations with observation points located far from the lightning channel, however the approach taken in such simulations is unique, and not discussed here.

The next chapter will present the Single Cell FDTD method, which is the last of the methods presented in this work. This method is a combination of the Finite Antenna method, and the FDTD method, and uses many of the same assumptions presented in this chapter.

Chapter 7

Single Cell FDTD Method

This chapter presents the Single Cell FDTD method, also known as the hybrid method. This method combines the theory of the Finite Antenna method, and the FDTD method to simulate lightning EM fields, and has the advantage of being more computationally efficient than the other methods presented in this work. The method is however limited to simple simulation environments. This chapter presents the theory of the method using Spherical co-ordinates, and the 2D assumptions used in the 2D FDTD chapter.

7.1 Overview

The standard FDTD method presented in the previous chapters has shown that it is possible to simulate EM fields in space by discretizing the simulation space into smaller spatial cells, as well as evaluating the space in small time increments. In doing so this method allows for highly complex simulation environments to be simulated, such as curved lightning channel paths, multilayer conductive ground planes, ground contours, shielding effects of objects and electrical effects on distribution lines. The primary disadvantage of the 3D FDTD method is that it has a high computer memory requirement, even for simple simulations. If the simulation environment is kept simple then it is possible to use the 2D FDTD method, which has the advantage of simulating straight lightning channels over flat conducting ground planes. The 2D method requires significantly less memory than the 3D method, but still has a significant simulation run time due to the large amounts of computations required by the FDTD method.

As seen in Figure 5.10, both FDTD methods presented have the same evaluation process. This process involves calculating the electric and magnetic fields at every cell throughout the evaluation space for every time step of the simulation. This process can take a long time to run through a simulation, especially if the Yee Cells are made smaller. Although the FDTD process is advantageous due to its ability to provide EM field information at every Yee Cell throughout the space (with only a single simulation), it is often unnecessary to know all the field information. Often a problem will focus on simulating the EM field at a single point in space, such as a lightning detection node. In such cases using the FDTD method is excessive, and the Finite Antenna method may provide a better suited alternative.

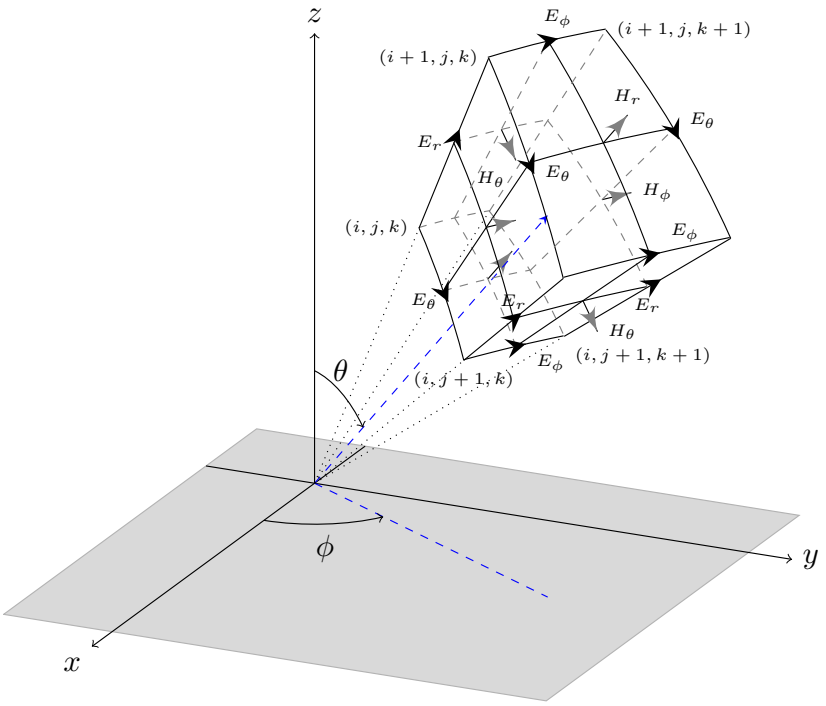


Figure 7.1: 3D Yee Cell - Spherical Co-ordinates - H Field centred

Another alternative is the Single Cell FDTD method, which operates in a similar manner to the Finite Antenna Method, but uses the evaluation process (update equations) of the FDTD method. The update equations in the 3D and 2D FDTD methods show that electric (or magnetic) fields are calculated by their previous time step value, plus the magnetic (or electric) fields that surround the cell of interest. Because this method relies on the fields surrounding the cell of interest, the FDTD process needs every cell in the space to be evaluated for every time step. However if either the electric or magnetic fields are known at the points surrounding the cell of interest, then it is possible to evaluate the opposite field.

This novel solution for LEMP calculations was first suggested by Sartori *et al* [58], and then improved upon by other authors [41, 53, 59, 60]. The fundamental principle of the Single Cell FDTD method uses the magnetic fields calculated by the Finite Antenna method to evaluate the electric fields in the FDTD method. This "hybrid" method [59] is referred to as the "Single Cell FDTD" method, as only one Yee cell is evaluated rather than the entire FDTD plane.

What follows is the mathematical explanation of the method using Spherical co-ordinates. It is important to note that this method will work equally well in other co-ordinate systems, however the magnetic field equations from the Finite Antenna method are already in Spherical co-ordinates, and therefore are convenient to use directly. In addition to this the FDTD mathematics presented here will show the steps involved in creating the update equations in the last of the co-ordinate systems discussed in this work. Figure 7.1 shows a 3D Yee Cell in Spherical co-ordinates.

7.2 Model Development

Faraday's Law (Equation 3.15) and Ampere's Law (Equation 3.17), in combination with Equation 3.25c are used to define the required Maxwell's equations in Spherical coordinates. Equations 7.1 and 7.2 show the expanded differential form of these equations.

$$\begin{aligned}\nabla \times \vec{E} &= \begin{bmatrix} \widehat{r} & \widehat{\theta} & \widehat{\phi} \\ \frac{\partial}{\partial r} & \frac{1}{r}\frac{\partial}{\partial \theta} & \frac{1}{r.\sin\theta}\frac{\partial}{\partial \phi} \\ E_r & E_\theta & E_\phi \end{bmatrix} = \frac{1}{r}\frac{1}{r.\sin\theta}\begin{bmatrix} \widehat{r} & r\widehat{\theta} & r.\sin\theta\widehat{\phi} \\ \frac{\partial}{\partial r} & \frac{\partial}{\partial \theta} & \frac{\partial}{\partial \phi} \\ E_r & r.E_\theta & r.\sin\theta E_\phi \end{bmatrix} = -\frac{\partial \vec{B}}{\partial t} \\ &= \widehat{r}\,\frac{1}{r.\sin\theta}\left(\frac{\partial(\sin\theta E_\phi)}{\partial\theta} - \frac{\partial(E_\theta)}{\partial\phi}\right) - \widehat{\theta}\,\frac{1}{r}\left(\frac{\partial(r.E_\phi)}{\partial r} - \frac{1}{\sin\theta}\frac{\partial E_r}{\partial\phi}\right) \\ &\quad + \widehat{\phi}\,\frac{1}{r}\left(\frac{\partial(r.E_\theta)}{\partial r} - \frac{\partial E_r}{\partial\theta}\right) \end{aligned} \tag{7.1}$$

$$\begin{aligned}\nabla \times \vec{B} &= \begin{bmatrix} \widehat{r} & \widehat{\theta} & \widehat{\phi} \\ \frac{\partial}{\partial r} & \frac{1}{r}\frac{\partial}{\partial \theta} & \frac{1}{r.\sin\theta}\frac{\partial}{\partial \phi} \\ B_r & B_\theta & B_\phi \end{bmatrix} = \frac{1}{r}\frac{1}{r.\sin\theta}\begin{bmatrix} \widehat{r} & r\widehat{\theta} & r.\sin\theta\widehat{\phi} \\ \frac{\partial}{\partial r} & \frac{\partial}{\partial \theta} & \frac{\partial}{\partial \phi} \\ B_r & r.B_\theta & r.\sin\theta B_\phi \end{bmatrix} = \mu\epsilon\frac{\partial \vec{E}}{\partial t} + \mu\sigma^e\vec{E} + \mu\vec{J_i} \\ &= \widehat{r}\,\frac{1}{r.\sin\theta}\left(\frac{\partial(\sin\theta B_\phi)}{\partial\theta} - \frac{\partial(B_\theta)}{\partial\phi}\right) - \widehat{\theta}\,\frac{1}{r}\left(\frac{\partial(r.B_\phi)}{\partial r} - \frac{1}{\sin\theta}\frac{\partial B_r}{\partial\phi}\right) \\ &\quad + \widehat{\phi}\,\frac{1}{r}\left(\frac{\partial(r.B_\theta)}{\partial r} - \frac{\partial B_r}{\partial\theta}\right) \end{aligned} \tag{7.2}$$

Grouping the common vector components of Equations 7.1 and 7.2 produce the 6 identities shown in Equations 7.3.

$$\mu\epsilon\frac{\partial E_r}{\partial t} = \frac{1}{r.\sin\theta}\left(\frac{\partial(\sin\theta B_\phi)}{\partial\theta} - \frac{\partial(B_\theta)}{\partial\phi}\right) - \mu\sigma_r^e E_r - \mu J_{ir} \tag{7.3a}$$

$$\mu\epsilon\frac{\partial E_\theta}{\partial t} = \frac{1}{r}\left(\frac{1}{\sin\theta}\frac{\partial B_r}{\partial\phi} - \frac{\partial(r.B_\phi)}{\partial r}\right) - \mu\sigma_\theta^e E_\theta - \mu J_{i\theta} \tag{7.3b}$$

$$\mu\epsilon\frac{\partial E_\phi}{\partial t} = \frac{1}{r}\left(\frac{\partial(r.B_\theta)}{\partial r} - \frac{\partial B_r}{\partial\theta}\right) - \mu\sigma_\phi^e E_\phi - \mu J_{i\phi} \tag{7.3c}$$

$$-\frac{\partial B_r}{\partial t} = \frac{1}{r.\sin\theta}\left(\frac{\partial(\sin\theta E_\phi)}{\partial\theta} - \frac{\partial(E_\theta)}{\partial\phi}\right) \tag{7.3d}$$

$$-\frac{\partial B_\theta}{\partial t} = \frac{1}{r}\left(\frac{1}{\sin\theta}\frac{\partial E_r}{\partial\phi} - \frac{\partial(r.E_\phi)}{\partial r}\right) \tag{7.3e}$$

$$-\frac{\partial B_\phi}{\partial t} = \frac{1}{r}\left(\frac{\partial(r.E_\theta)}{\partial r} - \frac{\partial E_r}{\partial\theta}\right) \tag{7.3f}$$

These 6 identities can be simplified further by assuming that the simulation space consists of free space with a PEC ground plane. Therefore all σ terms are zero. Further simplifications are made by removing unnecessary source terms. The lightning channel in these models will be perpendicular to the ground plane, directed along the $\widehat{z}$ axis of Figure 7.1. Therefore current terms (J) can only exist along the $\widehat{r}$ unit vector direction (when $\theta = 0$). Therefore $J_{i\theta}$ and $J_{i\phi}$ terms are set to zero.

Assuming that the lightning channel is perpendicular to the ground plane, and that the ground plane is symmetrical surrounding the channel, it is possible to use the same 2D simplifications used in Chapter 6. The most important requirement is that the simulation space is axially symmetric around the lightning channel, and therefore EM fields are ϕ independent ($\frac{d}{d\phi} = 0$). After applying these simplifications the 6 identities of Equations 7.3 are grouped into two mutually independent sets of equations shown in Equations 7.4 and 7.5.

$$\frac{\partial E_r}{\partial t} = \frac{1}{\mu.\epsilon.r.\sin\theta}\left(\frac{\partial(\sin\theta B_\phi)}{\partial\theta}\right) - \frac{1}{\epsilon}J_{ir} \tag{7.4a}$$

$$\frac{\partial E_\theta}{\partial t} = -\frac{1}{\mu.\epsilon.r}\left(\frac{\partial(r.B_\phi)}{\partial r}\right) \tag{7.4b}$$

$$\frac{\partial B_\phi}{\partial t} = -\frac{1}{r}\left(\frac{\partial(r.E_\theta)}{\partial r} - \frac{\partial E_r}{\partial\theta}\right) \tag{7.4c}$$

$$\frac{\partial B_r}{\partial t} = -\frac{1}{r.\sin\theta}\left(\frac{\partial(\sin\theta E_\phi)}{\partial\theta}\right) \tag{7.5a}$$

$$\frac{\partial B_\theta}{\partial t} = \frac{1}{r}\left(\frac{\partial(r.E_\phi)}{\partial r}\right) \tag{7.5b}$$

$$\frac{\partial E_\phi}{\partial t} = \frac{1}{\mu.\epsilon.r}\left(\frac{\partial(r.B_\theta)}{\partial r} - \frac{\partial B_r}{\partial\theta}\right) \tag{7.5c}$$

Equations 7.4 and 7.5 are defined as the Transverse Electric (TE) mode and Transverse Magnetic (TM) mode equations respectively. These equations are similar to those in Chapter 6 (Equations 6.5 and 6.6), however in this context the mode refers to the electric or magnetic fields propagating in the 2D "$\widehat{r} - \widehat{\theta}$" plane. From the assumptions made it is clear that Equations 7.5 have no source terms, and will therefore remain zero. From this only the TE mode equations need to be evaluated.

$$E_r\Big|_{(i+\frac{1}{2},j)}^{n+1} = E_r\Big|_{(i+\frac{1}{2},j)}^{n} + \frac{\Delta t}{\mu.\epsilon.r_{i+\frac{1}{2}}.\sin\theta_j.\Delta\theta}\cdot \left[B_\phi\Big|_{(i+\frac{1}{2},j+\frac{1}{2})}^{n+\frac{1}{2}}.\sin\theta_{j+\frac{1}{2}} - B_\phi\Big|_{(i+\frac{1}{2},j-\frac{1}{2})}^{n+\frac{1}{2}}.\sin\theta_{j-\frac{1}{2}}\right] \tag{7.6}$$

$$E_\theta\Big|_{(i,j+\frac{1}{2})}^{n+1} = E_\theta\Big|_{(i,j+\frac{1}{2})}^{n} - \frac{\Delta t}{\mu.\epsilon.r_i.\Delta r}\cdot\left[B_\phi\Big|_{(i+\frac{1}{2},j+\frac{1}{2})}^{n+\frac{1}{2}}.r_{i+\frac{1}{2}} - B_\phi\Big|_{(i-\frac{1}{2},j+\frac{1}{2})}^{n+\frac{1}{2}}.r_{i-\frac{1}{2}}\right] \tag{7.7}$$

Equations 7.6 and 7.7 show the discrete (spatial and temporal) versions of the TE mode difference Equations 7.4a and 7.4b. The process of converting from differential to difference equations is the same as seen in Equations 5.7 of Section 5.2.1. These equations also assume that μ and ϵ are spatially independent through the simulation space, and that the cell under review is located off the lightning channel ($J_{ir} = 0$). Equations 7.6 and 7.7, in combination with the difference form of Equation 7.4c, could be used to implement a full 2D (Spherical) FDTD simulation, however that is not necessary for this method. These difference equations, as well as the Single Cell FDTD

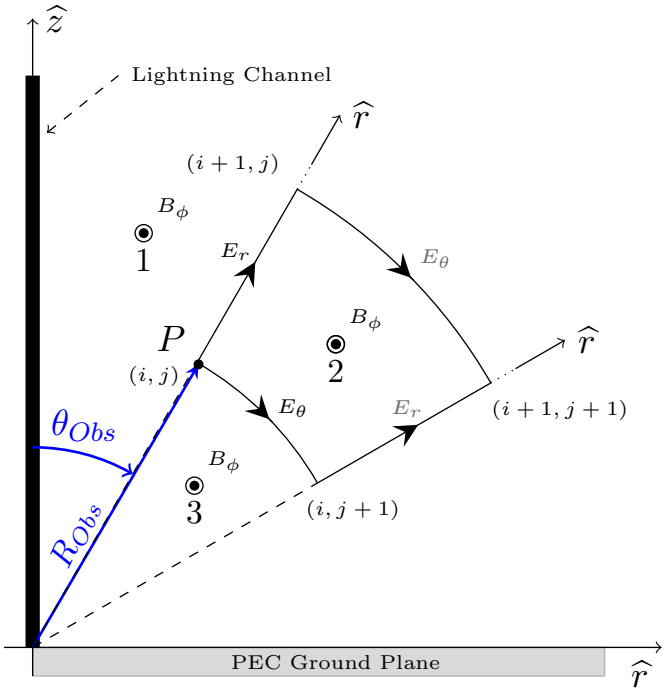


Figure 7.2: 2D Yee Cell - Spherical Co-ordinates - $\widehat{r}-\widehat{\theta}$ plane

method are best understood by reviewing Figure 7.2.

The notation used in Equations 7.6 and 7.7, as well as the figure, are explained in Equation 7.8. These equations also have spatially variant r and θ terms which are explained in Equations 7.9 and 7.10.

$$F\Big|_{(i,j)}^{n} = F(i,j,n) \rightarrow F(i.\Delta r, j.\Delta\theta, n.\Delta t) = F(r,\theta,t) \tag{7.8}$$

$$r_i = i.\Delta r \tag{7.9}$$

$$\sin\theta_j = \sin\left(j.\Delta\theta\right) \tag{7.10}$$

Figure 7.2 shows a single Yee Cell in the 2D Spherical plane. The observation point (P) is located on the lower left hand corner of the cell. As with the other FDTD cell orientations, it is clear that each field component is located at its own unique spatial location (no fields overlap). In order to calculate E_r and E_θ (closest to observation point P), it is clear that the B_ϕ field only needs to be calculated at three locations (numbered in the figure). These three B_ϕ locations are defined by Equations 7.6 and 7.7 which share a common B_ϕ term. The equation for calculating the B_ϕ terms is taken from the Finite Antenna Method in Chapter 4, and is rewritten in Equation 7.11.

This is the fundamental principle of the Single Cell FDTD method. By evaluating the B_ϕ fields at three unique locations it is possible to evaluate the electric fields surrounding the observation point of interest. This method is simple to implement, and does not suffer from the same disadvantages of full FDTD implementations [53]. Only the fields in the cell of interest are evaluated, rather than the full FDTD simulation space. There are no boundary plane limitations, no stability or dispersion issues, and no complex creation

of simulation space objects (such as sources). This is, however, only an advantage for simple simulation environments.

$$\vec{B}(r,\theta,t)\ \widehat{\phi} = \vec{B}_{\phi}(r,\theta,t) =$$

$$+ \frac{1}{4\pi\epsilon_0 c^2}\int_0^{L'(t)} \frac{\sin\alpha(z)}{R^2(z)}\left\{ i_{RS}(0, t - \frac{R(z)}{c} - \frac{z}{v_c})\right\} dz\ \widehat{\phi} \tag{7.11a}$$

$$+ \frac{1}{4\pi\epsilon_0 c^2}\int_0^{L''(t)} \frac{\sin\alpha(-z)}{R^2(-z)}\left\{ i_{RS}(0, t - \frac{R(-z)}{c} - \frac{z}{v_c})\right\} dz\ \widehat{\phi} \tag{7.11b}$$

$$+ \frac{1}{4\pi\epsilon_0 c^2}\int_0^{L'(t)} \frac{\sin\alpha(z)}{cR(z)}\left\{ \frac{\partial i_{RS}(0, t - \frac{R(z)}{c} - \frac{z}{v_c})}{\partial t}\right\} dz\ \widehat{\phi} \tag{7.11c}$$

$$+ \frac{1}{4\pi\epsilon_0 c^2}\int_0^{L''(t)} \frac{\sin\alpha(-z)}{cR(-z)}\left\{ \frac{\partial i_{RS}(0, t - \frac{R(-z)}{c} - \frac{z}{v_c})}{\partial t}\right\} dz\ \widehat{\phi} \tag{7.11d}$$

Initially this method may appear redundant given the similarities to the Finite Antenna Method, however it has one significant advantage. As mentioned in Chapter 4, the Finite Antenna electric field equations contain a term with the integral of the return stroke current (Electrostatic field component). This method relies solely on the magnetic field equation, which does not contain this term and therefore removes unnecessary complexity from the evaluation process [53]. This also allows additional flexibility in the current impulse model used for the RS model [59].

The functionality of this method is extended further by using alternative methods of evaluating the magnetic field. It is possible to use the Wavetilt approximation, and the Rubenstein approximation (discussed in [59]) to evaluate the magnetic fields above a conducting ground plane. If these magnetic fields are used in the Single Cell FDTD update equations, then the resulting electric fields will also be affected by the conducting ground plane [41]. This method has not been investigated or implemented in this work.

7.3 Model Simulations

The simulations used in this chapter follow the same model assumptions as the previous chapters. The lightning channel is assumed straight and perpendicular to a flat perfectly conducting ground plane. The area surrounding the lightning channel is free space (with no obstacles) where $\mu_r = \mu_0$, $\epsilon_r = \epsilon_0$, $\sigma^e = 0$ and the propagation velocity $V_p = c$. The simulation space is illustrated in Figure 7.2.

The lightning channel current distribution is described by the transmission line (TL) return stroke model (Equation 3.12) with the current impulse described by Equation 3.7 (Terespolsky version of the popular current impulse), and a return stroke speed of $0.5\,c$. However for the Single Cell FDTD method and Equation 7.11 it is easier to implement Equation 3.6 (Heidler version of the popular current impulse), but for consistency with the other methods the current impulse model has been kept the same.

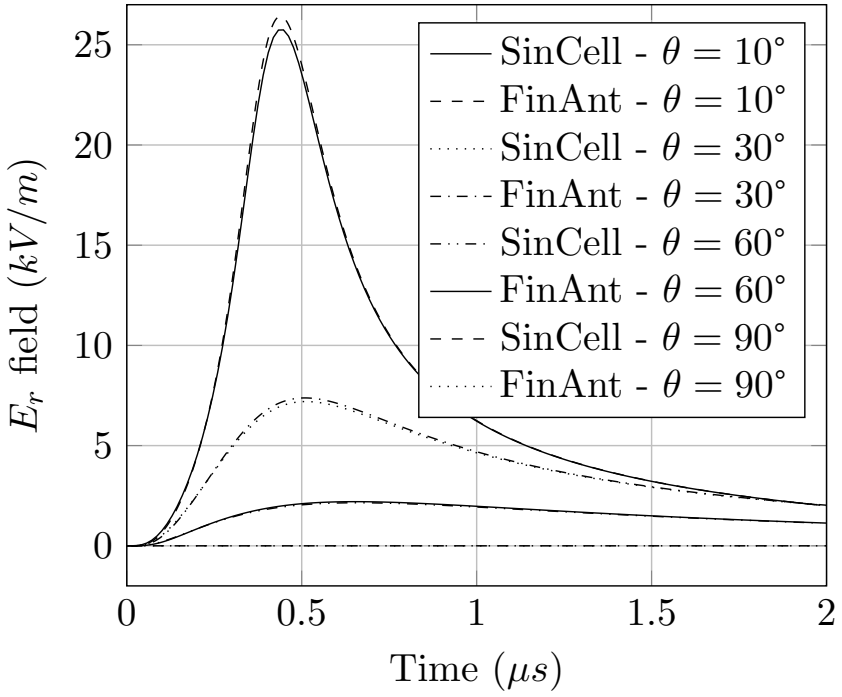


Figure 7.3: Single Cell E_r Field - $R_{Obs} = 100\,\text{m}$, Δr =0.5 m, $\Delta\theta = 0.5°$ - Different θ_{Obs}.

A full FDTD implementation in this co-ordinate system would have limitations on the location of the observation point, however the Single Cell FDTD does not. P can be placed anywhere in the $\widehat{r} - \widehat{\theta}$ plane, and is not limited by Δr, $\Delta\theta$ or Δt.

This method is also not affected by the stability criteria of a normal FDTD implementation, and has freedom in the selection of cell size (Δr and $\Delta\theta$) as well as the time step (Δt). For consistency in methodology this section will use the optimal CFL time step, which is described by Equation 7.12 for the Yee Cell in Figure 7.2.

$$\Delta t \leq \frac{1}{c\sqrt{\frac{1}{(\Delta r)^2} + \frac{1}{(r.\Delta\theta)^2}}} \tag{7.12}$$

Figure 7.3 shows the E_r fields calculated at a radial distance of 100 m, and a range of vertical offset angles. The Yee Cell used $\Delta r = 0.5\,\text{m}$ and $\Delta\theta = 0.5°$. This figure shows that the fields from the Single Cell FDTD method compare well to fields produced by the Finite Antenna Method. These fields can also be compared to the work in [50].

Figure 7.3 shows a difference between the methods, which is clearer in Figure 7.4. This figure shows the E_r field of the Yee Cell with the observation point set at $R_{Obs} = 100\,\text{m}$ and vertical offset angle $\theta_{Obs} = 10°$. With the fixed observation point the Yee Cell dimensions are then varied. As the Yee Cell is increased the difference between the Finite Antenna field at the observation point increases. This effect is caused by the difference in Yee Cell field locations (Figure 7.2), and the location of the observation point. If the Finite Antenna field locations are shifted to those on the Yee Cell, then the fields are the same. However from experience it is best to keep the angular step

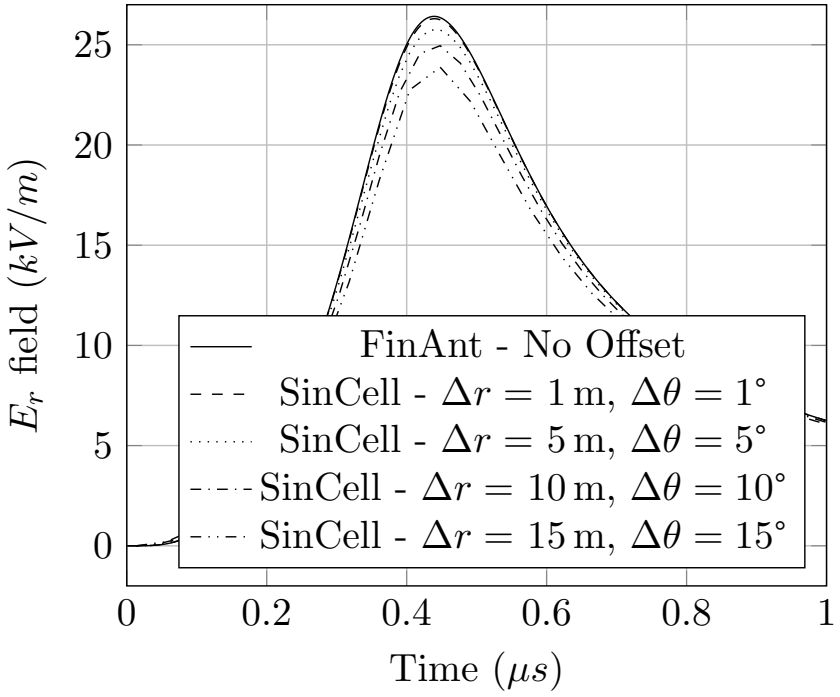


Figure 7.4: Single Cell E_r Field - $R_{Obs} = 100\,\text{m}$, $\theta_{Obs} = 10°$ - Different Cell Size.

($\Delta\theta$) as small as possible. As seen in Figure 7.2 the Yee Cell is not uniform throughout the plane, and will increase in size as the radial distance increases. Therefore at large observation distances even small $\Delta\theta$ values can have large Yee Cells.

Figure 7.5 shows the E_r field at the same location ($R_{Obs} = 100\,\text{m}$ and $\theta_{Obs} = 10°$), with a fixed Yee Cell ($\Delta r = 1\,\text{m}$, $\Delta\theta = 1°$), however the time step Δt is varied. The optimal time step for a full FDTD simulation would require a value of 2.9 µs or less, otherwise the simulation would become unstable. However as seen in this figure, even time steps far greater than the optimal CFL value still produce a stable simulated field.

7.4 Chapter Summary

The Single Cell FDTD method has been presented using the Spherical co-ordinate system in the 2D $\widehat{r} - \widehat{\theta}$ plane. The mathematical derivation of the method has enough detail to allow a full FDTD implementation in the spherical co-ordinate system (which is rarely documented). Example fields have been provided to demonstrate the usage of the method. The code used to produce these fields has been included in Appendix F, and can be adapted to suit other needs. In addition to this a conference paper on this method has also been included in Appendix B.

This alternative approach to modelling lightning EM fields draws on the benefits from both the FDTD and Finite Antenna methods. The FDTD component of the method offers a simple approach to solving the electric fields, without all the numerical consequences of a full FDTD implementation. A full FDTD implementation often requires

large amounts of computer memory to store the fields in the simulation space, however this method only requires memory for the single cell being evaluated. This also reduces computational time due to the fact that the field is only solved for the location of interest, rather than every cell in the simulation space. In addition to this the Single Cell FDTD method does not suffer from stability issues, and does not require boundary conditions.

The benefit of its simplicity is also a limitation on its range of application. This method is bound by the same simulation restrictions as the Finite Antenna method due to the use of the magnetic field equation. However alternative magnetic field equations, such as those that consider the effects of a conducting ground plane, can be used to increase the range of application for this method. For complex simulation environments it is best to implement a full FDTD simulation. In comparison to the Finite Antenna method, the numerical approach to solving the electric fields are simpler due to the fact that the magnetic field equation does not contain the Electrostatic field component. Additionally this allows for more freedom in choosing a current impulse model to suit the simulation.

The next chapter will discuss the outcomes of the presented methods, as well as the potential for future work.

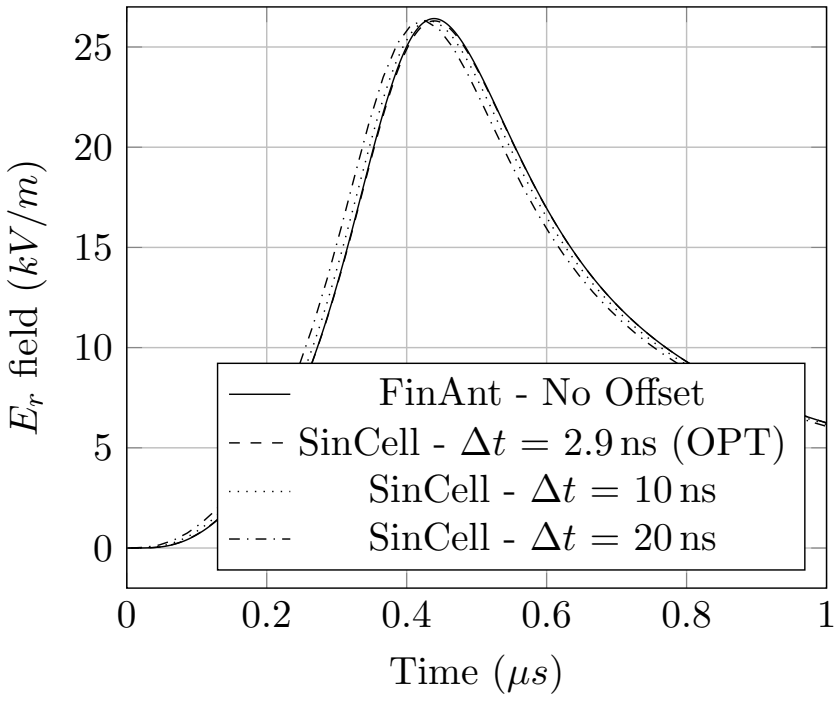


Figure 7.5: Single Cell E_r Field - $R_{Obs} = 100\,\text{m}$, $\theta_{Obs} = 10°$ - Different Time Step.

Chapter 8

Discussion and Future Work

Four methods of simulating lightning return stroke EM fields have been presented using time domain techniques. It has been established that the EM fields that radiate from a lightning return stroke current have the potential to couple onto electrical networks and are responsible for LIOVs that damage the electrical components on the network. Using the methods presented in this work it becomes possible to simulate the expected lightning EM fields, and expand the potential of future research into lightning EM protection.

The methods used in modelling the lightning EM fields have been limited to time domain techniques, due to the transient nature of the lightning current impulse waveform. This is in contrast to standard EM modelling techniques which use frequency domain methods, and are best suited to periodic sources. After identifying the time domain as the best approach for the lightning field analysis, the problem of how the lightning EM fields are modelled was simplified into two primary components. The first is how the lightning EM source is modelled, and the second is how the EM fields are evaluated in the area surrounding the lightning channel.

The first component relies on the physical properties of the lightning return stroke current. The models used to describe the source of the EM field are directly linked to the time variation of the current distribution within the lightning channel. These models are developed by making a number of assumptions that may not represent the real world event accurately, but can be generalised to most lightning events. One such assumption is that the lightning channel is straight and perpendicular to the ground plane. These models also require a current impulse model to describe the current waveshape, and a return stroke model to mathematically describe the current distribution along the channel. This work presented commonly used current impulse models, such as the double exponential and Heidler functions, and also presented the Terespolsky function. This function is a recent Heidler function approximation that has the advantage of having an analytical integral solution, and is advantageous to the Finite Antenna EM model.

The second component uses the lightning EM source, as well as a model environment to simulate the EM fields that surround the lightning channel. Four models using three techniques have been presented for this purpose. As seen in Table 8.1, each of

Table 8.1: EM Field Propagation Models - Property Summary

	Finite Antenna	**3D FDTD**	**2D FDTD**	**Single Cell**
Maximum Range	Any	Near Fields	Moderate	Any
Complex 3D Environment	×	✓	×	×
Non-Ideal ground	×	✓	✓	×
Computer Memory Requirements	Low	High	Moderate	Low
Computational Efficiency	✓	×	✓	✓
Implementation Difficulty	Moderate	High	Moderate	Low
Single Point EM Fields	✓	×	×	✓
Multiple Points along a line	×	✓	×	×

these methods have specific applications and advantages in the field of lightning EM research. The Finite Antenna method is the classic approach to this research field, and is developed through the use of the Lorentz condition, where the lightning channel is modelled as the combination of finitely small dipole antennas. The method itself is reasonably simple to understand and implement, however is highly restricted to basic simulation environments (perfect ground plane, straight lightning channel, no obstacles in the simulation space). The method has the advantage of having the same computational requirement for evaluating fields at any distance from the lightning channel. A LIOV based simulation would require knowledge of the EM fields surrounding the length of the line throughout the time period of interest. If using the Finite Antenna method multiple simulations would need to be run to gain this information, however the 3D FDTD method does not have these limitations.

The Finite-Difference Time-Domain method deconstructs the simulation space of interest (2D or 3D) into smaller spatial cells. The EM fields in each of these cells are then evaluated in discrete time steps. The FDTD method requires the properties of each of these cells to be included in the model. Although this functionality was not demonstrated in great detail, the ability to define each of the cells properties allows for highly complex simulation environments to be constructed. These environments can include infinite variations in lightning source (current and path), as well as multilayer ground planes, ground plane contours, and objects (such as buildings and electrical networks) to be included in the simulation. The most important property of both FDTD implementations is the inclusion of non-ideal ground planes, which significantly affect the EM fields near the channel.

Although the 3D FDTD method is conceptually simple to understand, it has the disadvantage of coding complexity, as well as the high computer memory required to create even simple simulation environments. Due to this, 3D FDTD simulation environments have a limited size, and are best suited for simulating fields in complex environments near the lightning channel. A significant advantage of this method is that the fields at every (discrete) location in the simulation space are known for every time step of the simulation. This method also allows the inclusion of electrical components in the simulation space (such as the RLC elements of a distribution line). For LIOV based research this allows for a single run of a FDTD simulation to provide not only the EM fields

throughout the space, but also the overvoltage values across the modelled distribution line. An example of this nature was not included in this work, but does demonstrate the potential of future work with the methods discussed.

The 2D FDTD method is only appropriate for simulation environments that are axially symmetric surrounding the lightning channel. This is not advantageous for modelling complex environments with electrical networks, but it does allow for simple integration of a conducting ground plane into the environment. It also has the advantage of requiring significantly less (0.06%) computer memory than the equivalent 3D FDTD implementation, and therefore allows for modelling of fields at larger distances from the lightning channel, and with finer spatial/temporal resolution. Large distance simulation techniques have not been discussed, but can be derived. A problem with both the 2D and 3D FDTD methods is that they have a number of numerical limitations and are prone to stability and dispersion issues. Additionally FDTD requires boundary conditions to manage the fields at the edges of the simulation space.

The Single Cell FDTD method is a recent addition to the field of lightning EM field research. This method is a hybrid, and uses the magnetic field equations from the Finite Antenna method, and the theory of the FDTD method, to evaluate the fields surrounding a single Yee Cell in a 2D domain. The disadvantage of this method is that it is limited by the same restrictions as the Finite Antenna method, however its mathematical evaluation is simpler and allows for greater choice in the EM source models used. Additionally it does not suffer from the same numerical issues as a full FDTD implementation. Using alternative methods of solving the magnetic field equations has the potential to include the effects of a conducting ground plane to the method, however this is left for future research.

Future work for the 3D FDTD method include constructing complex environments that model complete distribution lines in the presence of a lightning stroke. An additional application could be to model the tortuous paths of observed lightning strokes, and compare the simulated fields to models using a straight perpendicular channel. This will aid in deciding which environmental factors and assumptions are the most significant.

The primary role of the 2D FDTD method in future applications would be to evaluate the effects of a conducting ground plane on lightning EM fields. This should first be tested by comparing recorded lightning EM fields, against the simulated EM fields in an equivalent simulation environment. Another application would be to investigate the lightning EM fields that exist inside the conducting ground. This may help further our understanding of the important role EM fields have on soil ionisation and earthing.

Chapter 9

Conclusion

Lightning return stroke electromagnetic (EM) fields have been modelled using four different time domain techniques. This process consists of two steps. The first describes the lightning channel as an EM source through the use of an engineering return stroke model, a current impulse model, and a number of other environmental assumptions. The second is choosing a method which best describes the EM field surrounding the lightning channel. This work has explained and critically evaluated the four methods.

The Finite Antenna method is best suited for simulating EM fields located a far distances from the lightning channel, or for simplistic simulation environments which assume ground to be a perfect electrical conductor. The 3D FDTD method is the most comprehensive method and allows for highly complex environments to be modelled and simulated. This method does however suffer from high computer memory requirements, as well as increased implementation complexity. This method is best suited for simulating fields near the lightning channel.

The 2D FDTD method does not allow for the same level of environment complexity as the 3D FDTD method, but it does allow for modelling conductive ground planes. Therefore this method offers a significant advantage over the Finite Antenna Method. In addition to this, the method requires far less computer memory than the 3D FDTD method, which makes it suitable for modelling EM fields at greater distances with improved spatial resolutions.

The Single Cell FDTD method is a novel technique which uses a combination of the previous two techniques to evaluate the EM field at a single point in space. This is the most computationally efficient model to implement, however it is limited by the same simple simulation environments as the Finite Antenna method. Alternative magnetic field models may offer this method greater scope in future research.

From this it is clear that the 3D FDTD method should be used when simulating the effects of lightning EM fields in complex environments, with or without, electrical networks such as distribution lines. The 2D FDTD method should be used for simulations that require knowledge of the lightning EM field at a single location in an environment with a conducting ground plane.

References

[1] C. A. Nucci and F. Rachidi, *The Lightning Flash*, ser. Power Series 34. Institute of Electrical Engineers, 2003, ch. 8 - Interaction of electromagnetic fields generated by lightning with overhead electrical networks, pp. 425–478.

[2] C. A. Nucci and F. Rachidi, *The Lightning Flash 2nd Edition*, ser. Power and Energy Series 69. Institution of Engineering and Technology (IET), 2014, ch. 12 - Interaction of electromagnetic fields generated by lightning with overhead electrical networks, pp. 559–609.

[3] C. A. Nucci and F. Rachidi, *Lightning Protection*, ser. Power and Energy Series 58. Institution of Engineering and Technology (IET), 2010, ch. 13, pp. 635–680.

[4] E. Perez, J. Herrera, and H. Torres, "Sensitivity analysis of induced voltages on distribution lines," in *IEEE Power Tech Conference*, Bologna, Italy., June 23rd-26th 2003.

[5] G. Diendorfer, "Induced voltage on an overhead line due to nearby lightning," *IEEE Transactions on Electromagntic Compatibility*, vol. 32, no. 4, pp. 292–299, November 1990.

[6] A. Piantini and J. M. Janiszewski, *Lightning Electromagnetics*, ser. Power and Energy Series 62. Institute of Engineering and Technology, 2012, ch. 19 - Scale models and their application to the study of lightning transients in power systems, pp. 719–764.

[7] M. Paolone, F. Rachidi, A. Borghetti, C. A. Nucci, M. Rubinstein, V. A. Rakov, and M. A. Uman, "Lightning electromagnetic field coupling to overhead lines: Theory, numerical simulations, and experimental validation," *IEEE Transactions on Electromagentic Compatibility*, vol. 51, no. 3, pp. 532–547, August 2009.

[8] M. Gijben, "The lightning climatology of South Africa," *South African Journal of Science*, vol. 108, p. #740, March 2012.

[9] V. A. Rakov and F. Rachidi, "Overview of recent progress in lightning research and lightning protection," *IEEE Transactions on Electromagentic Compatibility*, vol. 51, no. 3, pp. 428–442, August 2009.

[10] C. A. Nucci, S. Guerrieri, M. T. Correia de Barros, and F. Rachidi, "Influence of corona on the voltages induced by nearby lightning on overhead distribution lines," *IEEE Transactions on Power Delivery*, vol. 15, no. 4, pp. 1265–1273, October 2000.

[11] A. K. Agrawal, H. J. Price, and S. H. Gurbaxani, "Transient response of multiconductor transmission lines excited by a nonuniform electromagnetic field," *IEEE Transactions on Electromagnetic Compatibility*, vol. EMC-22, no. 2, pp. 119–129, May 1980.

[12] C. A. Nucci and F. Rachidi, "Lightning-induced overvoltages," in *IEEE Transmission and Distribution Conference, Pannel Session "Distribution Lightning Protection", New Orleans*, April 1999.

[13] F. Napolitano, A. Borghetti, C. A. Nucci, F. Rachidi, and M. Paolone, "Use of the full-wave finite element method for the numerical electromagnetic analysis of LEMP and its coupling to overhead lines," in *7th Asia-Pacific International Conference on Lightning (APL), Chengdu.*, November 2011, pp. 308–313, also published in ELSEVIER: Electric Power Systems Research, 2012.

[14] P. D. Kannu and J. Thomas, "Influence of ground conductivty on the over voltages induced on overhead power distribution lines due to an indirect lightning stroke," *IEEE Transactions on Electromagentic Compatibility*, vol. 2, pp. 949–954, 2000.

[15] Y. Baba and V. A. Rakov, *Lightning Electromagnetics*, ser. Power and Energy Series 62. Institute of Engineering and Technology, 2012, ch. 8 - Electromagnetic models of lightning return strokes, pp. 263–313.

[16] C. F. Barbosa and J. O. S. Paulino, "An approximate time-domain formula for the calculation of the horizontal electric field from lightning," *IEEE Transactions on Electromagentic Compatibility*, vol. 49, no. 3, pp. 593–601, August 2007.

[17] K. Tanabe, "Novel method for analyzing the transient behavior of grounding systems based on the finite-difference time-domain method," in *Power Engineering Society Winter Meeting, Columbus, OH*, vol. 3, 2001, pp. 1128–1132.

[18] V. Cooray, *The Lightning Flash 2nd Edition*, ser. Power and Energy Series 69. Institution of Engineering and Technology (IET), 2014, ch. 9 - Return stroke models with special attention to engineering applications, pp. 405–475.

[19] M. A. Uman and D. K. Mclain, "Magnetic field of lightning return stroke," *Journal of Geophysical Research*, vol. 74, no. 28, pp. 6899–6910, December 1969.

[20] V. A. Rakov and M. A. Uman, *Lightning Physics and Effects*. Cambridge University Press, 2004.

[21] V. Cooray, *The Lightning Flash*, ser. Power Series 34, V. Cooray, Ed. Institution of Engineering and Technology (IET), 2003.

[22] V. Cooray, *Lightning Electromagnetics*, ser. Power and Energy Series 62. Institute of Engineering and Technology, 2012, ch. 3 - Basic discharge process in the atmosphere, pp. 67–85.

[23] V. Cooray, *The Lightning Flash 2nd Edition*, ser. Power and Energy Series 69, V. Cooray, Ed. Institution of Engineering and Technology (IET), 2014.

[24] C. A. Nucci, C. Mazetti, F. Rachidi, and M. Ianoz, "On lightning return stroke models for LEMP calculations," *International Conference on Lightning Protection*, no. 4.7, pp. 463–470, 1988.

[25] *Protection Against Lightning - Part 1: General Principles*, IEC Std. 62 305-1, 2006.

[26] Q. Zhang, L. Zhang, W. Hou, and J. Su, "Validation of the approximate time-domain method for the lightning-horizontal electric field at the surface of two-layer earth by using FDTD," *IEEE Transactions on Electromagentic Compatibility*, vol. 56, no. 5, pp. 1121–1128, October 2014.

[27] C. W. I. McAfee and K. J. Nixon, "Lightning return stroke modeling with reference to lightning electromagnetic fields," *South African Universities Power and Energy Conference (SAUPEC), Johannesburg*, January 2015.

[28] F. Heidler and J. M. Cvetic, "A class of analytical function to study the lightning effects assosiated with the current front," *ETEP*, vol. 12, no. 2, pp. 141–150, March/April 2002.

[29] D. Lovrić, S. Vujević, and T. Modrić, "On the estimation of heidler function parameters for reproduction of various standardized and recorded lightning current waveshapes," *International Transactions on Electrical Energy Systems*, vol. 23, no. 2, pp. 290–300, 2011.

[30] B. R. Terespolsky and K. J. Nixon, "Developing an approximation to the Heidler Function - with an analytical transformation into the frequency domain." *International Conference on Lightning Protection*, pp. 1134–1138, 2014.

[31] B. R. Terespolsky, "An approximation to the heidler function with an analytical integral for engineering applications using lightning currents," Master's thesis, School of Electrical and Information Engineering, University of the Witwatersrand, South Africa, 2015.

[32] W. Jia and Z. Xiaoqing, "Double-Exponential expression of lightning current waveforms," *CEEM*, no. 3A1-09, pp. 320–323, 2006.

[33] C. A. Nucci, G. Diendorfer, M. A. Uman, F. Rachidi, M. Ianoz, and C. Mazetti, "Lightning return stroke current models with specified channel-base current: A review and comparison," *Journal of Geophysical Research*, vol. 95, no. D12, pp. 20 395–20 408, November 1990.

[34] Y. Baba and V. A. Rakov, "Electromagnetic models of the lightning return stroke," *Journal of Geophysical Research*, vol. 112, no. D04102, p. 17 pages, 2007.

[35] Y. Baba, S. Miyazaki, and M. Ishii, "Reproduction of lightning electromagnetic field waveforms by engineering model of return stroke," *IEEE Transactions on Electromagentic Compatibility*, vol. 46, no. 1, pp. 130–133, February 2004.

[36] Y. Baba and V. A. Rakov, "Applications of the FDTD method to lightning electromagnetic pulse and surge simulations," in *International Conference on Lightning Protection*. Shanghai, China, 2014, pp. 2084–2098.

[37] Y. Baba and V. A. Rakov, "Applications of the FDTD method to lightning electromagnetic pulse and surge simulations," *IEEE Transactions on Electromagentic Compatibility*, vol. 56, no. 6, pp. 1506–1521, December 2014.

[38] M. A. Uman, D. K. Mclain, and E. P. Krider, "The electromagnetic radiation from a finite antenna," *American Journal of Physics*, vol. 43, pp. 33–38, January 1975.

[39] M. Ishii, K. Michishita, Y. Hongo, and S. Oguma, "Lightning-induced voltage on an overhead wire dependent on ground conductivity," *IEEE Transactions on Power Delivery*, vol. 9, no. 1, pp. 109–118, January 1994.

[40] Z.-D. Jiang, B.-H. Zhou, Y.-W. Liu, and B. Yang, "A multiresolution time-domain method for LEMP calculation and comparison with FDTD," *IEEE Transactions on Electromagentic Compatibility*, vol. 56, no. 2, pp. 419–426, April 2014.

[41] Z. E. Azzouz, A. Mimouni, B. Ghemri, and A. Cherifi, "Analysis of radiated-lightning electromagnetic fields anbove imperfect ground using a Quasi-FDTD hybrid method," *Acta Electrotechnica et Informatica*, vol. 8, no. 4, pp. 16–23, 2008.

[42] M. Popov, S. He, and R. Thottappillil, "Reconstruction of lightning currents and return stroke model parameters using remote electromagnetic fields," *Journal of Geophysical Research*, vol. 105, no. D19, pp. 24 469–24 481, October 2000.

[43] D. Djalel, H. Ali, and C. Fayçal, "The return-stroke of lightning current, source of electromagnetic fields (study, analyis and modelling)," *American Journal of Applied Sciences*, pp. 42–48, 2007.

[44] S. Rusck, "Induced lightning over-voltages on power-transmission lines with special reference to the over-voltage protection of low-voltage networks," Master's thesis, Transactions of the Royal Institute of Technology, 1958.

[45] Y. Bo, Z. Bi-hua, C. Yong-guang, and M. Xin, "Influence of mesh size and lightning channel length on lightning electromagnetic field calculation using FDTD algorithm," in *Environmental Electromagnetics, 5th Asia-Pacific Conference on Lightning*, 2009.

[46] V. A. Rakov and A. A. Dulzon, "A modified transmission line model for lightning return stroke field calculations," *International Symposium on EMC*, no. 44H1, pp. 229–234, 1991.

[47] J. D. Kraus and R. J. Marhefka, *Antennas for all applications*, 3rd ed. McGraw and Hill, 2003.

[48] U. S. Inan and R. A. Marshall, *Numerical Electromagnetics: The FDTD Method.* Cambridge University Press, 2011.

[49] A. Z. Elsherbeni and V. Demir, *The Finite-Difference Time-Domain Method for Electromagnetics with MATLAB® Simulations*, 2nd ed. SciTech Publishing, 2015.

[50] R. Thottappillil, *The Lightning Flash 2nd Edition*, ser. Power and Energy Series 69. Institution of Engineering and Technology (IET), 2014, ch. 8 - Computation of electromagnetic fields from lightning discharge, pp. 351–403.

[51] M. A. Uman, "Lightning return stroke electric and magnetic fields," *Journal of Geophysical Research - Atmospheres*, vol. 90, no. D4, pp. 6121–6130, June 1985.

[52] R. Thottappillil and V. A. Rakov, "On different approaches to calculating lightning electric fields," *Journal of Geophysical Research*, vol. 106, no. D13, pp. 14 191–14 205, July 2001.

[53] M. Izadi, M. Z. A. Ab Kadir, C. Gomes, and W. F. Wan Ahmad, "An analytical second-FDTD method for evaluation of electric and magnetic fields at intermediate distances from lightning channel," *Progress In Electromagnetics Research*, vol. 110, pp. 329–352, 2010.

[54] K. S. Yee, "Numerical solution of initial boundry value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. AP-14, no. 3, pp. 302–307, May 1966.

[55] J. B. Schneider, "Understanding the Finite-Difference Time-Domain method," 2015. [Online]. Available: www.eecs.wsu.edu/~schneidj/ufdtd

[56] E. Soto, C. Younes, and E. Pérez, "Influence of non flat terrain on lightning induced voltages in distribution lines," in *International Conference on Lightning Protection (ICLP), Vienna, Austria*, 2012.

[57] H.-M. Ren, B.-H. Zhou, V. A. Rakov, L.-H. Shi, C. Gao, and J.-H. Yang, "Analysis of lightning-induced voltages on overhead lines using a 2-D FDTD method and agrawal coupling model." *IEEE Transactions on Electromagentic Compatibility*, vol. 50, no. 3, pp. 651–659, August 2008.

[58] C. A. F. Sartori and J. R. Cardoso, "An Analytical-FDTD method for near LEMP calculation," *IEEE Transactions on Magnetics*, vol. 36, no. 4, pp. 1631–1634, July 2000.

[59] N. M'ziou, L. Mokhnache, and A. Boubakeur, "Experimental validation of the hybrid method for near lightning electromagnetic field calculation taking into account the conductivity of the soil," in *International Symposium on High Voltage Engineering (ISH)*, ser. G-30, 2009.

[60] C. W. I. McAfee, K. M. Ortlepp, and K. J. Nixon, "Finite Antenna and Single Cell FDTD methods applied to near LEMP calculations." Balneario Camboriu, Brazil: International Symposium on Lightning Protection (SIPDA XIII), September 2015, pp. 41–46.

Bibliography

Aodsup, K. and Kulworawanichpong, T. (2014). FDTD method for lightning surge propogation of power transmission lines. *The Standard International Journals (The SIJ), Transactions on Computer Networks and Communications Engineering (CNCE)*, 2(3):31–35.

Berger, K., Anderson, R., and Kroninger, H. (1975). Parameters of lightning flashes. *Electra*, (41):23–37.

CIGRÉ WG 33-01, . (1991). Guide to procedures for estimating the lightning performance of transmission lines. *CIGRÉ Monograph*, 63.

Cooray, V. (1994). Calculating lightning-induced overvoltages in power lines: A comparison of two coupling models. *IEEE Transactions on Electromagnetic Compatability*, 36(3):179–182.

Cooray, V. (2010). *Lightning Protection*. Power and Energy Series 58. Institution of Engineering and Technology (IET).

Cooray, V. (2012). *Lightning Electromagnetics*. Power and Energy Series 62. Institution of Engineering and Technology (IET).

Cooray, V. and Cooray, G. (2012). *Lightning Electromagnetics*, chapter 24 - Excitation of visual sensory experiences by electromagnetic fields of lightning, pages 855–872. Power and Energy Series 62. Institute of Engineering and Technology.

Djalel, D., Lazhar, R., and Hocine, L. (2012). A new model of electromagnetic fields radiated by lightning. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(4):182–190.

Dragan, G. (2001). Technica tensiunilor inalte. *Editura Academiei Romane*, 2.

Dragan, G., Florea, G., Nucci, C. A., and Paolone, M. (2010). On the influence of corona on lightning-induced overvoltages. In *International Conference on Lightning Protection*.

Georgiadis, N., Rubinstein, M., Uman, M. A., Medelius, P., and Thomson, E. (1992). Lightning-induced voltages at both ends of a 488-m power distribution line. *IEEE Transactions on Electromagentic Compatibility*, 34(4).

Gómez, P. and Escamilla, J. C. (2013). Frequency domain modeling of nonuniform multiconductor lines excited by indirect lightning. *ELSEVIER Int. Journal of Electrical Power & Energy Systems*, 45(1):420–426.

Grando, J., Issac, F., Lemistre, M., and Alliot, J. (1993). Stability analysis including wires of arbitary radius in FD-TD code. In *Antennas and Propagation Society International Symposium*.

Heidler, F., Cvetic, J. M., and Stanic, B. V. (1999). Calculation of lightning current parameters. *IEEE Transactions on Power Delivery*, pages 399–404.

Høidalen, H. K. (1999). Calculation of lightning-induced overvoltages using MODELS. In *Proceedings of the International Conference on Power System Transients, Budapest, Hungary.*, pages 359–364.

Hu, W. and Cummer, S. A. (2006). An FDTD model for low and high altitude lightning-generated em fields. *IEEE Transactions on Antennas and Propogation*, 54(5):1513–1522.

Izadi, M., Kadir, M. Z. A. A., Ahmad, W. F. W., Nawi, Z. M., and Askari, M. T. (2009). On comparison between cooray-rubinstein and FDTD mmethod for ground conductivity effect on horizontal electric field evaluation in time domain. In *IEEE Student Conference on Research and Development (SCOReD), UPM Serdang, Malaysia*.

Maruvada, P. S. (2011). *Corona in Transmission Systems. Theory, Design and Performance*. Crown Publications.

Maruvada, P. S., Menemenlis, H., and Malewski, R. (1977). Corona charachteristics of conductor bundles under impulse voltages. *IEEE Transactions on Power Apparatus and Systems*, PAS-96(1):102–115.

Master, M., Uman, M. A., Beasley, W., and Darveniza, M. (1984). Lightning induced voltages on power lines: Experiment. *IEEE Transactions on Power Apparatus and Systems*, PAS-103(9):2519–2529.

Master, M. J. and Uman, M. A. (1983). Transient electric and magnetic fields associated with establishing a finite electrostatic dipole. *American Journal of Physics*, 51(2):118–126.

Omick, S. R. and Castillo, S. P. (1993). A new finite-difference time-domain algorithm for the accurate modeling of wide-band electromagnetic phenomena. *IEEE Transactions on Electromagnetic Compatability*, 35:215–222.

Rachidi, F., Nucci, C. A., Guerrieri, S., and Correia de Barros, M. T. (2001). On the amplitude enhancement of voltages induced by external EM field on transmission lines due to ground losses and corona phenomenon. *IEEE Transactions on Electromagnetic Compatibility*, 1:600–604.

Rubinstein, M. and Uman, M. A. (1991). Transient electric and magnetic fields assosiated with establishing a finite electrostatic dipole, revsited. *IEEE Transactions on Electromagentic Compatibility*, 33(4):312–320.

Sewkumar, P. (2001). Modeling the effect of adjacent lightning strikes to bare overhead medium voltage lines in South Africa. Master's thesis, Faculty of Engineering and the Built Environment, School of Electrical and Electronic Engineering, University of the Witwatersrand, South Africa.

Shoory, A., Rachidi, F., Rubinstein, M., and Thottappillil, R. (2001). On the measurement and calculation of horizontal electric fields from lightning. *IEEE Transactions on Electromagentic Compatibility*, 53(3):792–801.

Tang, T., Sun, X.-b., Liu, K., Guo, Z.-h., and Chen, D. (2011). Electromagnetic analyss for lightning propagation in lossy soil by using FDTD. In *Asia-Pacific International COnference on Lightning*, pages 65–68.

Taobin, J., Jing, L., and Jie, J. (2013). Calculation of the lightning electromagnetic field for vertically stratified ground using the novel scheme FDTD. In *International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)*, pages 599–603.

Thang, T. H., Baba, Y., Nagaoka, N., Ametani, A., Itamoto, N., and Rakov, V. A. (2014). FDTD simulations of corona effect on lightning-induced voltages. *IEEE Transactions on Electromagnetic Compatibility*, 56(1):168–176.

Thang, T. H., Baba, Y., Nagaoka, N., Ametani, A., Takami, J., Okabe, S., and Rakov, V. A. (2012a). FDTD simulation of lightning surge on overhead wires in the presence of corona discharge. *IEEE Transactions on Electromagentic Compatibility*, 54(6):1234–1243.

Thang, T. H., Baba, Y., Nagaoka, N., Ametani, A., Takami, J., Okabe, S., and Rakov, V. A. (2012b). A simplified model of corona discharge on overhead wire for FDTD computations. *IEEE Transactions on Electromagentic Compatibility*, 54(3):585–593.

Visacro, S. (2004). A representation curve for lightning current waveshape of first negative stroke. *Geophysical Research Letters*, 31(L07112).

Vu, L. A. P., Vu, P. T., and Ho, V. T. (2012). Calculation of lightning-induced voltages on overhead power lines using the RBF-FDTD method. In *International Power and Energy Conference*, pages 573–577.

Wei, H., Ba-lin, X., and You-gang, G. (2004). Analysis of the lightning waveshape. In *Radio Science Conference*.

Yang, C. and Zhou, B. (2004). Calculation method of electromagnetic fields very close to lightning. *IEEE Transactions on Electromagnetic Compatability*, 46(1):133–141.

Yang, D., Zhao, Z., Cui, X., and Chen, J. (2009). Analysis of the effects of ground resistivity on the lightning radiation fields based on FDTD method. In *Asia-Pacific Power and Energy Engineering Conference (APPEEC)*.

Yao, C., Wu, H., Mi, Y., Ma, Y., Shen, Y., and Wang, L. (2013). Finite difference time domain simulations of lightning transient electromagnetic fields on transmission lines. *IEEE Transactions on Dielectrics and Electrical Insulation*, 20(4):1239–1246.

Yokoyama, S., Miyake, K., Mitani, H., and Yakanishi, A. (1983). Simultaneous measurement of lightning induced voltages with assosiated stroke ccurrent. *IEEE Transactions on Power Apparatus and Systems*, PAS-102(8):2420–2429.

LIGHTNING RETURN STROKE MODELLING WITH REFERENCE TO LIGHTNING ELECTROMAGNETIC FIELDS

C.W.I. McAfee and K.J. Nixon*

* *School of Electrical and Information Engineering, Faculty of Engineering and the Built Environment, University of the Witwatersrand, South Africa*

Abstract: The theory regarding lightning strokes, and return stroke models is presented along with the functions needed to model the return stroke. Three engineering models are presented: TLM, MTLE, MTLL. Three current impulse functions (required by the return stroke model) are also presented in detail: Double Exponential, Heidler and Terespolsky. The emphasis of the theory and models is focused towards future use in evaluating radiated lightning electromagnetic (EM) fields. Although no EM models or simulations are presented here, the use of the return stroke models is discussed with reference to a simplified electric field model. A comparison found that the Terespolsky function is better suited for modelling the return stroke for future use in EM models.

Key words: Return Stroke, Channel Base Current, Current Impulse Model, Induced Electromagnetic Fields

1. INTRODUCTION

In lightning protection the primary focus is often on direct lightning strikes attaching to ground objects. This is the most intuitive form of damage caused from lightning, and often results in serious property damage or loss of life [1]. However indirect (nearby) lightning is also responsible for property damage and needs to be designed for in lightning protection. There are a number of deleterious effects caused by indirect lightning, however the effect of lightning electromagnetic (EM) radiation is the primary focus presented here. During a lightning strike a large current flows in the leader channel. This moving current results in moving EM waves that propagate from the channel. When these EM waves move past electrical networks such as distribution lines, or telecommunications lines (or any conducting object) a current is induced in the network. Often these induced currents are large enough to result in damage to the network, or the electrical equipment attached to the network nodes [2].

A particular interest is the effect of a Lightning Electromagnetic Pulse (LEMP) inducing an overvoltage on a power distribution network. These overvoltages result in network damage, as well as interrupted energy delivery. In comparison. indirect strikes occur more frequently than direct strikes and are therefore more important [1] when considering the lightning protection of distribution lines (not transmission lines). In order to evaluate the induced overvoltages it is necessary to first evaluate the EM field that interacts with the line. In order to do this it is first necessary to develop a model that describes the current that flows in the lightning channel. This is referred to as the lightning Return Stroke model, and is attributed to causing the most damage to distribution networks [1–4]. Once a return stroke model has been developed it is then possible to use the model in conjunction with Maxwell's equations to evaluate the EM field in space and time [5].

Theory regarding return strokes is presented, and then the different modelling techniques are discussed. One of the aims is to simplify this topic for engineering applications. The different channel base current (current impulse) models are discussed, and simulated. The electromagnetic radiation theory of a lightning strike is briefly presented (without calculations) to show how the return stroke models will need to be evaluated in calculating electromagnetic fields. The results of these models, and decisions made are then discussed.

2. RETURN STROKE THEORY

There are four types of lightning as defined by Rakov [6], however only downward negative lightning will be discussed as it accounts for 90% of cloud to ground lightning globally. A lightning strike or lightning flash is composed of multiple lightning strokes. A lightning stroke is composed of a downward leader, followed by an upward return stroke [6]. For downward negative lightning, a stepped downward leader begins to travel downwards from the cloud to ground (due to a charge difference between the two locations). When the stepped leader gets close to ground (within a couple hundred meters) the electric field strength increases enough to cause upward leaders to form. When the upward leader connects with the downward stepped leader, the ground point that initiated the upward leader is "struck" by lightning.

As a stepped leader moves towards earth it leaves a conducting channel, as well as depositing negative charge. When the stepped leader connects to ground, the ground charge sees a conducting path to the cloud. Ground charge then moves into the channel to neutralise the negative charge. This flow of charge is called the return stroke, and it flows from ground to cloud. Under this situation both the leader and return stroke effectively transport negative charge to ground [6]. First strokes are often followed by subsequent strokes in negative downward lightning. A

subsequent stroke begins with the progression of a "dart" leader from cloud to ground (not necessarily along the initial stepped leader channel), and then followed by a return stroke from ground to cloud. It is also important to note that a return stroke may contain a low "continuing current" that follows a return stroke [6], however this will not be dealt with in the models. Subsequent return strokes will typically have less peak current than the first return stroke, therefore only the first return stroke will be modelled.

3. RETURN STROKE MODELS

Section 2. has discussed the theory of a return stroke from a physics viewpoint. Essentially a return stroke is a current that travels from the ground to a cloud. There are a number of factors that need to be considered when constructing a return stroke model:

- Path travelled by the return stroke.
- Channel height.
- Current impulse along channel.
- Current attenuation along the channel.
- Velocity of impulse moving upward in the channel.

Some of these model components are simple to define, however others can only be assumed. It is important to understand that a model is a mathematical construct which can be used to describe observed phenomena. These models are then used to understand variations of a phenomena under study [1]. This may seem obvious, but is important in understanding the meaning of the return stroke models being presented.

Measurements of return strokes have primarily been done through:

Optical/Photographic Measurements: Light intensity of a stroke indicating current density.

Channel Base Current Measurements: Current is measured at the base of a lightning channel which is initiated either by a tall conducting structure/tower, or rocket triggered lightning [1].

EM Field Measurements: The EM fields of a lightning stroke are measured, in addition to the stroke location [7].

It is important to note that none of these measurements can accurately describe the current throughout the leader channel, throughout time. Therefore all of these measurements are used to create models that are able to validate the measurements made, and infer the current distribution in the channel.

There are three main groups of return stroke models: *Electrothermodynamic models*, *RLC Transmission line models* and *Semi-Physical and Engineering models* [1]. However for the purpose of this paper only the Semi-Physical and Engineering models will be discussed. Both the Semi-Physical and Engineering approaches have adapted concepts from the RLC transmission line models. This group of models is better described in two different subcategories: *Current propagation models* (predominantly engineering models) and *Current generation models* (predominantly semi-physical models). As stated in Section 1., the main focus of this report is on engineering applications, and therefore only the Current Propagation models will be discussed in detail, however more information on the alternative models can be found in [1, 6].

3.1 Current Propagation (CP) models

In this approach the return stroke is modelled as a transmission line driven by a current source connected to the ground plane. In engineering models the current on the channel is described in space and time, and this is used to calculate remote EM fields. Figure 1 shows a visual representation of the model with a return stroke current impulse moving upwards from ground to cloud. The current impulse $I(h,t)$ is able to describe the current at any position (height,h) on the channel, at any time (t). This figure also shows that the stepped leader channel is modelled as a perpendicular uniform path on top of a perfectly conducting ground plane. These model components are not real representations of a lightning stroke. A stepped leader path down to earth is actually defined by multiple short paths that branch randomly in different directions (as seen in any typical lightning picture). The conducting path may vary due to the charge distribution, and the ground plane is never a perfect conductor due to ground loss and terrain layout.

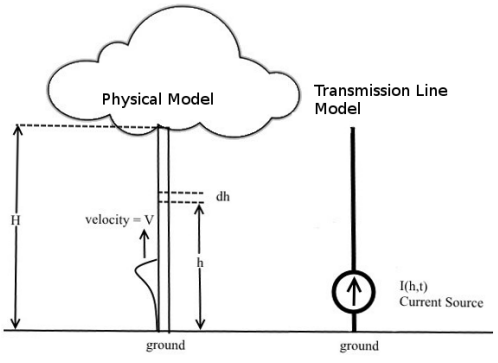


Figure 1: CP Return stroke model

Figure 1 also shows that the return stroke impulse travels up the channel at constant speed $v = 0.5 * c$ where c is the speed of light. In reality the velocity of the return stroke speed decreases with height, however for the purpose of simplifying the model the speed is considered constant between $0.8 - 2.8 * 10^8 m/s$ [1, 6].

Equations 1 to 3 show the three main mathematical equations used to model the current distribution along the return stroke channel. The left hand side of these equations shows that the current distribution is height and time dependant, however the rights hand side shows the channel base current ($h = 0$). As the name would suggest, the channel base current is typically specified as the current measured at the base as a function of time ($I(t)$). In order to compensate for the displacement of the return stroke current pulse along the return channel the time variable is shifted: $t_o = t_n - \frac{h}{v}$ where t_o is the original time component, and t_n is the new time component simply referred to as t in the model. It is also important to note that channel base current only exists for $t \geq \frac{h}{v}$, which is mathematically included in the equations by using a unit step function $U(t_o)$.

$$\text{TLM: } I(h,t) = I(0, t - \frac{h}{v}) \cdot (U(t - \frac{h}{v})) \quad (1)$$

$$\text{MTLE: } I(h,t) = \exp(-\frac{h}{\lambda_c}) \cdot (I(0, t - \frac{h}{v})) \cdot (U(t - \frac{h}{v})) \quad (2)$$

$$\text{MTLL: } I(h,t) = (1 - \frac{h}{H}) \cdot (I(0, t - \frac{h}{v})) \cdot (U(t - \frac{h}{v})) \quad (3)$$

Equation 1 is the *Transmission Line Model* from Uman and McLain [1]. This model has an accuracy of about 20%, however it fails to account for the attenuation of the current along the channel. It has been observed that the current amplitude decreases as the return stroke current approaches the cloud (due to the charge in the corona sheath that forms around the leader channel [8]). In other words the problem with the TLM is that it does not account for charge neutralisation along the channel. To account for this the TLM was modified by decreasing the current amplitude while still maintaining the waveshape. Equation 2 is the *Modified Transmission Line Exponential* model proposed by Nucci *et al* [8]. As the name would suggest, the amplitude of the current decreases exponentially with height, where $\lambda_c = 2000m$. An alternative is to use Equation 3, which is the *Modified Transmission Line Linear* model, as proposed by Rakov *et al* [9]. With this model the current decreases linearly with height.

Both the modified models show an improvement with measured results as compared to the TLM. The main difference between the modified models is that the MTLL model is more accurate with measurements made near ($50m$) to lightning strikes.

4. CHANNEL BASE CURRENT MODELS

There are a number of different models used to represent the current of a lightning return stroke base current (Current Impulse Model). These models have been derived from measurements made at ground level, and attempt to account for the current wave shape, front rise time and peak current values (the characteristics of the wave). For this study the current impulse will be a $10/350$ *us* waveshape with a peak current of 200 *kA*. These values were chosen from the IEC Std 62305-1 [10]. The choice of a 200 *kA* peak current was made for the worst case scenario (high level protection), however it is important to note that 50% of first return strokes have a value of 30 *kA* [6]. The choice of the waveshape was made due to the long duration of the $10/350$ *us* impulse, however there is no reason why another wave shape couldn't be chosen. This paper will discuss three current impulse models: *Double Exponential*, *Heidler* and *Terespolsky*.

4.1 Double Exponential Function

Equation 4 shows the mathematical equation for the double exponential function [11]. The parameters for the function were taken from [12] in order to achieve the required waveshape.

$$I(t) = I_p A \cdot (e^{-\alpha t} - e^{-\beta t}) \quad (4)$$

Where

A	=	Scaling Factor	=	1.025
I_p	=	Peak Current	=	200 kA
α	=	Time Constant	=	$2.05 * 10^3\ s^{-1}$
β	=	Time Constant	=	$5.64 * 10^5\ s^{-1}$

This function has been used in a number of lightning related research projects, however it does not accurately represent a physically realisable return stroke current waveform. As seen in Figure 2, the rise time of this function is instantaneous, which is not physically possible [6]. As the stepped leader approaches the ground during a lightning stroke, ground leaders start to form (causing a slow increase in current), and once the leader connects to upward leader, the current increases rapidly. In addition to this, due to the incorrect shape of the waveform there are incorrect representations of the radiated frequency components of the stroke. Even though the model does not accurately represent a lightning stroke, the mathematical properties of this function make it useful in preliminary modelling.

4.2 Heidler Function

The Heidler function (Equation 5) is another popular function used to model the return stroke current waveform. The parameters for the function were taken from [10] in order to achieve the required waveshape.

$$I(t) = \frac{I_p}{k} \cdot \frac{(\frac{t}{\tau_1})^n}{1 + (\frac{t}{\tau_1})^n} \cdot e^{-\frac{t}{\tau_2}} \quad (5)$$

Where

k	=	Scaling Factor	=	0.93
I_p	=	Peak Current	=	200 kA
n	=	Steepness Factor	=	10
$\tau 1$	=	Time Constant	=	$19 * 10^{-6}$ s
$\tau 2$	=	Time Constant	=	$485 * 10^{-6}$ s

This function is a closer approximation to the physical processes of a lightning stroke. As seen in Figure 2, shape of the waveshape as it rises is a better fit for the lightning stroke, and therefore overcomes the problems of the double exponential function. The problems associated with this function will be described in Section 5..

4.3 Terespolsky Function

Equation 6 shows the Terespolsky function. This function is an approximation to the Heidler function, and has less than 1.5% error [13] when adjusted to the IEC Std 62305-1 [10].

$$I(t) = \frac{I_p}{k} \cdot \left(1 - e^{-\omega_0 t}\left(\sum_{i=0}^{n} \frac{\omega_0^i t^i}{i!}\right)\right) \cdot e^{-\frac{t}{\tau_2}} \tag{6}$$

Where

k	=	Scaling Factor	=	0.93
I_p	=	Peak Current	=	200 kA
n	=	Steepness Factor	=	33
ω_0	=	Rise Time Constant	=	1768211
$\tau 2$	=	Time Constant	=	$485 * 10^{-6}$ s

The advantage in using the Terespolsky function is that it is simpler to manipulate mathematically [13], and therefore simplifies lightning protection models. As seen in Figure 2 and 3 the Terespolsky function closely resembles the Heidler function, and although not yet accredited as an IEC standard, the Terespolsky function is a good approximation. This means that model results from this function should be comparable to those required by IEC Std 62305-1.

5. LIGHTNING ELECTROMAGNETIC PULSE EQUATION

Equation 7 shows a special case of the vertical electric field measured at a point on the ground plane a distance r from the return stroke channel (Figure 2.). The full electric and magnetic field equations can be found in [5, 7].

$$E_z(r,t) = \frac{1}{2\pi\varepsilon_0}\Big[\int_0^H \frac{2h^2 - r^2}{R^5}\left(\int_0^t I(\tau - \frac{R}{c} - \frac{h}{v})d\tau\right)dh + \int_0^H \frac{2h^2 - r^2}{cR^4}\left(I(t - \frac{R}{c} - \frac{h}{v})\right)dh - \int_0^H \frac{r^2}{c^2R^3}\left(\frac{\partial I(t - \frac{R}{c} - \frac{h}{v})}{\partial t}\right)dh\Big]\vec{z} \tag{7}$$

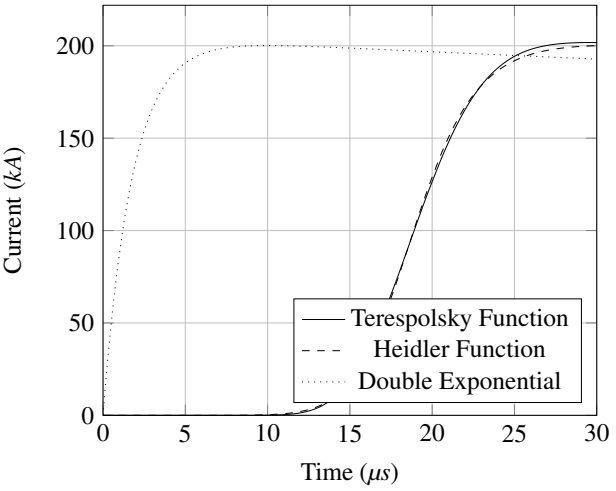


Figure 2: Rise Time - Channel Base Current Models

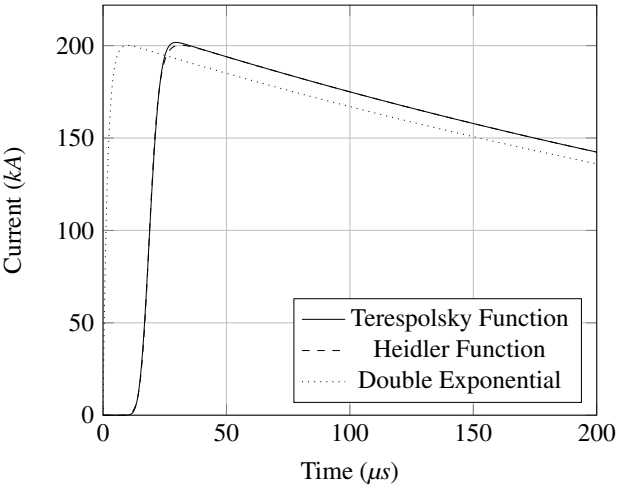


Figure 3: Channel Base Current Models

Where

H	=	Channel Height
R	=	$sqrt(r^2 + h^2)$
h	=	Height of a point along the channel
r	=	Distance from channel base to point
ε_0	=	Permittivity of free space $4\pi * 10^7$

The purpose of including this equation here is not to simulate the electric field, but rather to emphasise the choice between the channel base current model chosen when modelling the EM fields. As seen in Equation 7, the return stroke current, $I(t)$, will need to be integrated twice, as well as differentiated. This is easily done when the double exponential function is chosen, however it is not an accurate model. It is however useful in understanding how a pulse propagates.

The next choice is to use the Heidler function, however after inspecting Equation 5 it should become clear that integrating this function is not a simple task. Different numerical techniques need to be used in order to achieve a result. This is where the Terespolsky function is most useful. Integrating and differentiating this function is far simpler, and easy to implement in a computer simulation [13].

6. CONCLUSION

The theory regarding lightning strikes and lightning return strokes has been presented, and the concepts necessary for understanding physical processes have been discussed. The different methodologies for modelling the return stroke current along the leader channel have been presented for engineering applications, and three mathematical models have been provided for future simulations. The return stroke channel base currents have also been described. The *Double Exponential*, *Heidler* and *Terespolsky* lightning impulse functions were presented along with practical constants that were used to model a $10/350$ *us* waveshape with a peak current of 200 *kA* (worst case lightning stroke). An equation used to model the radiated lightning electric field was presented, and the benefits of using each of the respective current impulse models (in conjunction with the return stroke models) were presented. In this respect it was found that using the Terespolsky function to model the return stroke current would be the best option. This is because the Terespolsky function is simpler to manipulate mathematically than the Heidler function. In order to evaluate the EM fields radiated from a lightning stroke it is necessary to integrate the lightning current model. The theory presented in this paper should act as a guide in progressing to models involving EM radiation.

ACKNOWLEDGEMENTS

The authors would like to thank CBI-electric for funding the Chair of Lightning at the University of the Witwatersrand and for direct support of the Research Group. They would also like to thank Eskom for the support of the Lightning/EMC Research Group through the TESP programme. Thanks are extended to the department of Trade and Industry (DTI) for THRIP funding as well as to the National Research Foundation (NRF) for direct funding of the Research Group.

Special thanks to Brett Terespolsky for the help in implementing his new function, and to Darryn Cornish for graphical help.

REFERENCES

[1] C. A. Nucci and F. Rachidi, *The Lightning Flash*, ser. Power Series 34, V. Cooray, Ed. Institute of Electrical Engineers, 2003.

[2] M. Paolone, F. Rachidi, A. Borghetti, C. A. Nucci, M. Rubinstein, V. A. Rakov, and M. A. Uman, "Lightning electromagnetic field coupling to overhead lines: Theory, numerical simulations, and experimental validation," *IEEE Transactions on Electromagentic Compatibility*, vol. 51, no. 3, pp. 532–547, August 2009.

[3] P. Sewkumar, "Modeling the effect of adjacent lightning strikes to bare overhead medium voltage lines in south africa," Master's thesis, Faculty of Engineering and the Built Environment, School of Electrical and Electronic Engineering, University of the Witwatersrand, South Africa, 2001.

[4] C. A. Nucci and F. Rachidi, *Lightning Protection*, ser. 58, V. Cooray, Ed. The Institute of Engineering and Technology, 2010, chapter 13.

[5] M. A. Uman, D. K. Mclain, and E. P. Krider, "The electromagnetic radiation from a finite antenna," *American Journal of Physics*, vol. 43, pp. 33–38, January 1975.

[6] V. A. Rakov and M. A. Uman, *Lightning Physics and Effects*. Cambridge University Press, 2004.

[7] M. A. Uman, "Lightning return stroke electric and magnetic fields," *Journal of Geophysical Research - Atmospheres*, vol. 90, no. D4, pp. 6121–6130, June 1985.

[8] C. A. Nucci, C. Mazetti, F. Rachidi, and M. Ianoz, "On lightning return stroke models for lemp calculations," *International Conference on Lightning Protection*, no. 4.7, pp. 463–470, 1988.

[9] V. A. Rakov and A. A. Dulzon, "A modified transmission line model for lightning return stroke field calculations," *International Symposium on EMC*, no. 44H1, pp. 229–234, 1991.

[10] *Protection Against Lightning - Part 1: General Principles*, IEC Std., 2006.

[11] K. Berger, R. Anderson, and H. Kroninger, "Parameters of lightning flashes," *Electra*, no. 41, pp. 23–37, 1975.

[12] W. Jia and Z. Xiaoqing, "Double-exponential expression of lightning current waveforms," *CEEM*, no. 3A1-09, pp. 320–323, 2006.

[13] B. R. Terespolsky and K. J. Nixon, "Developing an approximation to the heidler function - with an analytical transformation into the frequency domain." *International Conference on Lightning Protection*, pp. 1134–1138, 2014.

2015 International Symposium on Lightning Protection (XIII SIPDA), Balneário Camboriú, Brazil, 28th Sept. – 2nd Oct. 2015.

Finite Antenna and Single Cell FDTD methods applied to near LEMP calculations

Carson W. I. McAfee, Kerren M. Ortlepp, Ken J. Nixon
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg, South Africa
Email: ken.nixon@wits.ac.za

***Abstract*—Two methods for modeling the electromagnetic fields near a lightning channel are considered. These methods can be used in evaluating the effect of a lightning electromagnetic pulse (LEMP) on a distribution line. The first method for evaluating LEMP fields in the time domain uses a Heidler function approximation for the channel base current in the classic Finite Antenna method. This approximation has an analytical integral solution which simplifies the field calculations. The second method is a combination of standard FDTD theory and the mathematics of the Finite Antenna method. The proposed Single Cell or Hybrid FDTD method is derived for spherical co-ordinates and tested against the Finite Antenna method. The results show that under simple modeling criteria both methods are valid for LEMP calculations near a lightning channel. The Single Cell FDTD method only evaluates a single FDTD cell, rather than an entire plane of cells described by the standard FDTD method. This greatly reduces the computational intensity, and makes the Single Cell FDTD method a valuable tool.**

***Index Terms*—Lightning, LEMP, Nearby Fields, Hybrid FDTD, Finite Antenna, Heidler Approximation, Time-Domain.**

I. Introduction

The field of lightning electromagnetics is important when considering lightning induced overvoltages (LIOV) on electrical networks. A specific area of interest is modeling the LIOV on a distribution line, which is more prone to nearby lightning failures than direct lightning failures [1]. The expected electromagnetic (EM) fields at any point surrounding the lightning channel need to be calculated before the effects of a lightning electromagnetic pulse (LEMP) are evaluated on an electrical network. Standard EM computation methods such as the method of moments technique often rely on calculations done in the frequency domain, which is best suited when working with continuous and periodic fields. Frequency domain methods can be used in the calculation of a LEMP [2], however, lightning is transient in nature and there is an advantage evaluating its effects in the time domain.

One method for evaluating a LEMP in the time domain is to consider a lightning channel as a sum of finite dipole antennas [3], however the mathematical evaluation of this method is difficult when using a Heidler function in the current impulse model (especially for locations near the channel), as discussed in Section II. A possible solution is to use an alternative current impulse model, which can be analytically integrated.

Another method available for LEMP calculation is the Finite-Difference Time-Domain (FDTD) method, which is well suited for time domain analysis. The standard FDTD method typically involves the discretization of the entire evaluation area, which can be computationally intensive. This paper investigates the application of a Single Cell FDTD method which combines standard FDTD theory with the mathematics of the Finite Antenna method.

Section II discusses the background theory to the models, and Section III outlines the assumptions made. The alternative current impulse (channel base current) model is presented in Section IV, and then applied to the Finite Antenna method in Section V. The Single Cell FDTD mathematics for this particular application (in spherical co-ordinates) are derived and discussed in Section VI. The results from the Finite Antenna method (under simplified modeling criteria) are verified against pre-established values, which are then used to verify the results from the Single Cell FDTD method in Section VII. Section VIII discusses the simulation results, and the accuracy requirements of the Single Cell FDTD method, from which conclusions are drawn.

II. Background

The first work done on time domain lightning EM calculation was by Uman *et al* [3] which has been used and improved in a number of different research papers [4], [5], [6], [7]. This method involves modeling a lightning channel as a sum of finite dipole antennas over the length of the lightning channel. The resulting EM fields are then related to the integral of a return stroke based term over the lightning channel height at a point in time. The Heidler function is a popular choice for a channel base current model, as it best models the physical properties of lightning current [8]. As discussed in Section V, the EM equations have a term that depends directly on the time integral of the channel base current. This term cannot be analytically resolved when using a Heidler function, which adds additional complexity to calculating the EM fields. When looking at a point far from the lightning channel, this term can often be neglected, however this is not the case for a nearby location ($50\,\mathrm{m}$ to $500\,\mathrm{m}$ [9]) as it would have a significant effect on a distribution line LIOV [10], [11]. One possible solution is to use a channel base current model that has an analytical integral solution, which is presented in Section IV.

A relatively new method to lightning EM calculation is the Finite-Difference Time-Domain (FDTD) technique, which

is well suited for transient field calculation in the time domain [12], [13]. The method changes Maxwell's differential equations into difference equations. These difference equations (finite changes in space and time) decompose the evaluation space into cells/cubes (2D/3D). The size of the evaluation space is defined by the channel height and the distance to the observation point. The EM fields in each of these cells (known as a Yee cell) [12], [13] is evaluated by previous values, and surrounding cell values. This process has a number of important benefits, such as including complex simulation environments and materials such as the ground plane. It has the disadvantage of being computationally intensive as well as having a number of numerical errors particularly around the boundary conditions.

An novel solution for LEMP calculations has been suggested by Sartori *et al* [14], where a combination of the Finite Antenna method and the standard FDTD method is used. This "hybrid" method will be referred to as the "Single Cell FDTD" method, as only one Yee cell is evaluated.

III. Assumptions

The geometry of the problem is depicted in Figure 1. This shows that the lightning channel is modeled as being perfectly straight, and perpendicular to a flat ground plane. In these simulations the ground plane is considered as a perfect conductor. This assumption is unrealistic given that ground has a finite conductivity, however, for simplicity the effect is be ignored.

Figure 1 also shows that there are no obstacles in the simulation space. The assumption being that the surrounding area is "ideal free space" where the permeability and permittivity are μ_0 and ϵ_0. This results in a propagation velocity of c (the speed of light). The effects of ground plane reflections are considered by an image channel below the ground plane. This figure also shows that Spherical co-ordinates are used in defining the vector directions.

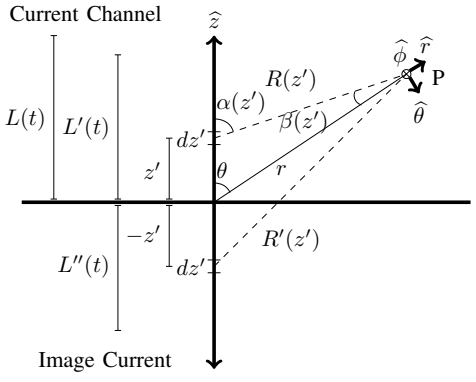


Fig. 1. Geometry of EM field calculations.

IV. Channel Base Current Model

The current impulse (describing the channel base current) used in this paper is described by Equation 1 and 2. This current impulse has been used in a number of other EM calculations [7], [15], [16], and its parameter values have been chosen to represent a typical subsequent return stroke. Although first return strokes have a higher peak current, subsequent return strokes have a faster rise time which has a significant effect on the LEMP.

$$i(t) = \frac{I_1}{\eta} \cdot \left(\frac{(\frac{t}{\tau_1})^2}{(\frac{t}{\tau_1})^2 + 1} \right) \cdot e^{-\frac{t}{\tau_2}} + I_2(e^{\frac{-t}{\tau_3}} - e^{\frac{-t}{\tau_4}}) \quad (1)$$

$$i(t) = \frac{I_3}{\eta} \cdot \left(1 - e^{-\omega_0 t} \left(\sum_{j=0}^{n_a} \frac{\omega_0^j t^j}{j!} \right) \right) \cdot e^{-\frac{t}{\tau_2}} + I_2(e^{\frac{-t}{\tau_3}} - e^{\frac{-t}{\tau_4}}) \quad (2)$$

Where

z'	=	Channel height		
v	=	Velocity of wavefront	=	$150\,\mathrm{m/(\mu s)}$
I_1	=	Peak Current	=	$9.9\,\mathrm{kA}$
I_2	=	Peak Current	=	$7.5\,\mathrm{kA}$
I_3	=	Peak Current	=	$9.55\,\mathrm{kA}$
η	=	Scaling Factor	=	0.845
n_a	=	Steepness Factor	=	3
ω_0	=	Rise Time Constant	=	$41.66\,/(\mu s)$
$\tau 1$	=	Time Constant	=	$0.072\,\mu s$
$\tau 2$	=	Time Constant	=	$5\,\mu s$
$\tau 3$	=	Time Constant	=	$100\,\mu s$
$\tau 4$	=	Time Constant	=	$6\,\mu s$

Equation 1 is a combination of a Heidler function with a double exponential function [15]. Equation 2 replaces the Heidler function component with the Terespolsky function (Heidler function approximation) [17], [18], [19]. The Terespolsky function allows for the integral solution to be found analytically. The differential and integral forms of Equation 2 are shown in Equation 3 and 4 respectively.

$$i'(t) = \frac{I_1}{\eta} \cdot \left[\frac{e^{-\omega_0 t} \omega_0^{n_a+1} t^{n_a}}{n_a!} - \frac{1}{\tau_2} + \frac{e^{-\omega_0 t}}{\tau_2} \left(\sum_{j=0}^{n_a} \frac{\omega_0^j t^j}{j!} \right) \right] \cdot e^{-\frac{t}{\tau_2}} + I_2(-\frac{1}{\tau_3} e^{\frac{-t}{\tau_3}} + \frac{1}{\tau_4} e^{\frac{-t}{\tau_4}}) \quad (3)$$

$$\int i(t)\, dt = \frac{I_1 \tau_2 e^{-t(\frac{1}{\tau_2} + \omega_0)}}{\eta} \cdot \left(-e^{-\omega_0 t} + \sum_{i=0}^{n_a} \sum_{j=0}^{i} \frac{\omega_0^i \tau_2^j t^{i-j}}{(i-j)!(\tau_2 \omega_0 + 1)^{j+1}} \right) \cdot e^{-\frac{t}{\tau_2}} + I_2(-\tau_3 e^{\frac{-t}{\tau_3}} + \tau_4 e^{\frac{-t}{\tau_4}}) + C \quad (4)$$

The current impulse in Equation 1 (Heidler version) and the approximation in Equation 2 (Terespolsky version) have minor differences. Figure 2 shows the differential plot of these equations, and although there is a small time shift, the peak

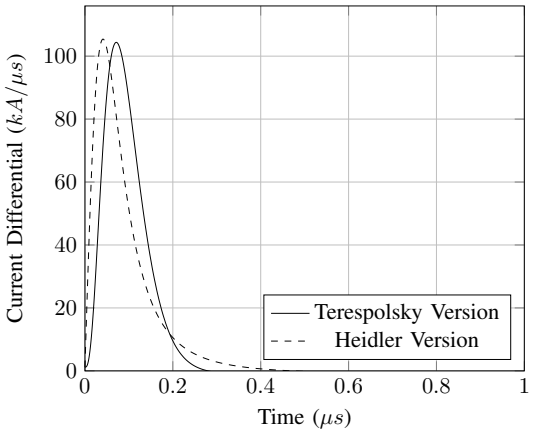


Fig. 2. Current impulse differential plot comparison.

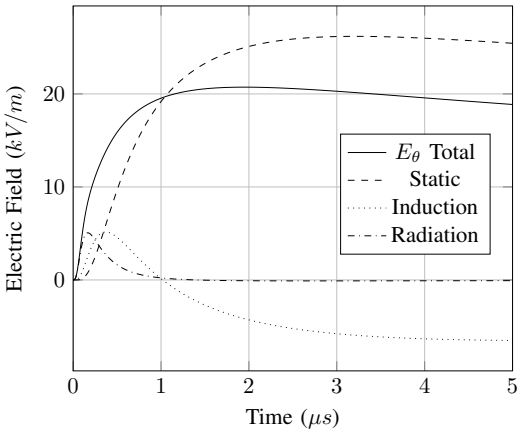


Fig. 3. E_z field components on the ground plane.

differential values (most significant in LEMP calculations) have approximately the same value.

The return stroke model used in this paper has been simplified to a lossless transmission line with a propagation velocity less than c. The current at any point on the channel is described by the channel base current at an earlier time, as seen in Equation 5. The unit step (U) ensures that the current is zero above the propagating wave front.

$$i(z',t) = i(t - \frac{z'}{v}) \cdot U(t - \frac{z'}{v}) \tag{5}$$

There are a number of return stroke models that are better suited for lightning simulations [20], such as the MTLE model [21], which can easily be included in Equation 5 for improved results.

V. Finite Antenna Method

The full mathematical derivation and equations to describe E_r, E_θ, and B_ϕ at a point above the ground plane (as seen in Figure 1) will not be derived in this paper, but are discussed in detail in [6] and [7]. A simplified case of the EM fields at a point on the ground plane is described by Equations 6 and 7, where $\theta = \pi/2$, $\widehat{z} = -\widehat{\theta}$, and $E_r = 0$ as described in [3], [4], [5].

$$\begin{aligned} &E(r,\theta,t)(2\pi\epsilon_0) = \\ &+\int_0^{L'(t)} \frac{3z^2 - R^2(z')}{R^5(z')} \left\{ \int_{t_b}^{t} i(0, \tau - \frac{R(z')}{c} - \frac{z'}{v}) d\tau \right\} dz'\widehat{z} \\ &+\int_0^{L'(t)} \frac{3z^2 - R^2(z')}{cR^4(z')} \cdot i(0, t - \frac{R(z')}{c} - \frac{z'}{v}) dz'\widehat{z} \\ &-\int_0^{L'(t)} \frac{r^2}{c^2R^3(z')} \frac{\partial i(0, t - \frac{R(z')}{c} - \frac{z'}{v})}{\partial t} dz'\widehat{z} \end{aligned} \tag{6}$$

$$\begin{aligned} &B(r,\theta,t) = \\ &+\frac{1}{2\pi\epsilon_0 c^2} \int_0^{L'(t)} \frac{r}{R^3(z')} i(0, t - \frac{R(z')}{c} - \frac{z'}{v}) dz'\widehat{\phi} \\ &+\frac{1}{2\pi\epsilon_0 c^2} \int_0^{L'(t)} \frac{r}{cR^2(z')} \frac{\partial i(0, t - \frac{R(z')}{c} - \frac{z'}{v})}{\partial t} dz'\widehat{\phi} \end{aligned} \tag{7}$$

Equation 6 has three field components described as the electrostatic field (current integral term), induction field (current term) and the radiation field (current differential term). Equation 7 has an induction field component (current term), and a radiation field component (current differential term). The equations describing E_r, E_θ, and B_ϕ at a point above the ground plane have a similar format to these equations. Calculating the electrostatic components of both E_r and E_θ requires solving the integral of the channel base current used in the return stroke model. Equation 1 contains a Heidler function, which cannot be resolved analytically. Which therefore adds additional complexity to solving the electrostatic field components of E_r and E_θ. A suggested solution to this is to use Equation 2, which can be analytically solved, and therefore simplifies one aspect of the calculation. Figure 3 shows the E_z field, and its components, calculated using Equation 2, at a distance of $r = 50\,\mathrm{m}$ with $\theta = \pi/2$. Figure 3 is comparable to the figures in [6], [7]. This demonstrates the validity in using Equation 2 as an approximation to Equation 1.

VI. Single Cell FDTD Method

The Finite-Difference Time-Domain (FDTD) method is a computational electromagnetic technique to evaluate EM fields in the time domain. A common method of modeling lightning electromagnetic waves in the FDTD is to describe the area around the lightning channel in cylindrical co-ordinates. This allows the entire evaluation space to be described in the two-dimensional $r - z$ plane [22]. This process can be computationally intensive, given that every cell in the plane needs to be evaluated at each time step of the simulation.

Additionally there is a stability criterion for the FDTD that often requires small time steps and cell sizes.

This section describes the Single Cell FDTD method, which is a combination of standard FDTD theory with the mathematics of the Finite Antenna method. When evaluating a single Yee cell, it is possible to calculate the electric field components from only the magnetic field components around the cell. The benefits of this method will become clear in the results. What follows is a derivation of the FDTD equations for a spherical co-ordinate system, which is different from the normal Cartesian or cylindrical co-ordinate systems. This system was chosen to match the mathematics from the Finite Antenna method which uses the same co-ordinate system. This co-ordinate system may have limited application to full FDTD applications due to the difficulty of adding the excitation sources along a lightning channel.

$$\nabla \times \overline{E} = -\frac{\partial \overline{B}}{\partial t} \qquad \nabla \times \overline{B} = (\mu_0 \epsilon_0)\frac{\partial \overline{E}}{\partial t} \tag{8}$$

Where

$\overline{E}$ = $\widehat{a}_r E_r + \widehat{a}_\theta E_\theta + \widehat{a}_\phi E_\phi$

$\overline{B}$ = $\widehat{a}_r B_r + \widehat{a}_\theta B_\theta + \widehat{a}_\phi B_\phi$

Consider Equation 8 which shows Faraday's Law, and Ampere's Law respectively (Maxwell's differential equations with no exciting current $\overline{J}$). Given the nature of the geometry shown in Figure 1, it is clear that the model is symmetrical around the $\widehat{z}$ axis, and therefore at any angle of ϕ. Therefore it is sufficient to review the problem in the $r-\theta$ plane, and set $\frac{\partial}{\partial \phi} = 0$. Using the spherical co-ordnate system [13], Equation 8 is expanded into a Transverse Electric (TE) and Transverse Magnetic (TM) form. The TE equations are shown in Equation 9. The TE fields describe E_r, E_θ and B_ϕ, which are the same as Section V.

$$\frac{\partial E_r}{\partial t} = \frac{1}{\mu_0 \epsilon_0 . r . Sin(\theta)} \frac{\partial}{\partial \theta}(B_\phi . Sin(\theta)) \tag{8a}$$

$$\frac{\partial E_\theta}{\partial t} = \frac{-1}{\mu_0 \epsilon_0 r} \frac{\partial}{\partial r}(B_\phi . r) \tag{8b}$$

$$\frac{\partial B_\phi}{\partial t} = -\frac{1}{r}\left[\frac{\partial}{\partial r}(E_\theta . r) - \frac{\partial}{\partial \theta} E_r\right] \tag{8c}$$

Figure 4 shows the Yee cell describing a cell in the TE spherical grid [13]. From this Yee cell, Equation 9 is written in the discrete second order FDTD form shown in Equations 9 and 10 (without the B_ϕ term).

$$E_r\Big|_{(i+\frac{1}{2},j)}^{n+1} = E_r\Big|_{(i+\frac{1}{2},j)}^{n} + \frac{c^2.\Delta t}{r_{i+\frac{1}{2}}.Sin(\theta_j).\Delta\theta} \cdot \left[B_\phi\Big|_{(i+\frac{1}{2},j+\frac{1}{2})}^{n+\frac{1}{2}} .Sin(\theta_{j+\frac{1}{2}}) - B_\phi\Big|_{(i+\frac{1}{2},j-\frac{1}{2})}^{n+\frac{1}{2}} .Sin(\theta_{j-\frac{1}{2}})\right] \tag{9}$$

$$E_\theta\Big|_{(i,j+\frac{1}{2})}^{n+1} = E_\theta\Big|_{(i,j+\frac{1}{2})}^{n} - \frac{c^2.\Delta t}{r_i.\Delta r} \cdot \left[B_\phi\Big|_{(i+\frac{1}{2},j+\frac{1}{2})}^{n+\frac{1}{2}} .r_{i+\frac{1}{2}} - B_\phi\Big|_{(i-\frac{1}{2},j+\frac{1}{2})}^{n+\frac{1}{2}} .r_{i-\frac{1}{2}}\right] \tag{10}$$

The notation used is from [13], and is explained in Equation 11. As seen in Equations 6 and 7, the equation for the magnetic field is less computationally intensive to evaluate than the equations for the electric field. Equations 9 and 10 show that E_r and E_θ can be calculated by only the B_ϕ fields surrounding the cell, and previous E fields.

$$\begin{aligned} E_\theta\Big|_{(i,j+\frac{1}{2})}^{n+1} &= E_\theta(r_c, \theta_c, t_c) \\ &= E_\theta(i\Delta r, j\Delta\theta + \frac{\Delta\theta}{2}, n\Delta t + \Delta t) \end{aligned} \tag{11}$$

Where

Radius at the observation point = r_{obs} = $(i).\Delta r$

Angle at the observation point = θ_{obs} = $(j).\Delta\theta$

Time at the observation point = t_{obs} = $(n).\Delta t$

In a standard FDTD approach the observation point would change to every cell in the plane, however, in this example it stays fixed at the observation point "P" in Figure 1. From Equations 9 and 10 it is clear that B_ϕ needs to be calculated at three distinct locations (shown in Figure 4). Equation 12 (the expansion of Equation 7) is used to calculate the magnetic field at these three locations, which is then used in Equations 9 and 10 to calculate the electric fields.

$$\begin{aligned} &B(r,t)(4\pi\epsilon_0 c^2) = \\ &+\int_0^{L'(t)} \frac{r.Sin\theta}{R^3(z')} i(0, t - \frac{R(z')}{c} - \frac{z'}{v}) dz' \widehat{\phi} \\ &+\int_0^{L'(t)} \frac{r.Sin\theta}{cR^2(z')} \frac{\partial i(0, t - \frac{R(z')}{c} - \frac{z'}{v})}{\partial t} dz' \widehat{\phi} \\ &+\int_0^{L''(t)} \frac{r.Sin\theta}{R^3(-z')} i(0, t - \frac{R(-z')}{c} - \frac{z'}{v}) dz' \widehat{\phi} \\ &+\int_0^{L''(t)} \frac{r.Sin\theta}{cR^2(-z')} \frac{\partial i(0, t - \frac{R(-z')}{c} - \frac{z'}{v})}{\partial t} dz' \widehat{\phi} \end{aligned} \tag{12}$$

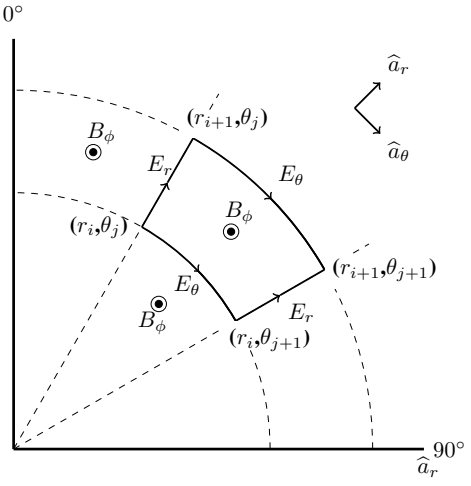


Fig. 4. Yee cell in spherical co-ordinates.

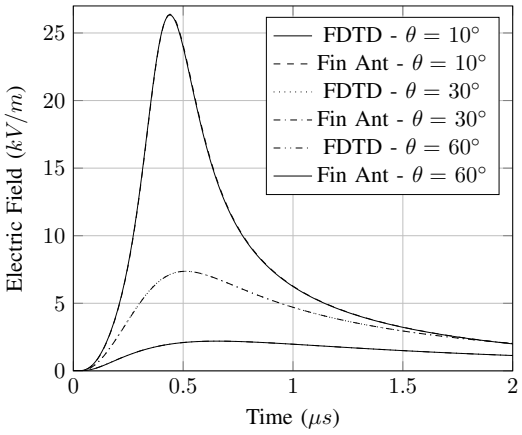


Fig. 5. Radial E Field - R = 100 m, Δr =0.5 m, $\Delta\theta$ = 0.5°.

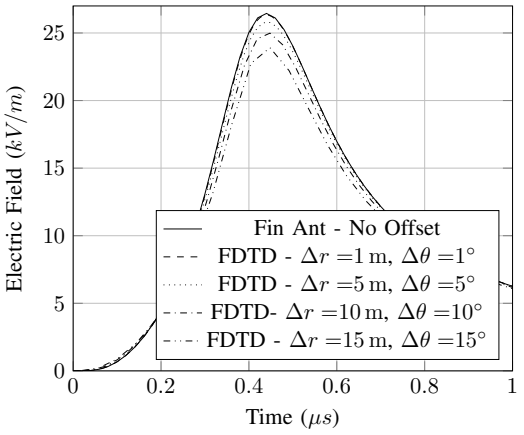


Fig. 6. Radial E Field - R = 100 m, θ = 10°.

The choice of Δr, $\Delta\theta$, and Δt has a significant effect on the accuracy of the Single Cell FDTD results, as well as the stability of the results. The stability criteria of the Yee cell in Figure 4 is described by Equation 13, which relates the distance traveled by a wave front and the speed of propagation (c) [12]. As seen in Figure 4, the Yee cell is not linear throughout the plane [13], and changes depending on the radius of the observation point. This is accounted for by considering the arc distance $r(\Delta\theta)$.

$$\Delta t \leq \frac{1}{c\sqrt{\frac{1}{\Delta i^2} + \frac{1}{\Delta j^2}}} \tag{13}$$

Where: $\Delta i = \Delta r \quad \Delta j = r(\Delta\theta)$

VII. Simulation Testing

A number of simulations were run (at different distances and angles) to test the Single Cell FDTD method, and the Finite Antenna method was used to verify the results. The current impulse used in the simulations is described by Equation 2 (Terespolsky function), which was chosen for its ease of implementation in the Finite Antenna method. However for the Single Cell FDTD method and Equation 12 it is easier to implement Equation 1.

Figure 5 shows the E_r fields calculated at a radial distance of 100 m, and a range of vertical offset angles. The Yee cell used Δr =0.5 m and $\Delta\theta$ = 0.5°. The figure shows the fields calculated by the Finite Antenna method, as well as the Single Cell FDTD method, and the results are identical (also see [7]). The same tests were run at distances of 50 m to 100 km, at angles of 0° to 90° and the results were consistently the same with negligible errors.

In all these tests the E_r and E_θ values in both methods are calculated with the appropriate spacial offsets described in Equations 9 and 10, where Δr =0.5 m and $\Delta\theta$ = 0.5°. Figure 6 shows the E_r fields calculated at a radial distance of 100 m at a angle of 10° for different Yee cell sizes. As expected, the resulting fields from the Single Cell FDTD method are different to the field predicted by the Finite Antenna method due to the location offset of Δr and $\Delta\theta$. However, when compared to the Finite Antenna field calculated at those specific offset values, it was found that there is error in the Single Cell FDTD results. Figure 7 shows the normalized error in time between the Finite Antenna and the Single Cell FDTD results. These errors are primarily caused by a time offset created by the Yee cell. Ideally the the Yee cell should be kept as small as possible, however as seen in Table I, there is a trade-off in computation time. As the Yee cell is made bigger, there are fewer time steps where Equation 12 needs to be evaluated. However as seen in Figure 7 there are higher errors at larger Yee cells.

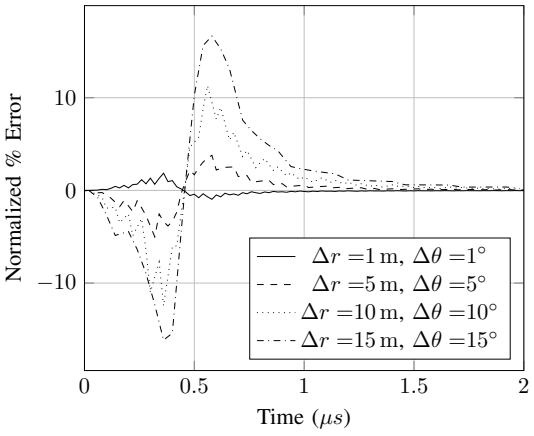


Fig. 7. Normalized Radial E Field Error - R = 100 m, θ = 10°.

TABLE I
STABILITY CRITERIA - YEE CELL AT R=100 m.

Δr (m)	$\Delta\theta$ (°)	Δt (ns)	**Time Steps in** 10 µs
0.01	0.01	0.0029	345514
0.1	0.1	0.0289	34551
0.5	0.5	0.1447	6910
1	1	0.289	3455
5	5	1.447	691
10	10	2.894	345
15	15	4.341	230

VIII. DISCUSSION

As seen in Section VII the Finite Antenna method and Single Cell FDTD method compare well when a small Yee cell is used. In the standard FDTD method application, it is recommended that Δi and Δj are chosen as a tenth of a wavelength, or in the case of lightning, at least a tenth of the distance propagated by the impulse range of interest. This distance is typically in the order of $100\,\mathrm{m}$ to $300\,\mathrm{m}$ (maximum), which is not acceptable for the Single Cell FDTD method at close distances. Another method for choosing Δi and Δj could be to choose the Δt small enough to accurately represent the field in the time range of interest. At locations close to the lightning channel, a minimum of 345 steps per µs is necessary. Note also that Δr and $\Delta\theta$ do not have to be the same value.

The tests run on the different Yee cells in Section VII focused primarily at locations close to the lightning channel. At distances far from the channel the size of the Yee cell will grow due to $\Delta j = r_{obs}(\Delta\theta)$, and therefore the choice of Δt will be limited by the choice of $\Delta\theta$. Another consideration at these distances is that the electrostatic component in the E field calculations may become negligible. Under this situation the computational requirements of the two methods become equivalent. To evaluate the fields at a point, the Finite Antenna method would calculate the E_r, E_θ and B_ϕ fields at one unique location, as compared to the Single Cell FDTD method that would need to calculate the B_ϕ fields at three unique locations.

In comparison to standard FDTD methods, the Single Cell FDTD approach has a number of advantages, as it does not suffer from boundary conditions (and the inherent numerical errors) [13]. It also does not rely on evaluating every Yee cell in the area of interest. To evaluate a LIOV on a distribution line, the Single Cell FDTD approach could be used to evaluate the EM fields at the Yee cells directly around the line, rather than all the cells between the lightning channel and the line (standard FDTD).

A disadvantage to both methods is the difficulty of including complex simulation environments, such as real ground planes, variable lightning channels, scattering and obstacles. In contrast, the standard FDTD approach is well suited to handle these complex modeling criteria, and allows the inclusion of real world characteristics.

IX. CONCLUSION

Two methods for modeling the lightning EM fields near a lightning channel have been presented and verified. The first method used a Heidler function approximation (Terespolsky function) with the classic Finite Antenna method. By using this current impulse function it was shown that resolving the electrostatic field component of the electric fields was simplified due to the analytical integral solution available from the Terespolsky function. The second method used a combination of standard FDTD theory, with the magnetic field equations from the Finite Antenna method, and evaluated the electric fields in a single Yee cell. The results from this Single Cell FDTD method compare well to the Finite Antenna method when an appropriately sized Yee cell is chosen. The benefit of the Single Cell FDTD method is that it is not affected by the boundary condition requirements that standard FDTD is bound by, and it has lower computational requirements.

ACKNOWLEDGMENT

The authors thank Eskom for the support of the Lightning/EMC Research Group through the TESP programme. Thanks are extended to the department of Trade and Industry (DTI) for THRIP funding as well as to the National Research Foundation (NRF) for direct funding of the Research Group.

REFERENCES

[1] C. A. Nucci and F. Rachidi, *The Lightning Flash 2nd Edition*, ser. Power and Energy Series 69. Institution of Engineering and Technology (IET), 2014, ch. 12 - Interaction of electromagnetic fields generated by lightning with overhead electrical networks, pp. 559–609.

[2] Y. Baba and V. A. Rakov, *Lightning Electromagnetics*, ser. Power and Energy Series 62. Institute of Engineering and Technology, 2012, ch. 8 - Electromagnetic models of lightning return strokes, pp. 263–313.

[3] M. A. Uman, D. K. Mclain, and E. P. Krider, "The electromagnetic radiation from a finite antenna," *American Journal of Physics*, vol. 43, pp. 33–38, January 1975.

[4] M. J. Master and M. A. Uman, "Transient electric and magnetic fields associated with establishing a finite electrostatic dipole," *American Journal of Physics*, vol. 51, no. 2, pp. 118–126, February 1983.

[5] M. A. Uman, "Lightning return stroke electric and magnetic fields," *Journal of Geophysical Research - Atmospheres*, vol. 90, no. D4, pp. 6121–6130, June 1985.

[6] R. Thottappillil and V. A. Rakov, "On different approaches to calculating lightning electric fields," *Journal of Geophysical Research*, vol. 106, no. D13, pp. 14 191–14 205, July 2001.

[7] R. Thottappillil, *The Lightning Flash 2nd Edition*, ser. Power and Energy Series 69. Institution of Engineering and Technology (IET), 2014, ch. 8 - Computation of electromagnetic fields from lightning discharge, pp. 351–403.

[8] *Protection Against Lightning - Part 1: General Principles*, IEC Std., 2006.

[9] G. Diendorfer, "Induced voltage on an overhead line due to nearby lightning," *IEEE Transactions on Electromagntic Compatibility*, vol. 32, no. 4, pp. 292–299, November 1990.

[10] S. Rusck, "Induced lightning over-voltages on power-transmission lines with special reference to the over-voltage protection of low-voltage networks," Master's thesis, Transactions of the Royal Institute of Technology, 1958.

[11] P. Sewkumar, "Modeling the effect of adjacent lightning strikes to bare overhead medium voltage lines in South Africa," Master's thesis, Faculty of Engineering and the Built Environment, School of Electrical and Electronic Engineering, University of the Witwatersrand, South Africa, 2001.

[12] K. S. Yee, "Numerical solution of initial boundry value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. AP-14, no. 3, pp. 302–307, May 1966.

[13] U. S. Inan and R. A. Marshall, *Numerical Electromagnetics: The FDTD Method*. Cambridge University Press, 2011.

[14] C. A. F. Sartori and J. R. Cardoso, "An Analytical-FDTD method for near LEMP calculation," *IEEE Transactions on Magnetics*, vol. 36, no. 4, pp. 1631–1634, July 2000.

[15] C. A. Nucci, G. Diendorfer, M. A. Uman, F. Rachidi, M. Ianoz, and C. Mazetti, "Lightning return stroke current models with specified channel-base current: A review and comparison," *Journal of Geophysical Research*, vol. 95, no. D12, pp. 20 395–20 408, November 1990.

[16] V. Cooray, *The Lightning Flash*, ser. Power Series 34, V. Cooray, Ed. Institution of Engineering and Technology (IET), 2003.

[17] B. R. Terespolsky, "An approximation to the heidler function with an analytical integral for engineering applications using lightning currents," Master's thesis, School of Electrical and Information Engineering, University of the Witwatersrand, South Africa, 2015.

[18] B. R. Terespolsky and K. J. Nixon, "Developing an approximation to the Heidler Function - with an analytical transformation into the frequency domain." *International Conference on Lightning Protection*, pp. 1134–1138, 2014.

[19] C. W. I. McAfee and K. J. Nixon, "Lightning return stroke modeling with reference to lightning electromagnetic fields," *South African Universities Power and Energy Conference (SAUPEC), Johannesburg*, January 2015.

[20] V. Cooray, *The Lightning Flash 2nd Edition*, ser. Power and Energy Series 69. Institution of Engineering and Technology (IET), 2014, ch. 9 - Return stroke models with special attention to engineering applications, pp. 405–475.

[21] C. A. Nucci, C. Mazetti, F. Rachidi, and M. Ianoz, "On lightning return stroke models for LEMP calculations," *International Conference on Lightning Protection*, no. 4.7, pp. 463–470, 1988.

[22] Y. Baba and V. A. Rakov, "Applications of the FDTD method to lightning electromagnetic pulse and surge simulations," in *International Conference on Lightning Protection*. Shanghai, China, 2014, pp. 2084–2098.

Appendix C

2D Spherical Finite Antenna Code

Listing C.1: main.cpp

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;

#include "FINANTCyl2D.h"

int main(int argc, char const *argv[]){

  FINANTCyl2D lightningStroke("2D_FINANT_Settings.csv",argc,argv);
  lightningStroke.PrintMinimalSimParameters();
  lightningStroke.evaluateFINANTintegral();
  return 0;
}
```

Listing C.2: FINANTCyl2D.h

```cpp
#include <cmath>
#include <math.h>
#include <iostream>
#include <fstream>
#include <future>
#include <thread>
using namespace std;

#include "BfieldP.h"
#include "EfieldR.h"
#include "EfieldT.h"

class FINANTCyl2D{

public:
  FINANTCyl2D(string settingsFile, int argc, char const *argv[]);
  void setSimulationParameters(string settingsFile);
  void PrintMinimalSimParameters();
  void evaluateFINANTintegral();

private:
```

```
  double vp;                      // RS Constants:
  double vchan;
  double lamdac;
  int currentDecay;

  double I1, I2, I3;              // Terespolsky function constants:
  double eta;
  double tau1, tau2, tau3, tau4;
  int Na;
  double omega0;

  double rObs;                    // User Set Simulation Parameters:
  double observationAngleDeg;
  double theta;
  int integrationSteps;
  double deltaT;
  double simRunTime;

  double timeToReachObservationPoint; // Time variables:
  int nSteps;

  double BpFieldHolder;           // Field value holders:
  double ErFieldHolder;
  double EtFieldHolder;

  const double pi = 4.0 * atan( 1.0 );  // Fixed Constants:
  const double c = 299792458;
  const double mu = 4 * pi * pow( 10.0, -7.0 );
  const double epsilon = 8.8541878176 * pow( 10.0, -12.0 );

  char partATestID[128];        // Data file field names:
  char partBTestID[128];
  char BpFieldcurvename[128];
  char ErFieldcurvename[128];
  char EtFieldcurvename[128];
  char ExFieldcurvename[128];
  char EzFieldcurvename[128];
};
```

Listing C.3: FINANTCyl2D.cpp

```
#include <algorithm>
#include "FINANTCyl2D.h"

// Constructor takes inputs being the Settings file followed by the command line arguments.
FINANTCyl2D::FINANTCyl2D(string settingsFile, int argc, char const *argv[]){

    // Read in settings from settings file.
    setSimulationParameters(settingsFile);

    // Character strings for file naming.
    sprintf(partATestID,"FinAnt_R%05gA%04g",rObs*100,observationAngleDeg*100);
    sprintf(partBTestID,"kstep%02g_dt%03g",integrationSteps/1000.0,deltaT*pow(10,9)*10);

    // Creating Output File names for the different fields. These files are opened, written
    // to and closed in the main evaluateFINANTintegral function. Also Note that this is
    // hard coded to save files to the folder: "Simulations" .
    sprintf(BpFieldcurvename,"./Simulations/%sBpField%s.curve",partATestID,partBTestID);
    sprintf(ErFieldcurvename,"./Simulations/%sErField%s.curve",partATestID,partBTestID);
    sprintf(EtFieldcurvename,"./Simulations/%sEtField%s.curve",partATestID,partBTestID);
    sprintf(ExFieldcurvename,"./Simulations/%sExField%s.curve",partATestID,partBTestID);
    sprintf(EzFieldcurvename,"./Simulations/%sEzField%s.curve",partATestID,partBTestID);

    theta = (observationAngleDeg/180)*pi;
    omega0 = ( double )Na / tau1;
    timeToReachObservationPoint = rObs/c;
```

```
26      nSteps = int(simRunTime/deltaT);
27  }
28
29  // This function is passed a file name string and extracts the required simulation variables
30  // from the file and assigns them to the relevant variables of the class. Care must be
31  // taken to ensure that the settings file is formatted correctly.
32  void FINANTCyl2D::setSimulationParameters(string settingsFile){
33
34      ifstream inputFromFile(settingsFile, ios::in);
35
36      if(!inputFromFile){
37        cerr << "File could not be opened!" << endl;
38        exit(1);
39      }
40
41      string UnitName, Value, string Units;
42
43      while(getline(inputFromFile, UnitName, ',')){
44
45        // This next line of code removes the newline character from my settings file when
46        // it is read into the variables. This was only really an issue with the UnitName
47        // variable. This code required the "#include <algorithm>" library.
48        UnitName.erase(std::remove(UnitName.begin(), UnitName.end(), '\n'), UnitName.end());
49        getline(inputFromFile, Value, ',');
50        getline(inputFromFile, Units, ',');// This variable does not get used.
51
52        if(UnitName == "vp"){vp = std::stof( Value );}
53        else if(UnitName == "vchan"){vchan = std::stof( Value );}
54        else if(UnitName == "deltaT"){deltaT = std::stof( Value ) * pow(10,-9);}
55        else if(UnitName == "simRunTime"){simRunTime = std::stof( Value) * pow(10,-9);}
56        else if(UnitName == "tau1"){tau1 = std::stof( Value) * pow(10,-6);}
57        else if(UnitName == "tau2"){tau2 = std::stof( Value) * pow(10,-6);}
58        else if(UnitName == "tau3"){tau3 = std::stof( Value) * pow(10,-6);}
59        else if(UnitName == "tau4"){tau4 = std::stof( Value) * pow(10,-6);}
60        else if(UnitName == "lamdac"){lamdac = std::stof( Value);}
61        else if(UnitName == "currentDecay"){currentDecay = std::stof( Value);}
62        else if(UnitName == "I1"){I1 = std::stof( Value);}
63        else if(UnitName == "I2"){I2 = std::stof( Value);}
64        else if(UnitName == "I3"){I3 = std::stof( Value);}
65        else if(UnitName == "eta"){eta = std::stof( Value);}
66        else if(UnitName == "Na"){Na = std::stof( Value);}
67        else if(UnitName == "rObs"){rObs = std::stof( Value);}
68        else if(UnitName == "observationAngleDeg"){observationAngleDeg = std::stof( Value);}
69        else if(UnitName == "integrationSteps"){integrationSteps = std::stof( Value);}
70        else{cout << "Variable "<< UnitName << " : " << Value <<" was not assigned." << endl;}
71      }
72  }
73
74  // This function outputs minimal Sim variables for batch simulation scripts tests.
75  void FINANTCyl2D::PrintMinimalSimParameters(){
76
77      cout << "*******************************************" << endl;
78      cout << "requested rObs: " << rObs << ". Observation point angle: "  << ↩
            observationAngleDeg << endl;
79      cout << "intSteps -deltaT -simTime" << endl;
80      cout << integrationSteps << "   " <<  deltaT << "    " << simRunTime << endl;
81  }
82
83  // This is the main workhorse of the class. Using the parameters given by the settings file,
84  // or the command line arguments, this function creates a field object for the Magnetic PHI
85  // field, Electric radial, and electric theta fields. These are all WRT a cylindrical
86  // co-ordinate system. Each of these fields are then evaluated over a time period and the
87  // resulting field waveforms are written to a .curve file for viewing in VisIt. I also have
88  // code to convert curve to CSV. What is really cool about this function is that each of
89  // the fields is calculated in its own thread, which effectively means that the fields are
90  // evaluated simultaneously. This has a significant improvement in program runtime. This
91  // function also produces the Ex and Ey field components for Cartesian co-ordinates.
92  void FINANTCyl2D::evaluateFINANTintegral(){
```

```
93
94      // Create field objects:
95        BfieldP simBfieldP(vchan, c, lamdac, I3, I2, eta, tau2, tau3, tau4, omega0, Na, ↩
              currentDecay);
96        EfieldR simEfieldR(vchan, c, lamdac, I3, I2, eta, tau2, tau3, tau4, omega0, Na, ↩
              currentDecay);
97        EfieldT simEfieldT(vchan, c, lamdac, I3, I2, eta, tau2, tau3, tau4, omega0, Na, ↩
              currentDecay);
98
99      // Open the output data field files:
100       ofstream observationPointBpDataFile(BpFieldcurvename);
101       ofstream observationPointErDataFile(ErFieldcurvename);
102       ofstream observationPointEtDataFile(EtFieldcurvename);
103       ofstream observationPointExDataFile(ExFieldcurvename);
104       ofstream observationPointEzDataFile(EzFieldcurvename);
105     // Write the output file headers:
106       observationPointBpDataFile << "#FinAnt_BfieldP" << endl;
107       observationPointErDataFile << "#FinAnt_EfieldR" << endl;
108       observationPointEtDataFile << "#FinAnt_EfieldT" << endl;
109       observationPointExDataFile << "#FinAnt_EfieldX" << endl;
110       observationPointEzDataFile << "#FinAnt_EfieldZ" << endl;
111
112     cout << "*******************************************" << endl;
113     cout << "Starting Simulation Time Steps: " << endl;
114
115     // This is where I iterate over all the time steps:
116     for(int n=0; n != nSteps; n++){
117
118       // Progress indicator:
119       cout << "Progress " << ((n*deltaT)/simRunTime)*100.00 << "\%  \r" ;
120       std::cout.flush();
121
122       // Create a thread for each field. These threads return the field at a point in time.
123       future< double > BpFieldThread = async(std::launch::async, &BfieldP::integrateZ, &↩
              simBfieldP, n*deltaT , theta, rObs, integrationSteps);
124       future< double > ErFieldThread = async(std::launch::async, &EfieldR::integrateZ, &↩
              simEfieldR, n*deltaT , theta, rObs, integrationSteps);
125       future< double > EtFieldThread = async(std::launch::async, &EfieldT::integrateZ, &↩
              simEfieldT, n*deltaT , theta, rObs, integrationSteps);
126
127       // Wait for threads to complete:
128       BpFieldHolder = BpFieldThread.get();
129       ErFieldHolder = ErFieldThread.get();
130       EtFieldHolder = EtFieldThread.get();
131
132
133       // Write field data to output files.
134         // The "if" check ensures that data is only written from the point when the EM
135         // field first reaches the observation point. This step also does the co-ordinate
136         // conversion for Spherical to Cartesian field components. Note that all these
137         // fields are measured on the Y=0/PHI=0 axis. I chose this because it simplifies
138         // co-ordinate conversion for the third direction in spherical. In this case,Ey=Ephi
139         if(n*deltaT >= timeToReachObservationPoint){
140           observationPointBpDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) ↩
                  << " " << BpFieldHolder << endl;
141           observationPointErDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) ↩
                  << " " << ErFieldHolder << endl;
142           observationPointEtDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) ↩
                  << " " << EtFieldHolder << endl;
143
144           observationPointExDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) ↩
                  << " " << (cos(theta)*EtFieldHolder + sin(theta)*ErFieldHolder) << endl;
145           observationPointEzDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) ↩
                  << " " << (-1*sin(theta)*EtFieldHolder + cos(theta)*ErFieldHolder) << endl;
146         }
147
148     }
149
```

```
150     cout << endl;
151     cout << "Simulation Complete." << endl;
152     cout << "*********************************************" << endl;
153     cout << endl;
154 
155     // Close Data file:
156     observationPointBpDataFile.close();
157     observationPointEtDataFile.close();
158     observationPointErDataFile.close();
159     observationPointExDataFile.close();
160     observationPointEzDataFile.close();
161 }
```

Listing C.4: EfieldT.h

```
1  #include <iostream>
2  #include <cmath>
3  #include <iomanip>
4  #include <fstream>
5  #include <cstdlib>
6  #include <string>
7  using namespace std;
8 
9  class EfieldT{
10 public:
11 
12   EfieldT( double Tvc, double Tvp, double Tlamdac, double TI3, double TI2, double Teta, ↩
         double Ttau2, double Ttau3, double Ttau4, double Tomega0, int TNa, int TcurrentDecay =↩
          0 );
13 
14   double EtDifferential( double z, double t, double theta, double rObs );
15 
16   double integrateZUpper( double t, double theta, double rObs, int N );
17   double integrateZLower( double t, double theta, double rObs, int N );
18   double integrateZ( double t, double theta, double rObs, int N );
19 
20 
21 private:
22   double vchan;
23   double vp;
24   double lamdac;
25   double I3;
26   double I2;
27   double eta;
28   double tau2;
29   double tau3;
30   double tau4;
31   double omega0;
32   int Na;
33   int factorial[13];
34   int currentDecay;
35 
36   const double pi = 4.0 * atan( 1.0 );      // Fixed Constants:
37   const double c = 299792458;
38   const double mu = 4 * pi * pow( 10.0, -7.0 );
39   const double epsilon = 8.8541878176 * pow( 10.0, -12.0 );
40 };
```

Listing C.5: EfieldT.cpp

```
#include "EfieldT.h"

// Note that there is an emphasis on memory management in this code. It may seem bulky, but
// every instance of the object is self contained in order to facilitate threading.

// This is the constructor class for the EfieldT object. The Electric (E) field in the
// Theta co-ordinate. WRT Cylindrical co-ordinates. This constructor takes in all the
// variables needed to later calculate the E field at instants in Time.
EfieldT::EfieldT( double Tvchan, double Tvp, double Tlamdac, double TI3, double TI2, double Teta, double Ttau2, double Ttau3, double Ttau4, double Tomega0, int TNa, int TcurrentDecay ){

  factorial[0] = 1;             // Fixed Constants:
  factorial[1] = 1;
  factorial[2] = 2;
  factorial[3] = 6;
  factorial[4] = 24;
  factorial[5] = 120;
  factorial[6] = 720;
  factorial[7] = 5040;
  factorial[8] = 40320;
  factorial[9] = 362880;
  factorial[10] = 3628800;
  factorial[11] = 39916800;
  factorial[12] = 479001600;

  I3 = TI3;                     // Terespolsky Function Constants:
  I2 = TI2;
  eta = Teta;
  tau2 = Ttau2;
  tau3 = Ttau3;
  tau4 = Ttau4;
  omega0 = Tomega0;
  Na = TNa;

  vchan = Tvchan;               // RS constants:
  vp = Tvp;
  lamdac = Tlamdac;
  currentDecay = TcurrentDecay;
}

// This is the main function of the class. It calculates what I like to call the "Field
// Strength". This is not the correct terminology but it works for me. What this actually
// is, is the electric field differential, at a point in space and time, due to an element at
// height Z on the lightning channel. In simple terms it is the function that returns the
// value of the term inside the integral component of the equation. Later functions will use
// this to calculate integral value. refer to the chapter on Finite Antenna Method for the
// mathematics and equations. The parameters it takes are the observation point Time (t),
// observation point angle (theta) and observation point radial distance (rObs). It also
// takes the height component value of the channel (z). The returned value is used in the
// integration equation/function.
double EfieldT::EtDifferential(double z,double t,double theta,double rObs){

    double singleSumTerm = 0;
    double doubleSumTermUpperIntegral = 0;
    double doubleSumTermLowerIntegral = 0;

    // The static term has an integral between two time points. The integral is performed
    // between tb -> t. "t" is the time point being evaluated, and "tb" is the effective zero
    // starting point. "tb" is defined so that the time shift term is zero. This ultimately
    // ties back to the unit step function in the current impulse/RS model and is necessary
    // so that no negative time points are used in the models. It is also the lower point
    // because before this time there would not have been any current components So the upper
    // limit in the static term integral is already catered for in the equations as
    // "timeShiftTerm". Therefore the lower limit is just zero after the time shift.
    double timeShiftTerm = t - abs( z ) / vchan - sqrt( pow( rObs, 2.0 ) + pow( z, 2.0 ) - 2.0 * rObs * z * cos( theta ) ) / c;
```

```
65      double lowerIntegralTime =  0;
66      double Rterm = sqrt( pow( rObs, 2.0) + pow( z, 2.0 ) - 2.0 * rObs * z * cos( theta ) );
67
68      if(timeShiftTerm < 0){ return 0; }
69      else{
70
71        for( int j=0; j <= Na; j++ ){
72          singleSumTerm += pow( omega0 * ( timeShiftTerm), j ) / factorial[ j ];
73        }
74
75        for( int i=0; i <= Na; i++){
76          for( int j=0; j <= i; j++){
77          doubleSumTermUpperIntegral += (pow(omega0,i)*pow(tau2,j)*pow(timeShiftTerm,i-j))/(↩
                factorial[i-j]*pow((tau2*omega0+1),j+1));
78
79          doubleSumTermLowerIntegral += (pow(omega0,i)*pow(tau2,j)*pow(lowerIntegralTime,i-j))↩
                /(factorial[i-j]*pow((tau2*omega0+1),j+1));
80          }
81        }
82
83        // E field Static theta element.
84        double EStElement = (exp((-1.0*abs(z)*currentDecay)/lamdac)/(4.0*pi*epsilon))*( ((sin(↩
              theta))/(pow(Rterm,3))) + (((-3.0*z*sin(theta))*(z-rObs*cos(theta)))/(pow(Rterm,5)↩
              )) );
85
86        double EtStatic     = EStElement * ((    (I3/eta * tau2 * exp(-1.0*timeShiftTerm/tau2 ↩
              -1.0*timeShiftTerm*omega0)) * (-1.0*exp(+1.0*omega0*timeShiftTerm) + ↩
              doubleSumTermUpperIntegral) + I2 * (-1.0*tau3*exp(-1.0*timeShiftTerm/tau3) + tau4*↩
              exp(-1.0*timeShiftTerm/tau4))    ) - (    (I3/eta * tau2 * exp(-1.0*↩
              lowerIntegralTime/tau2 -1.0*lowerIntegralTime*omega0)) * (-1.0*exp(+1.0*omega0*↩
              lowerIntegralTime) + doubleSumTermLowerIntegral) + I2 * (-1.0*tau3*exp(-1.0*↩
              lowerIntegralTime/tau3) + tau4*exp(-1.0*lowerIntegralTime/tau4))    ));
87
88        double EtInduction  = (exp((-1.0*abs(z)*currentDecay)/lamdac)/(4.0*pi*epsilon*pow(c↩
              ,1.0)))*(sin(theta)/(pow(Rterm,2)) - 3*(z - rObs*cos(theta))*z*sin(theta)/(pow(↩
              Rterm,4)))*((I3/eta * exp(-1.0*timeShiftTerm/tau2)) * ( 1.0 - exp( -1.0*omega0*↩
              timeShiftTerm) * singleSumTerm) + I2*( exp(-1.0*timeShiftTerm/tau3) - exp(-1.0*↩
              timeShiftTerm/tau4)));
89
90        double EtRadiation  = (exp((-1.0*abs(z)*currentDecay)/lamdac)/(4.0*pi*epsilon*pow(c↩
              ,2.0)))*(sin(theta)/(pow(Rterm,1)) - 1*(z - rObs*cos(theta))*z*sin(theta)/(pow(↩
              Rterm,3)))*((I3/eta * exp(-1.0*timeShiftTerm/tau2)) * ( exp(-1.0*omega0*↩
              timeShiftTerm)*pow(omega0,Na + 1)*pow(timeShiftTerm, Na) / factorial[Na] - 1.0/↩
              tau2 * ( 1.0 -1.0*exp( -1.0*omega0*timeShiftTerm)*singleSumTerm )) + I2*( -1.0/↩
              tau3 * exp(-1.0*timeShiftTerm/tau3) + 1.0/tau4 * exp(-1.0*timeShiftTerm/tau4)));
91
92        double EfieldT = EtStatic + EtRadiation + EtInduction;
93
94        return EfieldT;
95      }
96  }
97
98  // This function integrates the ErField strength of the positive lightning channel.
99  double EfieldT::integrateZUpper( double t, double theta, double rObs, int N ){
100
101   double upperZ = (rObs / (1.0 - pow( vchan / c, 2.0 ) ))*( -1.0 * pow( vchan / c, 2.0 ) * ↩
          cos( theta ) + ( vchan * t ) / rObs - vchan / c * sqrt( 1.0 - pow( vchan / c , 2.0 ) +↩
           pow( vchan * t / rObs, 2.0 ) + pow( vchan * cos( theta ) / c , 2.0 ) - 2.0 * vchan * ↩
          t * cos( theta ) / rObs ) );
102
103   // Trapezoidal Integration
104   double result = EtDifferential( 0.0, t, theta, rObs) + EtDifferential( upperZ, t, theta, ↩
          rObs );
105   double dz = ( upperZ ) / ( ( double ) N );
106
107   for ( double z = dz; z < upperZ; z+=dz ){
108     result += 2 * EtDifferential( z, t, theta, rObs);
109   }
```

```
110   result *= upperZ / ( 2.0 * ( double ) N );
111
112   return result;
113 }
114
115 // This integrates the ErField strength of the lightning channel image (below ground).
116 double EfieldT::integrateZLower( double t, double theta, double rObs, int N ){
117
118   double lowerZ = (rObs / ( 1.0 - pow( vchan / c, 2.0 ) ))*( -1.0 * pow( vchan / c, 2.0 ) * ↩
          cos( pi - theta ) + ( vchan * t ) / rObs - vchan / c * sqrt( 1.0 - pow( vchan / c , ↩
          2.0 ) + pow( vchan * t / rObs, 2.0 ) + pow( vchan * cos( pi - theta ) / c , 2.0 ) - ↩
          2.0 * vchan * t * cos( pi - theta ) / rObs ) );
119
120   // Trapezoidal Integration
121   double result = EtDifferential( 0.0, t, theta, rObs) + EtDifferential( -1.0*lowerZ, t, ↩
          theta, rObs );
122   double dz = ( lowerZ ) / ( ( double ) N );
123
124   for ( double z = dz; z < lowerZ; z+=dz ){
125     result += 2 * EtDifferential( -1.0*z, t, theta, rObs);
126   }
127   result *= lowerZ / ( 2.0 * ( double ) N );
128
129   return result;
130 }
131
132 // This function performs an integral over the positive and negative lightning
133 // channels. It performs this integral at a specific time instant over the
134 // entire applicable lightning channel. The applicable lightning channel length
135 // is dependant on the time taken for current to travel in the channel.
136 // The returned result is the field value at the time instant.
137 // The input parameters are the time instant of interest (t), observation point
138 // angle (theta), observation point radial distance (rObs), and N is the
139 // amount of integration steps in each integral stage, regardless of channel
140 // length.
141 double EfieldT::integrateZ( double t, double theta, double rObs, int N ){
142   return integrateZUpper(t,theta,rObs,N) + integrateZLower(t,theta,rObs,N);
143 }
```

Listing C.6: 2D_FINANT_Settings.csv

```
1  tau1,0.072,us,
2  tau2,5,us,
3  tau3,100,us,
4  tau4,6,us,
5  lamdac,2000,nucciconst,
6  currentDecay,0,trueorfalse,
7  I1,9900,A,
8  I2,7500,A,
9  I3,9550,A,
10 eta,0.845,number,
11 Na,3,teresnumber,
12 vp,299792458,m/s,
13 vchan,150000000,m/s,
14 rObs,50,m,
15 observationAngleDeg,90,deg,
16 integrationSteps,1000,int,
17 deltaT,5,ns,
18 simRunTime,5167,ns,
```

Appendix D

3D Cartesian FDTD Code

Listing D.1: main.cpp

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;

#include "FDTDCart3D.h"

int main(int argc, char const *argv[]){
  FDTDCart3D lightningStroke("3D_Settings_File.csv", argc, argv);
  lightningStroke.PrintMinimalSimParameters();
  // lightningStroke.fourierSource();
  lightningStroke.EvaluateFDTDPlane();
}
```

Listing D.2: FDTDCart3D.h

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;

class FDTDCart3D{
public:
  FDTDCart3D(string settingsFile, int argc, char const *argv[]);
  void EvaluateFDTDPlane();
  void setSimulationParameters(string settingsFile);
  void PrintMinimalSimParameters();
  double NucciTeresCurrent(double z, double t);

  // ****************************************************************
  // ******************** Boundary Related **************************
  void Mur2nRBC();
  void Xplanebound_Mur2nRBC();
  void Yplanebound_Mur2nRBC();
  void Zplanebound_Mur2nRBC();

```

```
24    void Xupright_edges();
25    void Yupright_edges();
26    void Zupright_edges();
27
28  private:
29    // These are variables provided in the Settings file.
30    float vp;
31    float vchan;
32    float f0;
33    float gridSizeX, gridSizeY, gridSizeZ;
34    float deltaX, deltaY, deltaZ;
35    float deltaT;
36    int useDTfromfile;
37    float simRunTime;
38    double tau1, tau2, tau3, tau4;
39    double lamdac;
40    int currentDecay;
41    double I1, I2, I3;
42    int Na;
43    double eta, omega0;
44    float rObs,hObs;
45
46    // These are the normal class variables. They need to be assigned in the constructor.
47    int currentTime;
48    int timeMinus1;
49    int timeMinus2;
50    int tempAddress;
51
52    float idealGridSize;
53    float optimalDeltaT;
54
55    float ExFSC, EyFSC, EzFSC;
56    float HxFSC, HyFSC, HzFSC;
57
58    // The "steps" define the number of steps in the respective directions. The origin is in
59    // the middle of the plane with gridsize variable length in every direction of the origin.
60    int isteps, jsteps, ksteps;
61    int nSteps;
62    int iorigin, jorigin, korigin;
63    int XOBS, ZOBS;
64
65    double*** EfieldX;
66    double*** EfieldY;
67    double*** EfieldZ;
68    double*** HfieldX;
69    double*** HfieldY;
70    double*** HfieldZ;
71
72    // These are the upper and lower boundries of the evaluation plane. This is the right and
73    // left hand walls respectively. In the lower boundry the outer column is 0, and the inner
74    // is 1. In the Upper boundry the outer column is 1, and the inner is 0.
75
76    // X- Bound
77    double*** EzXLowerBndrTmin1;
78    double*** EzXLowerBndrTmin2;
79    double*** EzXUpperBndrTmin1;
80    double*** EzXUpperBndrTmin2;
81
82    double*** EyXLowerBndrTmin1;
83    double*** EyXLowerBndrTmin2;
84    double*** EyXUpperBndrTmin1;
85    double*** EyXUpperBndrTmin2;
86
87    // Y- Bound
88    double*** EzYLowerBndrTmin1;
89    double*** EzYLowerBndrTmin2;
90    double*** EzYUpperBndrTmin1;
91    double*** EzYUpperBndrTmin2;
```

```
92
93    double*** ExYLowerBndrTmin1;
94    double*** ExYLowerBndrTmin2;
95    double*** ExYUpperBndrTmin1;
96    double*** ExYUpperBndrTmin2;
97
98    // Z- Bound
99    double*** EyZLowerBndrTmin1;
100   double*** EyZLowerBndrTmin2;
101   double*** EyZUpperBndrTmin1;
102   double*** EyZUpperBndrTmin2;
103
104   double*** ExZLowerBndrTmin1;
105   double*** ExZLowerBndrTmin2;
106   double*** ExZUpperBndrTmin1;
107   double*** ExZUpperBndrTmin2;
108
109   // These are the precalculated variables used to save computational time.
110   const double pi = ( 4.0 * atan( 1.0 ) );
111   const double c  = 299792458;
112   const double mu = 4 * pi * pow( 10.0, -7.0 );
113   const double epsilon = 8.8541878176 * pow( 10.0, -12.0 );
114   float theta;
115   float timeToReachObservationPoint;
116   int percentProgress;
117   int factorial[13];
118
119   char partATestID[128];
120   char partBTestID[128];
121   char ExFieldcurvename[128];
122   char EyFieldcurvename[128];
123   char EzFieldcurvename[128];
124   char ErFieldcurvename[128];
125   char EtFieldcurvename[128];
126
127 };
```

Listing D.3: FDTDCart3D.cpp

```
1  #include <iostream>
2  #include <cmath>
3  #include <iomanip>
4  #include <fstream>
5  #include <cstdlib>
6  #include <string>
7  #include <algorithm>
8
9  #include "FDTDCart3D.h"
10
11 using namespace std;
12
13 // This is a macro for the 3D array instantiation. This is a way to create a contiguous array
14 // of memory. Standard methods don't work due to array sizes, and heap memory size. The last
15 // index is K. IE, this is the index that should be used the most. From my understanding when
16 // memory is called for use the compiler will load a batch of the surrounding memory points.
17 // Therefore you will see in later code that the K index is always in the inner most loop
18 // (called most often). This macro also zero's the array.
19 #define create3Darray(fieldname, isteps, jsteps, ksteps)        \
20     double*** fieldname = (double***) new double**[isteps];     \
21     for(int i = 0; i < isteps; i++) {                           \
22       fieldname[i]=(double**)new double*[jsteps];               \
23       for (int j=0; j<jsteps; j++) {                            \
24         fieldname[i][j]=(double*)new double[ksteps];            \
25       }                                                         \
26     }                                                           \
27     for(int i=0; i != isteps ; i++){                            \
```

```
28      for(int j=0; j != jsteps ; j++){                          \
29        for(int k=0; k != ksteps ; k++)                         \
30          fieldname[i][j][k] =0;                                \
31      }                                                         \
32    }
33
34 // This is a macro for the 1D array instantiation.
35 #define create1Darray(fieldname, steps)                        \
36    double* fieldname = (double*) new double[steps];           \
37    for(int i=0; i != steps ; i++){                            \
38      fieldname[i] = 0;                                        \
39    }
40
41 // Constructor. It needs a settings file and command line arguments to be passed to it.
42 FDTDCart3D::FDTDCart3D(string settingsFile, int argc, char const *argv[]){
43
44    setSimulationParameters(settingsFile);
45
46    // Character strings for file naming.
47    sprintf(partATestID,"D%05gH%04g", rObs*100, hObs*100);
48    sprintf(partBTestID,"Size%04g_%04g_%04g_D%03g_%03g_%03g", gridSizeX, gridSizeY, gridSizeZ↩
         , deltaX*10,   deltaY*10, deltaZ*10);
49
50    // Creating Output File names for the different fields. These files are
51    // opened, written to and closed in the main evaluateFDTD function. Also
52    // Note that this is hard coded to save files to the folder: "Simulations".
53    sprintf(ExFieldcurvename, "./Simulations/%sExField%s.curve", partATestID, partBTestID);
54    sprintf(EyFieldcurvename, "./Simulations/%sEyField%s.curve", partATestID, partBTestID);
55    sprintf(EzFieldcurvename, "./Simulations/%sEzField%s.curve", partATestID, partBTestID);
56    sprintf(ErFieldcurvename, "./Simulations/%sErField%s.curve", partATestID, partBTestID);
57    sprintf(EtFieldcurvename, "./Simulations/%sEtField%s.curve", partATestID, partBTestID);
58
59    // These are the factorial numbers for Nucci-Teres current Function.
60    factorial[0] = 1;
61    factorial[1] = 1;
62    factorial[2] = 2;
63    factorial[3] = 6;
64    factorial[4] = 24;
65    factorial[5] = 120;
66    factorial[6] = 720;
67    factorial[7] = 5040;
68    factorial[8] = 40320;
69    factorial[9] = 362880;
70    factorial[10] = 3628800;
71    factorial[11] = 39916800;
72    factorial[12] = 479001600;
73
74    // Calculation of DeltaT. This assumes a CFL condition of 1.
75    optimalDeltaT = 1/(vp*sqrt(pow(deltaX,-2) + pow(deltaY,-2) + pow(deltaZ,-2) ));
76
77    // This is the "Thumbsuck Rule" for Terespolski Function.
78    omega0 = ( double )Na / tau1;
79
80    // To use the DT from the file, set the variable to 1, else the DT will be optimal.
81    if(useDTfromfile != 1){
82      deltaT = optimalDeltaT;
83    }
84
85    // These are the free space constants (FSC) used in my simplified example.
86    // Under complex environments with different objects, these constants would
87    // consist of 3D matrices that describe the different materials in space.
88    ExFSC = deltaT / epsilon;
89    EyFSC = deltaT / epsilon;
90    EzFSC = deltaT / epsilon;
91    HxFSC = deltaT / mu;
92    HyFSC = deltaT / mu;
93    HzFSC = deltaT / mu;
94
```

```
95      // These are the node steps in the 3 co-ordinate system.
96      isteps = int((gridSizeX*2)/deltaX);
97      jsteps = int((gridSizeY*2)/deltaY);
98      ksteps = int((gridSizeZ)/deltaZ); // I dont want the negative plane.
99      // Center point of simulation space.
100     iorigin = int(isteps/2);
101     jorigin = int(jsteps/2);
102     korigin = int(ksteps/2);
103
104     // Observation point distance to node conversion.
105     XOBS = rObs/deltaX;
106     ZOBS = hObs/deltaZ;
107
108     // Tim Steps
109     nSteps = int(simRunTime/deltaT);
110
111     // Time to source:
112     timeToReachObservationPoint = (sqrt(pow(rObs,2) + pow(hObs,2)))/vp;
113
114     // Co-ordinate conversions.
115     // This is the angle between the Z axis and observation point.
116     theta = atan(rObs/hObs);
117
118     // EM Fields
119     // Create the 3D array pointers here.
120     create3Darray(tempEfieldX, isteps-1, jsteps, ksteps);
121     create3Darray(tempEfieldY, isteps, jsteps-1, ksteps);
122     create3Darray(tempEfieldZ, isteps, jsteps, ksteps-1);
123     create3Darray(tempHfieldX, isteps, jsteps-1, ksteps-1);
124     create3Darray(tempHfieldY, isteps-1, jsteps, ksteps-1);
125     create3Darray(tempHfieldZ, isteps-1, jsteps-1, ksteps);
126     // Assign the 3D array pointers here.
127     EfieldX = tempEfieldX;
128     EfieldY = tempEfieldY;
129     EfieldZ = tempEfieldZ;
130     HfieldX = tempHfieldX;
131     HfieldY = tempHfieldY;
132     HfieldZ = tempHfieldZ;
133
134     // X- Bound
135     // Create boundary matrix pointers.
136     create3Darray(XtempBoundry1, 2, jsteps, ksteps-1);
137     create3Darray(XtempBoundry2, 2, jsteps, ksteps-1);
138     create3Darray(XtempBoundry3, 2, jsteps, ksteps-1);
139     create3Darray(XtempBoundry4, 2, jsteps, ksteps-1);
140     create3Darray(XtempBoundry5, 2, jsteps-1, ksteps);
141     create3Darray(XtempBoundry6, 2, jsteps-1, ksteps);
142     create3Darray(XtempBoundry7, 2, jsteps-1, ksteps);
143     create3Darray(XtempBoundry8, 2, jsteps-1, ksteps);
144     // Assign the boundary pointers here.
145     EzXLowerBndrTmin1 = XtempBoundry1;
146     EzXLowerBndrTmin2 = XtempBoundry2;
147     EzXUpperBndrTmin1 = XtempBoundry3;
148     EzXUpperBndrTmin2 = XtempBoundry4;
149
150     EyXLowerBndrTmin1 = XtempBoundry5;
151     EyXLowerBndrTmin2 = XtempBoundry6;
152     EyXUpperBndrTmin1 = XtempBoundry7;
153     EyXUpperBndrTmin2 = XtempBoundry8;
154
155     // Y- Bound
156     // Create boundary matrix pointers.
157     create3Darray(YtempBoundry1, isteps, 2, ksteps-1);
158     create3Darray(YtempBoundry2, isteps, 2, ksteps-1);
159     create3Darray(YtempBoundry3, isteps, 2, ksteps-1);
160     create3Darray(YtempBoundry4, isteps, 2, ksteps-1);
161     create3Darray(YtempBoundry5, isteps-1, 2, ksteps);
162     create3Darray(YtempBoundry6, isteps-1, 2, ksteps);
```

```
163     create3Darray(YtempBoundry7, isteps-1, 2, ksteps);
164     create3Darray(YtempBoundry8, isteps-1, 2, ksteps);
165     // Assign the boundary pointers here.
166     EzYLowerBndrTmin1 = YtempBoundry1;
167     EzYLowerBndrTmin2 = YtempBoundry2;
168     EzYUpperBndrTmin1 = YtempBoundry3;
169     EzYUpperBndrTmin2 = YtempBoundry4;
170 
171     ExYLowerBndrTmin1 = YtempBoundry5;
172     ExYLowerBndrTmin2 = YtempBoundry6;
173     ExYUpperBndrTmin1 = YtempBoundry7;
174     ExYUpperBndrTmin2 = YtempBoundry8;
175 
176     // Z- Bound
177     // Create boundary matrix pointers.
178     create3Darray(ZtempBoundry1, isteps, jsteps-1, 2);
179     create3Darray(ZtempBoundry2, isteps, jsteps-1, 2);
180     create3Darray(ZtempBoundry3, isteps, jsteps-1, 2);
181     create3Darray(ZtempBoundry4, isteps, jsteps-1, 2);
182     create3Darray(ZtempBoundry5, isteps-1, jsteps, 2);
183     create3Darray(ZtempBoundry6, isteps-1, jsteps, 2);
184     create3Darray(ZtempBoundry7, isteps-1, jsteps, 2);
185     create3Darray(ZtempBoundry8, isteps-1, jsteps, 2);
186     // Assign the boundary pointers here.
187     EyZLowerBndrTmin1 = ZtempBoundry1;
188     EyZLowerBndrTmin2 = ZtempBoundry2;
189     EyZUpperBndrTmin1 = ZtempBoundry3;
190     EyZUpperBndrTmin2 = ZtempBoundry4;
191 
192     ExZLowerBndrTmin1 = ZtempBoundry5;
193     ExZLowerBndrTmin2 = ZtempBoundry6;
194     ExZUpperBndrTmin1 = ZtempBoundry7;
195     ExZUpperBndrTmin2 = ZtempBoundry8;
196 }
197 
198 // This function is passed a file name string and extracts the required simulation variables
199 // from the file and assigns them to the relevant variables of the class. Care must be taken
200 // to ensure that the settings file is formatted correctly. Remember that some of the
201 // variables may be changed from file values if command line arguments are given.
202 void FDTDCart3D::setSimulationParameters(string settingsFile){
203 
204     ifstream inputFromFile(settingsFile, ios::in);
205 
206     if(!inputFromFile){
207       cerr << "File could not be opened!" << endl;
208       exit(1);
209     }
210 
211     string UnitName, Value, Units;
212 
213     while(getline(inputFromFile, UnitName, ',')){
214 
215       // This next line of code removes the newline character from my settings file when it
216       // is read into the variables. This was only an issue with the UnitName variable.
217       UnitName.erase(std::remove(UnitName.begin(), UnitName.end(), '\n'), UnitName.end());
218       getline(inputFromFile, Value, ',');
219       getline(inputFromFile, Units, ',');// This variable does not get used.
220 
221       if(UnitName == "vp"){vp = std::stof( Value );}
222       else if(UnitName == "vchan"){vchan = std::stof( Value );}
223       else if(UnitName == "f0"){f0 = std::stof( Value );}
224       else if(UnitName == "gridSizeX"){gridSizeX = std::stof( Value );}
225       else if(UnitName == "gridSizeY"){gridSizeY = std::stof( Value );}
226       else if(UnitName == "gridSizeZ"){gridSizeZ = std::stof( Value );}
227       else if(UnitName == "deltaX"){deltaX = std::stof( Value );}
228       else if(UnitName == "deltaY"){deltaY = std::stof( Value );}
229       else if(UnitName == "deltaZ"){deltaZ = std::stof( Value );}
230       else if(UnitName == "deltaT"){deltaT = std::stof( Value ) * pow(10,-9);}
```

```
231       else if(UnitName == "useDTfromfile"){useDTfromfile = std::stoi(Value);}
232       else if(UnitName == "simRunTime"){simRunTime = std::stof( Value) * pow(10,-9);}
233       else if(UnitName == "tau1"){tau1 = std::stof( Value) * pow(10,-6);}
234       else if(UnitName == "tau2"){tau2 = std::stof( Value) * pow(10,-6);}
235       else if(UnitName == "tau3"){tau3 = std::stof( Value) * pow(10,-6);}
236       else if(UnitName == "tau4"){tau4 = std::stof( Value) * pow(10,-6);}
237       else if(UnitName == "lamdac"){lamdac = std::stof( Value);}
238       else if(UnitName == "currentDecay"){currentDecay = std::stof( Value);}
239       else if(UnitName == "I1"){I1 = std::stof( Value);}
240       else if(UnitName == "I2"){I2 = std::stof( Value);}
241       else if(UnitName == "I3"){I3 = std::stof( Value);}
242       else if(UnitName == "eta"){eta = std::stof( Value);}
243       else if(UnitName == "Na"){Na = std::stof( Value);}
244       else if(UnitName == "rObs"){rObs = std::stof( Value);}
245       else if(UnitName == "hObs"){hObs = std::stof( Value);}
246       else{ cout << "Variable "<< UnitName << " : " << Value <<" was not assigned." << endl;}
247     }
248 }
249
250 // This function prints the minimal Sim variables for batch script simulations.
251 void FDTDCart3D::PrintMinimalSimParameters(){
252     cout << "***********************************************" << endl;
253     cout <<"Simulation Variables:" <<endl;
254     cout <<"Requested rObs: " <<rObs <<". Actual rObs: " <<XOBS*deltaX << endl;
255     cout <<"Requested hObs: " <<hObs <<". Actual hObs: " <<ZOBS*deltaZ << endl;
256     cout <<"SizeX -SizeY -SizeZ -dX -dY -dZ -simTime" <<endl;
257     cout <<gridSizeX << "    " << gridSizeY << "    " <<  gridSizeZ << "    " <<  deltaX << "↩
            " << deltaY << "  " << deltaZ << "  " << simRunTime << endl;
258 }
259
260 // This is the main workhorse of the Class. It will itterate through the matrix variables of
261 // the FDTD plane and calculate the EM fields at every cell at every time. I will eventually
262 // add smaller functions to handle the different aspects but for now this will be the main
263 // workhorse of the program. Has no inputs, but uses the private variables in the header.
264 void FDTDCart3D::EvaluateFDTDPlane(){
265
266     // Open the output data field files:
267       ofstream observationPointExField(ExFieldcurvename);
268       ofstream observationPointEyField(EyFieldcurvename);
269       ofstream observationPointEzField(EzFieldcurvename);
270       ofstream observationPointErField(ErFieldcurvename);
271       ofstream observationPointEtField(EtFieldcurvename);
272     // Write the output file headers:
273       observationPointExField << "#EfieldX" << endl;
274       observationPointEyField << "#EfieldY" << endl;
275       observationPointEzField << "#EfieldZ" << endl;
276       observationPointErField << "#EfieldR" << endl;
277       observationPointEtField << "#EfieldTheta" << endl;
278
279     cout << "***********************************************" << endl;
280     cout << "Starting Simulation Time Steps: " << endl;
281
282     // This is where the time stepping starts.
283     for(int n=0; n!=nSteps; n++){
284
285       cout <<"Progress " << ((n*deltaT)/simRunTime)*100.00 << "\%  \r" ;
286       std::cout.flush();
287
288       // Magnetic Field Update Equation
289         for(int i=0; i != isteps; i++){
290           for(int j=0; j != jsteps-1; j++){
291             for(int k=0; k != ksteps-1; k++){
292               HfieldX[i][j][k] = HfieldX[i][j][k] + HxFSC * ((EfieldY[i][j][k+1] - EfieldY[i↩
                      ][j][k])/deltaZ - (EfieldZ[i][j+1][k] - EfieldZ[i][j][k])/deltaY);
293             }
294           }
295         }
296         for(int i=0; i != isteps-1; i++){
```

```
297           for(int j=0; j != jsteps; j++){
298             for(int k=0; k != ksteps-1; k++){
299               HfieldY[i][j][k] = HfieldY[i][j][k] + HyFSC * ((EfieldZ[i+1][j][k] - EfieldZ[i↩
                      ][j][k])/deltaX - (EfieldX[i][j][k+1] - EfieldX[i][j][k])/deltaZ);
300             }
301           }
302         }
303         for(int i=0; i != isteps-1; i++){
304           for(int j=0; j != jsteps-1; j++){
305             for(int k=0; k != ksteps; k++){
306               HfieldZ[i][j][k] = HfieldZ[i][j][k] + HzFSC * ((EfieldX[i][j+1][k] - EfieldX[i↩
                      ][j][k])/deltaY - (EfieldY[i+1][j][k] - EfieldY[i][j][k])/deltaX);
307             }
308           }
309         }
310
311       // Electric Field Update Equation
312         for(int i=0; i != isteps-1; i++){
313           for(int j=1; j != jsteps-1; j++){
314             for(int k=1; k != ksteps-1; k++){
315               EfieldX[i][j][k] = EfieldX[i][j][k] + ExFSC * ((HfieldZ[i][j][k] - HfieldZ[i][j↩
                      -1][k])/deltaY - (HfieldY[i][j][k] - HfieldY[i][j][k-1])/deltaZ);
316             }
317           }
318         }
319         for(int i=1; i != isteps-1; i++){
320           for(int j=0; j != jsteps-1; j++){
321             for(int k=1; k != ksteps-1; k++){
322               EfieldY[i][j][k] = EfieldY[i][j][k] + EyFSC * ((HfieldX[i][j][k] - HfieldX[i][j↩
                      ][k-1])/deltaZ - (HfieldZ[i][j][k] - HfieldZ[i-1][j][k])/deltaX);
323             }
324           }
325         }
326         for(int i=1; i != isteps-1; i++){
327           for(int j=1; j != jsteps-1; j++){
328             for(int k=0; k != ksteps-1; k++){
329               EfieldZ[i][j][k] = EfieldZ[i][j][k] + EzFSC * ((HfieldY[i][j][k] - HfieldY[i↩
                      -1][j][k])/deltaX - (HfieldX[i][j][k] - HfieldX[i][j-1][k])/deltaY);
330             }
331           }
332         }
333
334       // Source
335         // Add the phased current elements along the lightning channel.
336         // x=0, y=0, z=0->Zmax
337         for(int k=0; k!=ksteps; k++){
338           // Note the negative sign below. This is to account for negative current.
339           EfieldZ[iorigin][jorigin][k] = EfieldZ[iorigin][jorigin][k] - (deltaT/(epsilon*↩
                deltaX*deltaY))*NucciTeresCurrent((k+0.5)*deltaZ, n*deltaT);
340         }
341         // EfieldZ[iorigin][jorigin][korigin] = 1000*sin(SINEwaveConstant*n);
342
343       // Boundary Condition
344         Mur2nRBC();
345
346       // Write field data to output files.
347         // The "if" check ensures that data is only written from the point when the EM field
348         // first reaches the observation point. This step also does the co-ordinate
349         // conversion for Cartesian to Spherical field components. Note that all these fields
350         // are measured on the Y=0 axis. This is hard coded for my work, but can be changed
351         // here for convenience. I chose this because it simplifies co-ordinate conversion
352         // for the third direction in spherical. In this case,Ey=Ephi
353         if(n*deltaT >= timeToReachObservationPoint){
354           observationPointExField<<(n*deltaT-timeToReachObservationPoint)*pow(10,+9)<<" "<<↩
                EfieldX[iorigin+XOBS][jorigin][ZOBS]<<endl;
355           observationPointEyField<<(n*deltaT-timeToReachObservationPoint)*pow(10,+9)<<" "<<↩
                EfieldY[iorigin+XOBS][jorigin][ZOBS] << endl;
```

```
356           observationPointEzField<<(n*deltaT-timeToReachObservationPoint)*pow(10,+9)<<" "<<↩
                  EfieldZ[iorigin+XOBS][jorigin][ZOBS] << endl;
357           observationPointErField<<(n*deltaT-timeToReachObservationPoint)*pow(10,+9)<<" "<< ↩
                  +1*(cos(theta)*EfieldZ[iorigin+XOBS][jorigin][ZOBS] + sin(theta)*EfieldX[↩
                  iorigin+XOBS][jorigin][ZOBS]) << endl;
358           observationPointEtField<<(n*deltaT-timeToReachObservationPoint)*pow(10,+9)<<" "<< ↩
                  +1*(-1*sin(theta)*EfieldZ[iorigin+XOBS][jorigin][ZOBS] + cos(theta)*EfieldX[↩
                  iorigin+XOBS][jorigin][ZOBS]) << endl;
359         }
360     }
361
362     cout << endl;
363     cout << "Simulation Complete." << endl;
364     cout << "*********************************************" << endl;
365     cout << endl;
366
367     // Close the data files.
368       observationPointExField.close();
369       observationPointEyField.close();
370       observationPointEzField.close();
371       observationPointErField.close();
372       observationPointEtField.close();
373 }
374
375 // This is a function that returns the current along a lightning channel.
376 // It takes in the time of interest and location on the channel.
377 double FDTDCart3D::NucciTeresCurrent(double z, double t){
378     // Max Current Is: 10975.6 at 280 ns
379     // 10% of Max Current Is: 1097.56 at 192182 ns
380     if(t - abs( z ) / vchan  < 0){ return 0; }
381     else{
382       double sum = 0;
383       for ( int i = 0; i <= Na; i++ ){
384         sum += pow( omega0 * ( t - abs( z ) / vchan  ), i ) / factorial[ i ];
385       }
386
387       double current = (exp((-1.0*abs(z)*currentDecay)/lamdac)) * ( ( I3 / eta * exp( -1.0 * ↩
              ( t - abs( z ) / vchan  ) / tau2 ) ) * ( 1.0 - exp( -1.0 * omega0 * ( t - abs( z )↩
               / vchan  ) ) * sum ) + I2 * ( exp( -1.0 * ( t - abs( z ) / vchan  ) / tau3 ) - ↩
              exp( -1.0 * ( t - abs( z ) / vchan  ) / tau4 ) ) );
388
389       return current;
390     }
391 }
392
393 // This function does the 2nd Order Mur bondary condition for the 3D space. It does not take
394 // inputs and accesses the relevant fields directly. This may be a bad design, but it seems
395 // pointless to input all 6 fields into this function.
396 void FDTDCart3D::Mur2nRBC(){
397
398     Xplanebound_Mur2nRBC();
399     Yplanebound_Mur2nRBC();
400     Zplanebound_Mur2nRBC();
401
402     Xupright_edges();
403     Yupright_edges();
404     Zupright_edges();
405
406     // corners:
407     EfieldX[0][0][0]                = EfieldX[0+1][0+1][0+1];
408     EfieldX[isteps-2][0][0]         = EfieldX[isteps-3][0+1][0+1];
409     EfieldX[isteps-2][jsteps-1][0]  = EfieldX[isteps-3][jsteps-2][0+1];
410     EfieldX[0][jsteps-1][0]         = EfieldX[0+1][jsteps-2][0+1];
411     EfieldX[0][0][ksteps-1]                 = EfieldX[0+1][0+1][ksteps-2];
412     EfieldX[isteps-2][0][ksteps-1]          = EfieldX[isteps-3][0+1][ksteps-2];
413     EfieldX[isteps-2][jsteps-1][ksteps-1]   = EfieldX[isteps-3][jsteps-2][ksteps-2];
414     EfieldX[0][jsteps-1][ksteps-1]          = EfieldX[0+1][jsteps-2][ksteps-2];
415
```

```
416     EfieldY[0][0][0]                = EfieldY[0+1][0+1][0+1];
417     EfieldY[isteps-1][0][0]         = EfieldY[isteps-2][0+1][0+1];
418     EfieldY[isteps-1][jsteps-2][0]  = EfieldY[isteps-2][jsteps-3][0+1];
419     EfieldY[0][jsteps-2][0]         = EfieldY[0+1][jsteps-3][0+1];
420     EfieldY[0][0][ksteps-1]                 = EfieldY[0+1][0+1][ksteps-2];
421     EfieldY[isteps-1][0][ksteps-1]          = EfieldY[isteps-2][0+1][ksteps-2];
422     EfieldY[isteps-1][jsteps-2][ksteps-1]   = EfieldY[isteps-2][jsteps-3][ksteps-2];
423     EfieldY[0][jsteps-2][ksteps-1]          = EfieldY[0+1][jsteps-3][ksteps-2];
424
425     EfieldZ[0][0][0]                = EfieldZ[0+1][0+1][0+1];
426     EfieldZ[isteps-1][0][0]         = EfieldZ[isteps-2][0+1][0+1];
427     EfieldZ[isteps-1][jsteps-1][0]  = EfieldZ[isteps-2][jsteps-2][0+1];
428     EfieldZ[0][jsteps-1][0]         = EfieldZ[0+1][jsteps-2][0+1];
429     EfieldZ[0][0][ksteps-2]                 = EfieldZ[0+1][0+1][ksteps-3];
430     EfieldZ[isteps-1][0][ksteps-2]          = EfieldZ[isteps-2][0+1][ksteps-3];
431     EfieldZ[isteps-1][jsteps-1][ksteps-2]   = EfieldZ[isteps-2][jsteps-2][ksteps-3];
432     EfieldZ[0][jsteps-1][ksteps-2]          = EfieldZ[0+1][jsteps-2][ksteps-3];
433 }
434
435 // This function takes in the Ez and Ey field and applies the Mur 2nd Order Radiating
436 // Boundary Conditions. This function is built for a field on the X boundary. Ie having an
437 // update equation that relies on deltaX. Currently the function is built for the Efield, but
438 // may be applicable to the Hfields. This is all related to the co-ordinate system reference
439 // used. My reference is on PG 73 of Umran FDTD TB. All Matrixes/Arrays are passed by
440 // pointed, so the function operates on the original memory.
441 void FDTDCart3D::Xplanebound_Mur2nRBC(){
442
443   //Boundary Conditions for Ez the x=0 and x=max walls. Entire Y-Z plane.
444     for(int j=1; j != jsteps-1; j++){
445       for(int k=1; k != ksteps-2; k++){// Note the size difference for this field range.
446         EfieldZ[0][j][k] = -EzXLowerBndrTmin2[1][j][k] - ((deltaX - vp*deltaT)/(deltaX + vp*↩
                deltaT))*(EfieldZ[1][j][k] + EzXLowerBndrTmin2[0][j][k]) + ((2*deltaX)/(deltaX +↩
                 vp*deltaT))*(EzXLowerBndrTmin1[0][j][k] + EzXLowerBndrTmin1[1][j][k]) + ((↩
                deltaX*vp*deltaT*vp*deltaT)/(2*deltaY*deltaY*(deltaX+vp*deltaT)))*(↩
                EzXLowerBndrTmin1[0][j+1][k] -2*EzXLowerBndrTmin1[0][j][k] + EzXLowerBndrTmin1↩
                [0][j-1][k] + EzXLowerBndrTmin1[1][j+1][k] -2*EzXLowerBndrTmin1[1][j][k] + ↩
                EzXLowerBndrTmin1[1][j-1][k] ) + ((deltaX*vp*deltaT*vp*deltaT)/(2*deltaZ*deltaZ↩
                *(deltaX+vp*deltaT)))*(EzXLowerBndrTmin1[0][j][k+1] -2*EzXLowerBndrTmin1[0][j][k↩
                ] + EzXLowerBndrTmin1[0][j][k-1] + EzXLowerBndrTmin1[1][j][k+1] -2*↩
                EzXLowerBndrTmin1[1][j][k] + EzXLowerBndrTmin1[1][j][k-1] ) ;
447
448         EfieldZ[isteps-1][j][k] = -EzXUpperBndrTmin2[0][j][k] - ((deltaX - vp*deltaT)/(deltaX↩
                 + vp*deltaT))*(EfieldZ[isteps-2][j][k] + EzXUpperBndrTmin2[1][j][k]) + ((2*↩
                deltaX)/(deltaX + vp*deltaT))*(EzXUpperBndrTmin1[1][j][k] + EzXUpperBndrTmin1↩
                [0][j][k]) + ((deltaX*vp*deltaT*vp*deltaT)/(2*deltaY*deltaY*(deltaX+vp*deltaT)))↩
                *(EzXUpperBndrTmin1[1][j+1][k] -2*EzXUpperBndrTmin1[1][j][k] + EzXUpperBndrTmin1↩
                [1][j-1][k] + EzXUpperBndrTmin1[0][j+1][k] -2*EzXUpperBndrTmin1[0][j][k] + ↩
                EzXUpperBndrTmin1[0][j-1][k] ) + ((deltaX*vp*deltaT*vp*deltaT)/(2*deltaZ*deltaZ↩
                *(deltaX+vp*deltaT)))*(EzXUpperBndrTmin1[1][j][k+1] -2*EzXUpperBndrTmin1[1][j][k↩
                ] + EzXUpperBndrTmin1[1][j][k-1] + EzXUpperBndrTmin1[0][j][k+1] -2*↩
                EzXUpperBndrTmin1[0][j][k] + EzXUpperBndrTmin1[0][j][k-1] );
449       }
450     }
451
452   //Updating the min 2 timestep values for the next time step run.
453     for(int j=0; j != jsteps; j++){
454       for(int k=0; k != ksteps-1; k++){
455         EzXLowerBndrTmin2[0][j][k] = EzXLowerBndrTmin1[0][j][k];
456         EzXLowerBndrTmin2[1][j][k] = EzXLowerBndrTmin1[1][j][k];
457         EzXLowerBndrTmin1[0][j][k] = EfieldZ[0][j][k];
458         EzXLowerBndrTmin1[1][j][k] = EfieldZ[1][j][k];
459
460         EzXUpperBndrTmin2[0][j][k] = EzXUpperBndrTmin1[0][j][k];
461         EzXUpperBndrTmin2[1][j][k] = EzXUpperBndrTmin1[1][j][k];
462         EzXUpperBndrTmin1[0][j][k] = EfieldZ[isteps-2][j][k];   //inner 0
463         EzXUpperBndrTmin1[1][j][k] = EfieldZ[isteps-1][j][k];   //outer 1
464       }
465     }
```

```
466
467   //Boundary Conditions Ey field at the x=0 and x=max walls. Entire Y-Z plane.
468     for(int j=1; j != jsteps-2; j++){ // Note the size difference for this field range.
469       for(int k=1; k != ksteps-1; k++){
470         EfieldY[0][j][k] = -EyXLowerBndrTmin2[1][j][k] - ((deltaX - vp*deltaT)/(deltaX + vp*↩
              deltaT))*(EfieldY[1][j][k] + EyXLowerBndrTmin2[0][j][k]) + ((2*deltaX)/(deltaX +↩
               vp*deltaT))*(EyXLowerBndrTmin1[0][j][k] + EyXLowerBndrTmin1[1][j][k]) + ((↩
              deltaX*vp*deltaT*vp*deltaT)/(2*deltaY*deltaY*(deltaX+vp*deltaT)))*(↩
              EyXLowerBndrTmin1[0][j+1][k] -2*EyXLowerBndrTmin1[0][j][k] + EyXLowerBndrTmin1↩
              [0][j-1][k] + EyXLowerBndrTmin1[1][j+1][k] -2*EyXLowerBndrTmin1[1][j][k] + ↩
              EyXLowerBndrTmin1[1][j-1][k] ) + ((deltaX*vp*deltaT*vp*deltaT)/(2*deltaZ*deltaZ↩
              *(deltaX+vp*deltaT)))*(EyXLowerBndrTmin1[0][j][k+1] -2*EyXLowerBndrTmin1[0][j][k↩
              ] + EyXLowerBndrTmin1[0][j][k-1] + EyXLowerBndrTmin1[1][j][k+1] -2*↩
              EyXLowerBndrTmin1[1][j][k] + EyXLowerBndrTmin1[1][j][k-1] ) ;
471
472         EfieldY[isteps-1][j][k] = -EyXUpperBndrTmin2[0][j][k] - ((deltaX - vp*deltaT)/(deltaX↩
               + vp*deltaT))*(EfieldY[isteps-2][j][k] + EyXUpperBndrTmin2[1][j][k]) + ((2*↩
              deltaX)/(deltaX + vp*deltaT))*(EyXUpperBndrTmin1[1][j][k] + EyXUpperBndrTmin1↩
              [0][j][k]) + ((deltaX*vp*deltaT*vp*deltaT)/(2*deltaY*deltaY*(deltaX+vp*deltaT)))↩
              *(EyXUpperBndrTmin1[1][j+1][k] -2*EyXUpperBndrTmin1[1][j][k] + EyXUpperBndrTmin1↩
              [1][j-1][k] + EyXUpperBndrTmin1[0][j+1][k] -2*EyXUpperBndrTmin1[0][j][k] + ↩
              EyXUpperBndrTmin1[0][j-1][k] ) + ((deltaX*vp*deltaT*vp*deltaT)/(2*deltaZ*deltaZ↩
              *(deltaX+vp*deltaT)))*(EyXUpperBndrTmin1[1][j][k+1] -2*EyXUpperBndrTmin1[1][j][k↩
              ] + EyXUpperBndrTmin1[1][j][k-1] + EyXUpperBndrTmin1[0][j][k+1] -2*↩
              EyXUpperBndrTmin1[0][j][k] + EyXUpperBndrTmin1[0][j][k-1] );
473       }
474     }
475
476   //Updating the min 2 timestep values for the next time step run.
477     for(int j=0; j != jsteps-1; j++){
478       for(int k=0; k != ksteps; k++){
479         EyXLowerBndrTmin2[0][j][k] = EyXLowerBndrTmin1[0][j][k];
480         EyXLowerBndrTmin2[1][j][k] = EyXLowerBndrTmin1[1][j][k];
481         EyXLowerBndrTmin1[0][j][k] = EfieldY[0][j][k];
482         EyXLowerBndrTmin1[1][j][k] = EfieldY[1][j][k];
483
484         EyXUpperBndrTmin2[0][j][k] = EyXUpperBndrTmin1[0][j][k];
485         EyXUpperBndrTmin2[1][j][k] = EyXUpperBndrTmin1[1][j][k];
486         EyXUpperBndrTmin1[0][j][k] = EfieldY[isteps-2][j][k];   //inner 0
487         EyXUpperBndrTmin1[1][j][k] = EfieldY[isteps-1][j][k];   //outer 1
488       }
489     }
490 }
491
492 // This function takes in the Ez and Ex field and applies the Mur 2nd Order Radiating
493 // Boundary Conditions. This function is built for a field on the Y boundary. Ie having an
494 // update equation that relies on deltaY. Currently the function is built for the Efield, but
495 // may be applicable to the Hfields. This is all related to the co-ordinate system reference
496 // used. My reference is on PG 73 of Umran FDTD TB. All Matrixes/Arrays are passed by
497 // pointed, so the function operates on the original memory.
498 void FDTDCart3D::Yplanebound_Mur2nRBC(){
499
500   //Boundary Conditions for Ez field at the y=0 and y=max walls. Entire X-Z plane.
501     for(int i=1; i != isteps-1; i++){
502       for(int k=1; k != ksteps-2; k++){
503         EfieldZ[i][0][k] = -EzYLowerBndrTmin2[i][1][k] - ((deltaY - vp*deltaT)/(deltaY + vp*↩
              deltaT))*(EfieldZ[i][1][k] + EzYLowerBndrTmin2[i][0][k]) + ((2*deltaY)/(deltaY +↩
               vp*deltaT))*(EzYLowerBndrTmin1[i][0][k] + EzYLowerBndrTmin1[i][1][k]) + ((↩
              deltaY*vp*deltaT*vp*deltaT)/(2*deltaX*deltaX*(deltaY+vp*deltaT)))*(↩
              EzYLowerBndrTmin1[i+1][0][k] -2*EzYLowerBndrTmin1[i][0][k] + EzYLowerBndrTmin1[i↩
              -1][0][k] + EzYLowerBndrTmin1[i+1][1][k] -2*EzYLowerBndrTmin1[i][1][k] + ↩
              EzYLowerBndrTmin1[i-1][1][k] ) + ((deltaY*vp*deltaT*vp*deltaT)/(2*deltaZ*deltaZ↩
              *(deltaY+vp*deltaT)))*(EzYLowerBndrTmin1[i][0][k+1] -2*EzYLowerBndrTmin1[i][0][k↩
              ] + EzYLowerBndrTmin1[i][0][k-1] + EzYLowerBndrTmin1[i][1][k+1] -2*↩
              EzYLowerBndrTmin1[i][1][k] + EzYLowerBndrTmin1[i][1][k-1] ) ;
504
```

```
505         EfieldZ[i][jsteps-1][k] = -EzYUpperBndrTmin2[i][0][k] - ((deltaY - vp*deltaT)/(deltaY+
                + vp*deltaT))*(EfieldZ[i][jsteps-2][k] + EzYUpperBndrTmin2[i][1][k]) + ((2*
                deltaY)/(deltaY + vp*deltaT))*(EzYUpperBndrTmin1[i][1][k] + EzYUpperBndrTmin1[i
                ][0][k]) + ((deltaY*vp*deltaT*vp*deltaT)/(2*deltaX*deltaX*(deltaY+vp*deltaT)))*(
                EzYUpperBndrTmin1[i+1][1][k] -2*EzYUpperBndrTmin1[i][1][k] + EzYUpperBndrTmin1[i
                -1][1][k] + EzYUpperBndrTmin1[i+1][0][k] -2*EzYUpperBndrTmin1[i][0][k] +
                EzYUpperBndrTmin1[i-1][0][k] ) + ((deltaY*vp*deltaT*vp*deltaT)/(2*deltaZ*deltaZ
                *(deltaY+vp*deltaT)))*(EzYUpperBndrTmin1[i][1][k+1] -2*EzYUpperBndrTmin1[i][1][k
                ] + EzYUpperBndrTmin1[i][1][k-1] + EzYUpperBndrTmin1[i][0][k+1] -2*
                EzYUpperBndrTmin1[i][0][k] + EzYUpperBndrTmin1[i][0][k-1] );
506       }
507     }
508
509   //Updating the min 2 timestep values for the next time step run.
510     for(int i=0; i != isteps; i++){
511       for(int k=0; k != ksteps-1; k++){
512         EzYLowerBndrTmin2[i][0][k] = EzYLowerBndrTmin1[i][0][k];
513         EzYLowerBndrTmin2[i][1][k] = EzYLowerBndrTmin1[i][1][k];
514         EzYLowerBndrTmin1[i][0][k] = EfieldZ[i][0][k];
515         EzYLowerBndrTmin1[i][1][k] = EfieldZ[i][1][k];
516
517         EzYUpperBndrTmin2[i][0][k] = EzYUpperBndrTmin1[i][0][k];
518         EzYUpperBndrTmin2[i][1][k] = EzYUpperBndrTmin1[i][1][k];
519         EzYUpperBndrTmin1[i][0][k] = EfieldZ[i][jsteps-2][k];   //inner 0
520         EzYUpperBndrTmin1[i][1][k] = EfieldZ[i][jsteps-1][k];   //outer 1
521       }
522     }
523
524   //Boundary Conditions for Ex field at the y=0 and y=max walls. Entire X-Z plane.
525     for(int i=1; i != isteps-2; i++){
526       for(int k=1; k != ksteps-1; k++){
527         EfieldX[i][0][k] = -ExYLowerBndrTmin2[i][1][k] - ((deltaY - vp*deltaT)/(deltaY + vp*
                deltaT))*(EfieldX[i][1][k] + ExYLowerBndrTmin2[i][0][k]) + ((2*deltaY)/(deltaY +
                 vp*deltaT))*(ExYLowerBndrTmin1[i][0][k] + ExYLowerBndrTmin1[i][1][k]) + ((
                deltaY*vp*deltaT*vp*deltaT)/(2*deltaX*deltaX*(deltaY+vp*deltaT)))*(
                ExYLowerBndrTmin1[i+1][0][k] -2*ExYLowerBndrTmin1[i][0][k] + ExYLowerBndrTmin1[i
                -1][0][k] + ExYLowerBndrTmin1[i+1][1][k] -2*ExYLowerBndrTmin1[i][1][k] +
                ExYLowerBndrTmin1[i-1][1][k] ) + ((deltaY*vp*deltaT*vp*deltaT)/(2*deltaZ*deltaZ
                *(deltaY+vp*deltaT)))*(ExYLowerBndrTmin1[i][0][k+1] -2*ExYLowerBndrTmin1[i][0][k
                ] + ExYLowerBndrTmin1[i][0][k-1] + ExYLowerBndrTmin1[i][1][k+1] -2*
                ExYLowerBndrTmin1[i][1][k] + ExYLowerBndrTmin1[i][1][k-1] ) ;
528
529         EfieldX[i][jsteps-1][k] = -ExYUpperBndrTmin2[i][0][k] - ((deltaY - vp*deltaT)/(deltaY+
                + vp*deltaT))*(EfieldX[i][jsteps-2][k] + ExYUpperBndrTmin2[i][1][k]) + ((2*
                deltaY)/(deltaY + vp*deltaT))*(ExYUpperBndrTmin1[i][1][k] + ExYUpperBndrTmin1[i
                ][0][k]) + ((deltaY*vp*deltaT*vp*deltaT)/(2*deltaX*deltaX*(deltaY+vp*deltaT)))*(
                ExYUpperBndrTmin1[i+1][1][k] -2*ExYUpperBndrTmin1[i][1][k] + ExYUpperBndrTmin1[i
                -1][1][k] + ExYUpperBndrTmin1[i+1][0][k] -2*ExYUpperBndrTmin1[i][0][k] +
                ExYUpperBndrTmin1[i-1][0][k] ) + ((deltaY*vp*deltaT*vp*deltaT)/(2*deltaZ*deltaZ
                *(deltaY+vp*deltaT)))*(ExYUpperBndrTmin1[i][1][k+1] -2*ExYUpperBndrTmin1[i][1][k
                ] + ExYUpperBndrTmin1[i][1][k-1] + ExYUpperBndrTmin1[i][0][k+1] -2*
                ExYUpperBndrTmin1[i][0][k] + ExYUpperBndrTmin1[i][0][k-1] );
530       }
531     }
532
533   //Updating the min 2 timestep values for the next time step run.
534     for(int i=0; i != isteps-1; i++){
535       for(int k=0; k != ksteps; k++){
536         ExYLowerBndrTmin2[i][0][k] = ExYLowerBndrTmin1[i][0][k];
537         ExYLowerBndrTmin2[i][1][k] = ExYLowerBndrTmin1[i][1][k];
538         ExYLowerBndrTmin1[i][0][k] = EfieldX[i][0][k];
539         ExYLowerBndrTmin1[i][1][k] = EfieldX[i][1][k];
540
541         ExYUpperBndrTmin2[i][0][k] = ExYUpperBndrTmin1[i][0][k];
542         ExYUpperBndrTmin2[i][1][k] = ExYUpperBndrTmin1[i][1][k];
543         ExYUpperBndrTmin1[i][0][k] = EfieldX[i][jsteps-2][k];   //inner 0
544         ExYUpperBndrTmin1[i][1][k] = EfieldX[i][jsteps-1][k];   //outer 1
545       }
```

```
546     }
547 }
548 
549 // This function takes in the Ey and Ex field and applies the Mur 2nd Order Radiating
550 // Boundary Conditions. This function is built for a field on the Z boundary. Ie having an
551 // update equation that relies on deltaZ. Currently the function is built for the Efield, but
552 // may be applicable to the Hfields. This is all related to the co-ordinate system reference
553 // used. My reference is on PG 73 of Umran FDTD TB. All Matrixes/Arrays are passed by
554 // pointed, so the function operates on the original memory.
555 void FDTDCart3D::Zplanebound_Mur2nRBC(){
556 
557   //Boundary Conditions for Ey field at the z=0 and z=max walls. Entire X-Y plane.
558     for(int i=1; i != isteps-1; i++){
559       for(int j=1; j != jsteps-2; j++){
560         // !!!!! Commented out for the Ground plane !!!!!
561         // EfieldY[i][j][0] = -EyZLowerBndrTmin2[i][j][1] - ((deltaZ - vp*deltaT)/(deltaZ + ↩
              vp*deltaT))*(EfieldY[i][j][1] + EyZLowerBndrTmin2[i][j][0]) + ((2*deltaZ)/(↩
              deltaZ + vp*deltaT))*(EyZLowerBndrTmin1[i][j][0] + EyZLowerBndrTmin1[i][j][1]) +↩
               ((deltaZ*vp*deltaT*vp*deltaT)/(2*deltaX*deltaX*(deltaZ+vp*deltaT)))*(↩
              EyZLowerBndrTmin1[i+1][j][0] -2*EyZLowerBndrTmin1[i][j][0] + EyZLowerBndrTmin1[i↩
              -1][j][0] + EyZLowerBndrTmin1[i+1][j][1] -2*EyZLowerBndrTmin1[i][j][1] + ↩
              EyZLowerBndrTmin1[i-1][j][1] ) + ((deltaZ*vp*deltaT*vp*deltaT)/(2*deltaY*deltaY↩
              *(deltaZ+vp*deltaT)))*(EyZLowerBndrTmin1[i][j+1][0] -2*EyZLowerBndrTmin1[i][j↩
              ][0] + EyZLowerBndrTmin1[i][j-1][0] + EyZLowerBndrTmin1[i][j+1][1] -2*↩
              EyZLowerBndrTmin1[i][j][1] + EyZLowerBndrTmin1[i][j-1][1] ) ;
562 
563         EfieldY[i][j][ksteps-1] = -EyZUpperBndrTmin2[i][j][0] - ((deltaZ - vp*deltaT)/(deltaZ↩
               + vp*deltaT))*(EfieldY[i][j][ksteps-2] + EyZUpperBndrTmin2[i][j][1]) + ((2*↩
              deltaZ)/(deltaZ + vp*deltaT))*(EyZUpperBndrTmin1[i][j][1] + EyZUpperBndrTmin1[i↩
              ][j][0]) + ((deltaZ*vp*deltaT*vp*deltaT)/(2*deltaX*deltaX*(deltaZ+vp*deltaT)))*(↩
              EyZUpperBndrTmin1[i+1][j][1] -2*EyZUpperBndrTmin1[i][j][1] + EyZUpperBndrTmin1[i↩
              -1][j][1] + EyZUpperBndrTmin1[i+1][j][0] -2*EyZUpperBndrTmin1[i][j][0] + ↩
              EyZUpperBndrTmin1[i-1][j][0] ) + ((deltaZ*vp*deltaT*vp*deltaT)/(2*deltaY*deltaY↩
              *(deltaZ+vp*deltaT)))*(EyZUpperBndrTmin1[i][j+1][1] -2*EyZUpperBndrTmin1[i][j↩
              ][1] + EyZUpperBndrTmin1[i][j-1][1] + EyZUpperBndrTmin1[i][j+1][0] -2*↩
              EyZUpperBndrTmin1[i][j][0] + EyZUpperBndrTmin1[i][j-1][0] );
564       }
565     }
566 
567   //Updating the min 2 timestep values for the next time step run.
568     for(int i=0; i != isteps; i++){
569       for(int j=0; j != jsteps-1; j++){
570         // !!!!! Commented out for the Ground plane !!!!!
571         // EyZLowerBndrTmin2[i][j][0] = EyZLowerBndrTmin1[i][j][0];
572         // EyZLowerBndrTmin2[i][j][1] = EyZLowerBndrTmin1[i][j][1];
573         // EyZLowerBndrTmin1[i][j][0] = EfieldY[i][j][0];
574         // EyZLowerBndrTmin1[i][j][1] = EfieldY[i][j][1];
575 
576         EyZUpperBndrTmin2[i][j][0] = EyZUpperBndrTmin1[i][j][0];
577         EyZUpperBndrTmin2[i][j][1] = EyZUpperBndrTmin1[i][j][1];
578         EyZUpperBndrTmin1[i][j][0] = EfieldY[i][j][ksteps-2];   //inner 0
579         EyZUpperBndrTmin1[i][j][1] = EfieldY[i][j][ksteps-1];   //outer 1
580       }
581     }
582 
583   //Boundary Conditions for Ex field at the z=0 and z=max walls. Entire X-Y plane.
584     for(int i=1; i != isteps-2; i++){
585       for(int j=1; j != jsteps-1; j++){
586         // !!!!! Commented out for the Ground plane !!!!!
```

```
587         // EfieldX[i][j][0] = -ExZLowerBndrTmin2[i][j][1] - ((deltaZ - vp*deltaT)/(deltaZ + ↩
                vp*deltaT))*(EfieldX[i][j][1] + ExZLowerBndrTmin2[i][j][0]) + ((2*deltaZ)/(↩
                deltaZ + vp*deltaT))*(ExZLowerBndrTmin1[i][j][0] + ExZLowerBndrTmin1[i][j][1]) +↩
                 ((deltaZ*vp*deltaT*vp*deltaT)/(2*deltaX*deltaX*(deltaZ+vp*deltaT)))*(↩
                ExZLowerBndrTmin1[i+1][j][0] -2*ExZLowerBndrTmin1[i][j][0] + ExZLowerBndrTmin1[i↩
                -1][j][0] + ExZLowerBndrTmin1[i+1][j][1] -2*ExZLowerBndrTmin1[i][j][1] + ↩
                ExZLowerBndrTmin1[i-1][j][1] ) + ((deltaZ*vp*deltaT*vp*deltaT)/(2*deltaY*deltaY↩
                *(deltaZ+vp*deltaT)))*(ExZLowerBndrTmin1[i][j+1][0] -2*ExZLowerBndrTmin1[i][j↩
                ][0] + ExZLowerBndrTmin1[i][j-1][0] + ExZLowerBndrTmin1[i][j+1][1] -2*↩
                ExZLowerBndrTmin1[i][j][1] + ExZLowerBndrTmin1[i][j-1][1] ) ;
588
589         EfieldX[i][j][ksteps-1] = -ExZUpperBndrTmin2[i][j][0] - ((deltaZ - vp*deltaT)/(deltaZ↩
                 + vp*deltaT))*(EfieldX[i][j][ksteps-2] + ExZUpperBndrTmin2[i][j][1]) + ((2*↩
                deltaZ)/(deltaZ + vp*deltaT))*(ExZUpperBndrTmin1[i][j][1] + ExZUpperBndrTmin1[i↩
                ][j][0]) + ((deltaZ*vp*deltaT*vp*deltaT)/(2*deltaX*deltaX*(deltaZ+vp*deltaT)))*(↩
                ExZUpperBndrTmin1[i+1][j][1] -2*ExZUpperBndrTmin1[i][j][1] + ExZUpperBndrTmin1[i↩
                -1][j][1] + ExZUpperBndrTmin1[i+1][j][0] -2*ExZUpperBndrTmin1[i][j][0] + ↩
                ExZUpperBndrTmin1[i-1][j][0] ) + ((deltaZ*vp*deltaT*vp*deltaT)/(2*deltaY*deltaY↩
                *(deltaZ+vp*deltaT)))*(ExZUpperBndrTmin1[i][j+1][1] -2*ExZUpperBndrTmin1[i][j↩
                ][1] + ExZUpperBndrTmin1[i][j-1][1] + ExZUpperBndrTmin1[i][j+1][0] -2*↩
                ExZUpperBndrTmin1[i][j][0] + ExZUpperBndrTmin1[i][j-1][0] );
590       }
591     }
592
593   //Updating the min 2 timestep values for the next time step run.
594     for(int i=0; i != isteps-1; i++){
595       for(int j=0; j != jsteps; j++){
596         // !!!!! Commented out for the Ground plane !!!!!
597         // ExZLowerBndrTmin2[i][j][0] = ExZLowerBndrTmin1[i][j][0];
598         // ExZLowerBndrTmin2[i][j][1] = ExZLowerBndrTmin1[i][j][1];
599         // ExZLowerBndrTmin1[i][j][0] = EfieldX[i][j][0];
600         // ExZLowerBndrTmin1[i][j][1] = EfieldX[i][j][1];
601
602         ExZUpperBndrTmin2[i][j][0] = ExZUpperBndrTmin1[i][j][0];
603         ExZUpperBndrTmin2[i][j][1] = ExZUpperBndrTmin1[i][j][1];
604         ExZUpperBndrTmin1[i][j][0] = EfieldX[i][j][ksteps-2];   //inner 0
605         ExZUpperBndrTmin1[i][j][1] = EfieldX[i][j][ksteps-1];   //outer 1
606       }
607     }
608 }
609
610 // This function does 1st order Mur boundary condition for the edges of the simulation space.
611 void FDTDCart3D::Xupright_edges(){
612     for(int i=1; i!=isteps-1; i++){
613       EfieldX[i][0][0] = EfieldX[i][0+1][0+1];
614       EfieldX[i][jsteps-1][0] = EfieldX[i][jsteps-1-1][0+1];
615       EfieldX[i][0][ksteps-1] = EfieldX[i][0+1][ksteps-1-1];
616       EfieldX[i][jsteps-1][ksteps-1] = EfieldX[i][jsteps-1-1][ksteps-1-1];
617     }
618
619     for(int i=1; i!=isteps; i++){
620       EfieldY[i][0][0] = EfieldY[i][0+1][0+1];
621       EfieldY[i][jsteps-2][0] = EfieldY[i][jsteps-3][0+1];
622       EfieldY[i][0][ksteps-1] = EfieldY[i][0+1][ksteps-1-1];
623       EfieldY[i][jsteps-2][ksteps-1] = EfieldY[i][jsteps-3][ksteps-1-1];
624     }
625
626     for(int i=1; i!=isteps; i++){
627       EfieldZ[i][0][0] = EfieldZ[i][0+1][0+1];
628       EfieldZ[i][jsteps-1][0] = EfieldZ[i][jsteps-1-1][0+1];
629       EfieldZ[i][0][ksteps-2] = EfieldZ[i][0+1][ksteps-3];
630       EfieldZ[i][jsteps-1][ksteps-2] = EfieldZ[i][jsteps-1-1][ksteps-3];
631     }
632 }
633
634 // This function does 1st order Mur boundary condition for the edges of the simulation space.
635 void FDTDCart3D::Yupright_edges(){
636     for(int j=1; j!=jsteps; j++){
```

```
637       EfieldX[0][j][0] = EfieldX[0+1][j][0+1];
638       EfieldX[isteps-2][j][0] = EfieldX[isteps-3][j][0+1];
639       EfieldX[0][j][ksteps-1] = EfieldX[0+1][j][ksteps-1-1];
640       EfieldX[isteps-2][j][ksteps-1] = EfieldX[isteps-3][j][ksteps-1-1];
641     }
642
643     for(int j=1; j!=jsteps-1; j++){
644       EfieldY[0][j][0] = EfieldY[0+1][j][0+1];
645       EfieldY[isteps-1][j][0] = EfieldY[isteps-1-1][j][0+1];
646       EfieldY[0][j][ksteps-1] = EfieldY[0+1][j][ksteps-1-1];
647       EfieldY[isteps-1][j][ksteps-1] = EfieldY[isteps-1-1][j][ksteps-1-1];
648     }
649
650     for(int j=1; j!=jsteps; j++){
651       EfieldZ[0][j][0] = EfieldZ[0+1][j][0+1];
652       EfieldZ[isteps-1][j][0] = EfieldZ[isteps-1-1][j][0+1];
653       EfieldZ[0][j][ksteps-2] = EfieldZ[0+1][j][ksteps-3];
654       EfieldZ[isteps-1][j][ksteps-2] = EfieldZ[isteps-1-1][j][ksteps-3];
655     }
656 }
657
658 // This function does 1st order Mur boundary condition for the edges of the simulation space.
659 void FDTDCart3D::Zupright_edges(){
660     for(int k=1; k!=ksteps; k++){
661       EfieldX[0][0][k] = EfieldX[0+1][0+1][k];
662       EfieldX[isteps-2][0][k] = EfieldX[isteps-3][0+1][k];
663       EfieldX[0][jsteps-1][k] = EfieldX[0+1][jsteps-1-1][k];
664       EfieldX[isteps-2][jsteps-1][k] = EfieldX[isteps-3][jsteps-1-1][k];
665     }
666
667     for(int k=1; k!=ksteps; k++){
668       EfieldY[0][0][k] = EfieldY[0+1][0+1][k];
669       EfieldY[isteps-1][0][k] = EfieldY[isteps-1-1][0+1][k];
670       EfieldY[0][jsteps-2][k] = EfieldY[0+1][jsteps-3][k];
671       EfieldY[isteps-1][jsteps-2][k] = EfieldY[isteps-1-1][jsteps-3][k];
672     }
673
674     for(int k=1; k!=ksteps-1; k++){
675       EfieldZ[0][0][k] = EfieldZ[0+1][0+1][k];
676       EfieldZ[isteps-1][0][k] = EfieldZ[isteps-1-1][0+1][k];
677       EfieldZ[0][jsteps-1][k] = EfieldZ[0+1][jsteps-1-1][k];
678       EfieldZ[isteps-1][jsteps-1][k] = EfieldZ[isteps-1-1][jsteps-1-1][k];
679     }
680 }
```

Listing D.4: Makefile

```
1  CC=g++
2  CFLAGS=-c -g -Wall -std=c++11 -fpermissive
3  LFLAGS=
4  all:
5    rm -rf build
6    mkdir -p build
7    $(CC) $(CFLAGS) main.cpp -o ./build/main.o $(LFLAGS)
8    $(CC) $(CFLAGS) FDTDCart3D.cpp -o ./build/FDTDCart3D.o $(LFLAGS)
9    $(CC) ./build/FDTDCart3D.o ./build/main.o -o Lightning_FDTD $(LFLAGS)
10
11 optimized:
12   rm -rf build
13   mkdir -p build
14   $(CC) -O3 $(CFLAGS) main.cpp -o ./build/main.o $(LFLAGS)
15   $(CC) -O3 $(CFLAGS) FDTDCart3D.cpp -o ./build/FDTDCart3D.o $(LFLAGS)
16   $(CC) ./build/FDTDCart3D.o  ./build/main.o -o Lightning_FDTD_OPT $(LFLAGS)
```

Listing D.5: 3D_Settings_File.csv

```
tau1,0.072,us,
tau2,5,us,
tau3,100,us,
tau4,6,us,
lamdac,2000,nucciconst,
currentDecay,0,trueorfalse,
I1,9900,A,
I2,7500,A,
I3,9550,A,
eta,0.845,number,
Na,3,teresnumber,
rObs,17.36,m,
hObs,98.48,m,
vp,299792458,m/s,
vchan,150000000,m/s,
f0,2000000,hz,
gridSizeX,400,m ,
gridSizeY,400,m ,
gridSizeZ,1500,m ,
deltaX,1.736,m ,
deltaY,3.472,m ,
deltaZ,9.848,m ,
deltaT,1,ns,
useDTfromfile,0,trueorfalse,
simRunTime,2334,ns,
```

Appendix E

2D Cylindrical FDTD Code

Listing E.1: main_Cylin2D.cpp

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;

// My Header Files
#include "FDTDCylin2D.h"

int main(int argc, char const *argv[]){
  FDTDCylin2D lightningStroke2D("2D_Cylin_Settings_File.csv", argc, argv);
  lightningStroke2D.PrintMinimalSimParameters();
  lightningStroke2D.EvaluateFDTDPlane();
}
```

Listing E.2: FDTDCylin2D.h

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;

class FDTDCylin2D{
public:
  FDTDCylin2D(string settingsFile, int argc, char const *argv[]);
  void EvaluateFDTDPlane();
  void setSimulationParameters(string settingsFile);
  void setCMDLineParameters(float TrObs, float ThObs, float TgridSizeRadial, float TgridSizeZ↩
       , float TdeltaD, float TdeltaZ, float TsimRunTime);
  void PrintMinimalSimParameters();
  double NucciTeresCurrent(double z, double t);

  // ****************************************************************
  // ******************** Boundary Related **************************
  void HphiField_Mur2nRBC(double** field);

```

```
22 private:
23   // These are variables provided in the Settings file. If more variables are added to the
24   // settings file then add them here as well as the "setSimulationParameters" function.
25   float vp, vchan;                // Propagation and Channel Velocity
26   float gridSizeRadial, gridSizeZ;
27   float deltaD, deltaZ, deltaT;
28   int useDTfromfile;
29   float simRunTime;
30   double tau1, tau2, tau3, tau4;
31   double lamdac;
32   int currentDecay;
33   double I1, I2, I3;
34   int Na;
35   double eta, omega0;
36   double rObs,hObs;
37
38   // These are the normal class variables. They need to be assigned in the constructor.
39   float idealGridSize;
40   float optimalDeltaT;
41   float EdFSC;
42   float EzFSC;
43   float HpFSC;
44   float AxisFSC;
45
46   // The "steps" define the number of steps in the respective directions. The origin is in
47   // the middle of the plane with gridsize variable length in every direction of the origin.
48   int isteps;
49   int ksteps;
50   int nSteps;
51   int iorigin;
52   int korigin;
53   int XOBS, ZOBS;
54
55   double** HfieldPHI;
56   double** EfieldD;
57   double** EfieldZ;
58   // These are the upper and lower boundaries of the evaluation plane. This is the right and
59   // left hand walls respectively. In the lower boundary the outer column is 0, and the inner
60   // is 1. In the Upper boundary the outer column is 1, and the inner is 0.
61   double** HphiXLowerBndrTmin1;
62   double** HphiXUpperBndrTmin1;
63   double** HphiXLowerBndrTmin2;
64   double** HphiXUpperBndrTmin2;
65   double** HphiZLowerBndrTmin1;
66   double** HphiZUpperBndrTmin1;
67   double** HphiZLowerBndrTmin2;
68   double** HphiZUpperBndrTmin2;
69
70   // These are the pre-calculated variables used to save computational time.
71   const double pi = ( 4.0 * atan( 1.0 ) );
72   const double c  = 299792458;
73   const double mu = 4 * pi * pow( 10.0, -7.0 );
74   const double epsilon = 8.8541878176 * pow( 10.0, -12.0 );
75   float SINEwaveConstant;
76   float theta;
77   float timeToReachObservationPoint;
78   int percentProgress;
79   int factorial[13];
80
81   char partATestID[128];
82   char partBTestID[128];
83   char EdFieldcurvename[128];
84   char EzFieldcurvename[128];
85   char HpFieldcurvename[128];
86   char ErFieldcurvename[128];
87   char EtFieldcurvename[128];
88 };
```

Listing E.3: FDTDCylin2D.cpp

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <string>
#include <algorithm>

#include "FDTDCylin2D.h"

using namespace std;

// This is a macro for the 2D array instantiation. This is a way to create a contiguous array
// of memory. Standard methods don't work due to array sizes, and heap memory size. The last
// index is K. IE, this is the index that should be used the most. From my understanding
// when memory is called for use the compiler will load a batch of the surrounding memory
// points. Therefore you will see in later code that the K index is always in the inner
// most loop (called most often). This macro also zero's the array.
#define create2Darray(fieldname, isteps, ksteps)           \
    double** fieldname = (double**) new double*[isteps];   \
    for(int i = 0; i < isteps; i++) {                      \
      fieldname[i]=(double*)new double[ksteps];            \
    }                                                      \
    for(int i=0; i != isteps ; i++){                       \
      for(int j=0; j != ksteps ; j++){                     \
        fieldname[i][j] =0;                                \
      }                                                    \
    }

// This is the constructor. It needs a settings file to be passed to it.
FDTDCylin2D::FDTDCylin2D(string settingsFile, int argc, char const *argv[]){

  // *************************************************************
  // This section of code checks for the command line arguments
    if(argc < 2){ //Then you must run the normal script
      cout << "*************************************************" << endl;
      cout << "Running Settings File: 2D_Cylin_Settings_File.csv." << endl;
      setSimulationParameters(settingsFile);
    }
    else if(argc == 8){
      cout << "*************************************************" << endl;
      cout << "Running with cmd line options, and Settings file." << endl;
      setSimulationParameters(settingsFile);
      setCMDLineParameters(atof(argv[1]), atof(argv[2]), atof(argv[3]), atof(argv[4]), atof(argv[5]), atof(argv[6]), atof(argv[7]));
    }
    else{
      cout << "*************************************************" << endl;
      cout << "You did not enter sufficient arguments." << endl;
      cout << "The format is as follows:" << endl;
      cout << "./2DCylinFDTD rObs hObs gridSizeRadial gridSizeZ deltaD deltaZ simRunTime" << endl;
      cout << "Running Settings File: 2D_Cylin_Settings_File.csv." << endl;
      setSimulationParameters(settingsFile);
    }
  // *************************************************************

    // Character strings for file naming.
    sprintf(partATestID,"D%05gH%04g", rObs*100, hObs*100);
    sprintf(partBTestID,"Size%04g_%04g_D%05g_%05g", gridSizeRadial , gridSizeZ , deltaD*1000, deltaZ*1000);

    // Creating Output File names for the different fields. These files are opened, written
    // to and closed in the main evaluateFDTD function. Also Note that this is hard coded to
    // save files to the folder: "Simulations" located in the runtime folder.
    sprintf(EdFieldcurvename, "./Simulations/%sEdField%s.curve", partATestID, partBTestID);
    sprintf(EzFieldcurvename, "./Simulations/%sEzField%s.curve", partATestID, partBTestID);
```

```
65      sprintf(HpFieldcurvename, "./Simulations/%sHpField%s.curve", partATestID, partBTestID);
66      sprintf(ErFieldcurvename, "./Simulations/%sErField%s.curve", partATestID, partBTestID);
67      sprintf(EtFieldcurvename, "./Simulations/%sEtField%s.curve", partATestID, partBTestID);
68
69      // These are the factorial numbers for current FN.
70      factorial[0] = 1;
71      factorial[1] = 1;
72      factorial[2] = 2;
73      factorial[3] = 6;
74      factorial[4] = 24;
75      factorial[5] = 120;
76      factorial[6] = 720;
77      factorial[7] = 5040;
78      factorial[8] = 40320;
79      factorial[9] = 362880;
80      factorial[10] = 3628800;
81      factorial[11] = 39916800;
82      factorial[12] = 479001600;
83
84      // Calculation of DeltaT. This assumes a CFL condition of 1.
85      optimalDeltaT = 1/(vp*( sqrt( pow(deltaD,-2) + pow(deltaZ,-2) ) ) );
86
87      // This is the "Thumbsuck Rule" for Terespolsky Function.
88      omega0 = ( double )Na / tau1;
89
90      // For some reason the sim space goes unstable when using full CFL value. This does not
91      // currently make sense, but making dT smaller does not significantly affect the results.
92      if(useDTfromfile != 1){
93        deltaT = 0.9*optimalDeltaT;     // Uses 90% of CFL Value.
94      }
95
96      // These are the free space constants (FSC) used in my simplified example. Under complex
97      // environments with different objects, these constants would consist of 3D matrices
98      // that describe the different materials on the space.
99      EdFSC = deltaT / epsilon;
100     EzFSC = deltaT / epsilon;
101     HpFSC = deltaT / mu;
102     AxisFSC = (4*deltaT)/(epsilon);
103
104     // These are the node steps in the 3 co-ordinate system.
105     isteps = int((gridSizeRadial)/deltaD);
106     ksteps = int((gridSizeZ)/deltaZ);
107
108     // Observation point distance to node conversion. Index Based distance.
109     XOBS = rObs/deltaD;
110     ZOBS = hObs/deltaZ;
111
112     // Time Steps
113     nSteps = int(simRunTime/deltaT);
114
115     // Time to source:
116     timeToReachObservationPoint = (sqrt(pow(rObs,2) + pow(hObs,2)))/vp;
117
118     // Co-ordinate conversions. This is the angle between the Z axis and observation point.
119     theta = atan(rObs/hObs);
120
121     // EM Fields
122     // Create the 2D array pointers here.
123     create2Darray(tempHfieldPHI, isteps+1, ksteps+1);
124     create2Darray(tempEfieldD, isteps, ksteps+1);
125     create2Darray(tempEfieldZ, isteps+1, ksteps);
126     // Assign 2D pointers here.
127     HfieldPHI = tempHfieldPHI;
128     EfieldD = tempEfieldD;
129     EfieldZ = tempEfieldZ;
130
131     // Create boundary matrix pointers.
132     create2Darray(tempBoundry1, 2, ksteps+1);
```

```
133     create2Darray(tempBoundry2, 2, ksteps+1);
134     create2Darray(tempBoundry3, 2, ksteps+1);
135     create2Darray(tempBoundry4, 2, ksteps+1);
136     create2Darray(tempBoundry5, isteps+1, 2);
137     create2Darray(tempBoundry6, isteps+1, 2);
138     create2Darray(tempBoundry7, isteps+1, 2);
139     create2Darray(tempBoundry8, isteps+1, 2);
140     // Assign boundary pointers here.
141     HphiXLowerBndrTmin1 = tempBoundry1;
142     HphiXUpperBndrTmin1 = tempBoundry2;
143     HphiXLowerBndrTmin2 = tempBoundry3;
144     HphiXUpperBndrTmin2 = tempBoundry4;
145     HphiZLowerBndrTmin1 = tempBoundry5;
146     HphiZUpperBndrTmin1 = tempBoundry6;
147     HphiZLowerBndrTmin2 = tempBoundry7;
148     HphiZUpperBndrTmin2 = tempBoundry8;
149   }
150 
151 // This function is passed a file name string and extracts the required simulation
152 // variables from the file and assigns then to the relevant variables of the class.
153 void FDTDCylin2D::setSimulationParameters(string settingsFile){
154 
155     ifstream inputFromFile(settingsFile, ios::in);
156 
157     if(!inputFromFile){
158       cerr << "File could not be opened!" << endl;
159       exit(1);
160     }
161 
162     string UnitName, Value, Units;
163 
164     while(getline(inputFromFile, UnitName, ',')){
165 
166       // This next line of code removes the newline character from my settings
167       // file when it is read into the variables. This was only an issue with
168       // the UnitName variable. Required the "#include <algorithm>" library.
169       UnitName.erase(std::remove(UnitName.begin(), UnitName.end(), '\n'), UnitName.end());
170       getline(inputFromFile, Value, ',');
171       getline(inputFromFile, Units, ',');     // Variable does not get used.
172 
173       if(UnitName == "vp"){vp = std::stof( Value );}
174       else if(UnitName == "vchan"){vchan = std::stof( Value );}
175       else if(UnitName == "gridSizeRadial"){gridSizeRadial=std::stof(Value);}
176       else if(UnitName == "gridSizeZ"){gridSizeZ = std::stof( Value );}
177       else if(UnitName == "deltaD"){deltaD = std::stof( Value );}
178       else if(UnitName == "deltaZ"){deltaZ = std::stof( Value );}
179       else if(UnitName == "deltaT"){deltaT = std::stof( Value ) * pow(10,-9);}
180       else if(UnitName == "useDTfromfile"){useDTfromfile = std::stoi(Value);}
181       else if(UnitName =="simRunTime"){simRunTime=std::stof(Value)*pow(10,-9);}
182       else if(UnitName == "tau1"){tau1 = std::stof( Value) * pow(10,-6);}
183       else if(UnitName == "tau2"){tau2 = std::stof( Value) * pow(10,-6);}
184       else if(UnitName == "tau3"){tau3 = std::stof( Value) * pow(10,-6);}
185       else if(UnitName == "tau4"){tau4 = std::stof( Value) * pow(10,-6);}
186       else if(UnitName == "lamdac"){lamdac = std::stof( Value);}
187       else if(UnitName == "currentDecay"){currentDecay = std::stof( Value);}
188       else if(UnitName == "I1"){I1 = std::stof( Value);}
189       else if(UnitName == "I2"){I2 = std::stof( Value);}
190       else if(UnitName == "I3"){I3 = std::stof( Value);}
191       else if(UnitName == "eta"){eta = std::stof( Value);}
192       else if(UnitName == "Na"){Na = std::stof( Value);}
193       else if(UnitName == "rObs"){rObs = std::stof( Value);}
194       else if(UnitName == "hObs"){hObs = std::stof( Value);}
195       else{cout << "Variable "<< UnitName << " : " << Value <<" was not assigned." << endl;}
196     }
197 }
198 
199 // This function accepts the command line arguments instead of the settings file.
```

```
200 void FDTDCylin2D::setCMDLineParameters(float TrObs, float ThObs, float TgridSizeRadial, float↩
      TgridSizeZ, float TdeltaD, float TdeltaZ, float TsimRunTime){
201
202     rObs = TrObs;
203     hObs = ThObs;
204     gridSizeRadial = TgridSizeRadial;
205     gridSizeZ = TgridSizeZ;
206     deltaD = TdeltaD;
207     deltaZ = TdeltaZ;
208     simRunTime = TsimRunTime*pow(10,-9);
209 }
210
211 // This function prints the minimal Sim variables for batch script simulations.
212 void FDTDCylin2D::PrintMinimalSimParameters(){
213     cout <<"************************************************" << endl;
214     cout <<"Simulation Variables:" << endl;
215     cout <<"Requested rObs: "<< rObs <<". Actual rObs: "<< XOBS*deltaD << endl;
216     cout <<"Requested hObs: "<< hObs <<". Actual hObs: "<< ZOBS*deltaZ << endl;
217     cout <<"SizeX -SizeZ -dD -dZ -simTime" << endl;
218     cout <<gridSizeRadial << "    " <<  gridSizeZ << "    " <<  deltaD << "  " << deltaZ << "↩
           " << simRunTime*pow(10,9) << endl;
219 }
220
221 // This is the main workhorse of the Class. It will iterate through the matrix variables of
222 // the FDTD plane and calculate the EM fields at every cell at every time.
223 void FDTDCylin2D::EvaluateFDTDPlane(){
224
225     // Open the output data field files:
226       ofstream observationPointEdField(EdFieldcurvename);
227       ofstream observationPointEzField(EzFieldcurvename);
228       ofstream observationPointHpField(HpFieldcurvename);
229       ofstream observationPointErField(ErFieldcurvename);
230       ofstream observationPointEtField(EtFieldcurvename);
231     // Write the output file headers:
232       observationPointEdField << "#EfieldD" << endl;
233       observationPointEzField << "#EfieldZ" << endl;
234       observationPointHpField << "#HfieldP" << endl;
235       observationPointErField << "#EfieldR" << endl;
236       observationPointEtField << "#EfieldTheta" << endl;
237
238     cout << "************************************************" << endl;
239     cout << "Starting Simulation Time Steps: " << endl;
240
241     // This is where the time stepping starts.
242     for(int n=0; n!=nSteps; n++){
243
244       cout <<"Progress " << ((n*deltaT)/simRunTime)*100.00 << "\%  \r" ;
245       std::cout.flush();
246
247       // Magnetic Field Update Equations.
248         for(int i=0; i != isteps; i++){
249           for(int k=0; k != ksteps; k++){
250             HfieldPHI[i][k] = HfieldPHI[i][k] + HpFSC*( +1*((EfieldZ[i+1][k] - EfieldZ[i][k])↩
                  /deltaD) -1*((EfieldD[i][k+1] - EfieldD[i][k])/deltaZ) );
251           }
252         }
253
254       // Boundary Condition
255         HphiField_Mur2nRBC(HfieldPHI);
256
257       // Electric Field Update Equations.
258         // Special Case for i=0 boundary, which is the channel.
259         for(int k=0; k != ksteps; k++){
260           EfieldZ[0][k] = EfieldZ[0][k] + ((4*deltaT)/(epsilon*deltaD))*( HfieldPHI[0][k] );
261         }
262         // Ez Evaluation space
263         for(int i=1; i != isteps+1; i++){
264           for(int k=0; k != ksteps; k++){
```

```
265             EfieldZ[i][k] = EfieldZ[i][k] + EzFSC*( ( ((i+0.5)*deltaD)*HfieldPHI[i][k] - ((i↩
                    -0.5)*deltaD)*HfieldPHI[i-1][k]) / (i*deltaD*deltaD) );
266           }
267         }
268         // Ed Evaluation space
269         for(int i=0; i != isteps; i++){
270           for(int k=1; k != ksteps+1; k++){
271             EfieldD[i][k] = EfieldD[i][k] - EdFSC*( (HfieldPHI[i][k] - HfieldPHI[i][k-1])/↩
                    deltaZ );
272           }
273         }
274 
275       // Source
276         for(int k=0; k!= ksteps-1; k++){
277           EfieldZ[0][k] = EfieldZ[0][k] - (deltaT/(epsilon*pi*0.25*deltaD*deltaD))*↩
                NucciTeresCurrent((k+0.5)*deltaZ, n*deltaT);
278         }
279 
280       // Write field data to output files.
281         // The "if" check ensures that data is only written from the point when the EM field
282         // first reaches the observation point. This step also does the co-ordinate
283         // conversion for Cartesian to Spherical field components.
284           if(n*deltaT >= timeToReachObservationPoint){
285             observationPointEdField << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) <<↩
                    " " << EfieldD[XOBS][ZOBS] << endl;
286             observationPointEzField << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) <<↩
                    " " << EfieldZ[XOBS][ZOBS] << endl;
287             observationPointHpField << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) <<↩
                    " " << HfieldPHI[XOBS][ZOBS] << endl;
288             observationPointErField << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) <<↩
                    " " << +1*(cos(theta)*EfieldZ[XOBS][ZOBS] + sin(theta)*EfieldD[XOBS][ZOBS])↩
                    << endl;
289             observationPointEtField << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) <<↩
                    " " << +1*(-1*sin(theta)*EfieldZ[XOBS][ZOBS] + cos(theta)*EfieldD[XOBS][↩
                   ZOBS]) << endl;
290           }
291     }
292 
293     cout << endl;
294     cout << "Simulation Complete." << endl;
295     cout << "*************************************************" << endl;
296     cout << endl;
297 
298     // Close the data files.
299       observationPointEdField.close();
300       observationPointEzField.close();
301       observationPointHpField.close();
302       observationPointErField.close();
303       observationPointEtField.close();
304 }
305 
306 // This function takes in the Hp field and applies the Mur 2nd Order RBC. Matrix passed by
307 // reference so that it operates on the original memory. matrix, and not a copy.
308 void FDTDCylin2D::HphiField_Mur2nRBC(double** field){
309 
310     //Boundary Conditions for the x=max wall. All the way up the Y cells.
311     for(int k=1; k != ksteps; k++){
312       field[isteps][k] = -HphiXUpperBndrTmin2[0][k] - ((deltaD - vp*deltaT)/(deltaD + vp*↩
              deltaT))*(field[isteps-1][k] + HphiXUpperBndrTmin2[1][k]) + ((2*deltaD)/(deltaD + ↩
              vp*deltaT))*(HphiXUpperBndrTmin1[1][k] + HphiXUpperBndrTmin1[0][k]) + ((deltaD*vp*↩
              deltaT*vp*deltaT)/(2*deltaZ*deltaZ*(deltaD+vp*deltaT)))*(HphiXUpperBndrTmin1[1][k↩
              +1] -2*HphiXUpperBndrTmin1[1][k] + HphiXUpperBndrTmin1[1][k-1] + ↩
              HphiXUpperBndrTmin1[0][k+1] -2*HphiXUpperBndrTmin1[0][k] + HphiXUpperBndrTmin1[0][↩
              k-1] );
313     }
314     //Updating the min 2 timestep values for the next time step run.
315     for(int k=0; k != ksteps+1; k++){
316       HphiXLowerBndrTmin2[0][k] = HphiXLowerBndrTmin1[0][k];
```

```
317       HphiXLowerBndrTmin2[1][k] = HphiXLowerBndrTmin1[1][k];
318       HphiXLowerBndrTmin1[0][k] = field[0][k];
319       HphiXLowerBndrTmin1[1][k] = field[1][k];
320
321       HphiXUpperBndrTmin2[0][k] = HphiXUpperBndrTmin1[0][k];
322       HphiXUpperBndrTmin2[1][k] = HphiXUpperBndrTmin1[1][k];
323       HphiXUpperBndrTmin1[0][k] = field[isteps-1][k];   //inner 0
324       HphiXUpperBndrTmin1[1][k] = field[isteps][k];   //outer 1
325     }
326
327     for(int i=1; i != isteps; i++){
328       field[i][ksteps] = -HphiZUpperBndrTmin2[i][0] - ((deltaZ - vp*deltaT)/(deltaZ + vp*↩
              deltaT))*(field[i][ksteps-1] + HphiZUpperBndrTmin2[i][1]) + ((2*deltaZ)/(deltaZ + ↩
              vp*deltaT))*(HphiZUpperBndrTmin1[i][1] + HphiZUpperBndrTmin1[i][0]) + ((deltaZ*vp*↩
              deltaT*vp*deltaT)/(2*deltaD*deltaD*(deltaZ+vp*deltaT)))*(HphiZUpperBndrTmin1[i↩
              +1][1] -2*HphiZUpperBndrTmin1[i][1] + HphiZUpperBndrTmin1[i-1][1] + ↩
              HphiZUpperBndrTmin1[i+1][0] -2*HphiZUpperBndrTmin1[i][0] + HphiZUpperBndrTmin1[i↩
              -1][0] );
329     }
330     //Updating the min 2 timestep values for the next time step run.
331     for(int i=0; i != isteps+1; i++){
332       HphiZUpperBndrTmin2[i][0] = HphiZUpperBndrTmin1[i][0];
333       HphiZUpperBndrTmin2[i][1] = HphiZUpperBndrTmin1[i][1];
334       HphiZUpperBndrTmin1[i][0] = field[i][ksteps-1];   //inner 0
335       HphiZUpperBndrTmin1[i][1] = field[i][ksteps];   //outer 1
336     }
337
338     // This is effectively a 1st order boundary for the corners.
339     field[isteps][0] = field[isteps-1][0];
340     field[0][ksteps] = field[0][ksteps-1];
341     field[isteps][ksteps] = field[isteps-1][ksteps-1];
342   }
343
344 // This is a function that returns the current along a lightning channel.
345 // It takes in the time of interest and location on the channel.
346 double FDTDCylin2D::NucciTeresCurrent(double z, double t){
347
348     // Max Current Is: 10975.6 at 280 ns
349     // 10% of Max Current Is: 1097.56 at 192182 ns
350
351     if(t - abs( z ) / vchan  < 0){return 0;}
352     else{
353       double sum = 0;
354       for ( int i = 0; i <= Na; i++ ){
355         sum += pow( omega0 * ( t - abs( z ) / vchan  ), i ) / factorial[ i ];
356       }
357
358       double current = (exp((-1.0*abs(z)*currentDecay)/lamdac)) * ( ( I3 / eta * exp( -1.0 * ↩
              ( t - abs( z ) / vchan  ) / tau2 ) ) * ( 1.0 - exp( -1.0 * omega0 * ( t - abs( z )↩
               / vchan  ) ) * sum ) + I2 * ( exp( -1.0 * ( t - abs( z ) / vchan  ) / tau3 ) - ↩
              exp( -1.0 * ( t - abs( z ) / vchan  ) / tau4 ) ) );
359
360       return current;
361     }
362 }
```

Listing E.4: Makefile

```
CC=g++
CFLAGS=-c -g -Wall -std=c++11 -fpermissive
LFLAGS=
all:
  rm -rf build
  mkdir -p build
  $(CC) -O3 $(CFLAGS) main_Cylin2D.cpp -o ./build/main_Cylin2D.o $(LFLAGS)
  $(CC) -O3 $(CFLAGS) FDTDCylin2D.cpp -o ./build/FDTDCylin2D.o $(LFLAGS)
  $(CC) ./build/FDTDCylin2D.o ./build/main_Cylin2D.o -o 2DCylinFDTD_OPT $(LFLAGS)

clean:
  rm -rf ./build/*.o ./ViSiT/*.bov ./ViSiT/*.dat ./ViSiT/*.curve
```

Listing E.5: 2D_Cylin_Settings_File.csv

```
tau1,0.072,us,
tau2,5,us,
tau3,100,us,
tau4,6,us,
lamdac,2000,nucciconst,
currentDecay,0,trueorfalse,
I1,9900,A,
I2,7500,A,
I3,9550,A,
eta,0.845,number,
Na,3,teresnumber,
rObs,50,m,
hObs,10,m,
vp,299792458,m/s,
vchan,150000000,m/s,
gridSizeRadial,200,m ,
gridSizeZ,340,m ,
deltaD,2,m ,
deltaZ,1,m ,
deltaT,1,ns,
useDTfromfile,0,trueorfalse,
simRunTime,3170,ns,
```

Appendix F

2D Spherical Single Cell FDTD Code

Listing F.1: main_Sph2D.cpp

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;

#include "SINCELSph2D.h"

int main(int argc, char const *argv[]){

  SINCELSph2D lightningStroke2D("2D_SINCELL_Settings.csv", argc, argv);
  lightningStroke2D.PrintMinimalSimParameters();
  lightningStroke2D.EvaluateSingleCells();
}
```

Listing F.2: SINCELSph2D.h

```
#include <cmath>
#include <math.h>
#include <iostream>
#include <fstream>
#include <future>
#include <thread>
using namespace std;

#include "BfieldP.h"

class SINCELSph2D{
public:
  SINCELSph2D(string settingsFile, int argc, char const *argv[]);
  void setSimulationParameters(string settingsFile);
  void PrintMinimalSimParameters();
  void EvaluateSingleCells();

private:

  // RS Constants:
```

```
21    double vp;
22    double vchan;
23    double lamdac;
24    int currentDecay;
25
26    // Terespolsky function constants:
27    double I1, I2, I3;
28    double eta;
29    double tau1, tau2, tau3, tau4;
30    int Na;
31    double omega0;
32
33    // User Set Simulation Parameters:
34    double rObs;
35    double observationAngleDeg, theta;
36    int integrationSteps;
37    double useDTfromfile, optimalDeltaT, deltaT;
38    double simRunTime;
39    double deltaR;
40    double deltaThetaDeg, deltaTheta;
41
42    // Time variables:
43    double timeToReachObservationPoint;
44    int nSteps;
45
46    // Field value holders:
47    double BpFieldHolder1;
48    double BpFieldHolder2;
49    double BpFieldHolder3;
50    double ErFieldHolder;
51    double prevErFieldHolder;
52    double EtFieldHolder;
53    double prevEtFieldHolder;
54
55    // FSC:
56    double ErConstant;
57    double EthetaConstant;
58    double sinPlusConstant;
59    double sinMinusConstant;
60    double rPlusConstant;
61    double rMinusConstant;
62
63    // Fixed Constants:
64    const double pi = 4.0 * atan( 1.0 );
65    const double c = 299792458;
66    const double mu = 4 * pi * pow( 10.0, -7.0 );
67    const double epsilon = 8.8541878176 * pow( 10.0, -12.0 );
68
69    // Data file field names:
70    char partATestID[128];
71    char partBTestID[128];
72    char BpFieldcurvename[128];
73    char ErFieldcurvename[128];
74    char EtFieldcurvename[128];
75    char ExFieldcurvename[128];
76    char EzFieldcurvename[128];
77 };
```

Listing F.3: SINCELSph2D.cpp

```
#include <algorithm>
#include "SINCELSph2D.h"
// The Purpose of this class is to manage the calculation of magnetic fields in threads, and
// then use these fields and FDTD to calculate the electric fields.

// This is the constructor. It takes a Settings file as an input.
SINCELSph2D::SINCELSph2D(string settingsFile, int argc, char const *argv[]){

    setSimulationParameters(settingsFile);

    // Character strings for file naming.
    sprintf(partATestID,"SINCELL_R%05gA%04g", rObs*100, observationAngleDeg*100);
    sprintf(partBTestID,"dR%03gdA%02gkstep%02g_dt%03g" , deltaR, deltaThetaDeg, ↩
        integrationSteps/1000.0, deltaT*pow(10,9)*10 );

    // Creating Output File names for the different fields. These files are opened, written
    // to and closed in the main evaluate function. Also Note that this is hard coded to
    // save files to the folder: "Simulations" located in the runtime folder.
    sprintf(BpFieldcurvename, "./Simulations/%sBpField%s.curve", partATestID, partBTestID);
    sprintf(ErFieldcurvename, "./Simulations/%sErField%s.curve", partATestID, partBTestID);
    sprintf(EtFieldcurvename, "./Simulations/%sEtField%s.curve", partATestID, partBTestID);
    sprintf(ExFieldcurvename, "./Simulations/%sExField%s.curve", partATestID, partBTestID);
    sprintf(EzFieldcurvename, "./Simulations/%sEzField%s.curve", partATestID, partBTestID);

    // Degree to Radian Conversion
    theta = (observationAngleDeg/180)*pi;
    deltaTheta = (deltaThetaDeg/180)*pi;

    // Calculation of DeltaT. This assumes a CFL condition of 1.
    optimalDeltaT = 1/(vp*(sqrt( pow(deltaR,-2) + pow(deltaTheta*rObs,-2) )));

    // To use the DT from the file, set the variable to 1, else the DT will be optimal.
    if(useDTfromfile != 1){ deltaT = optimalDeltaT; }

    // This is the "Thumbsuck Rule" for Terespolski Function.
    omega0 = ( double )Na / tau1;

    // FDTD Free Space Equation Constants:
    ErConstant = deltaT/(deltaTheta*mu*epsilon*(rObs + deltaR/2.0)*sin(theta));
    EthetaConstant = (-1.0 * deltaT)/( mu*epsilon*rObs*deltaR );
    sinPlusConstant = sin(theta + deltaTheta/2.0);
    sinMinusConstant = sin(theta - deltaTheta/2.0);
    rPlusConstant = rObs + deltaR/2.0;
    rMinusConstant = rObs - deltaR/2.0;

    // Time Steps
    nSteps = int(simRunTime/deltaT);

    // Time to source:
    timeToReachObservationPoint = rObs/vp;

    // Null the E field holder variables:
    prevErFieldHolder = 0; prevEtFieldHolder = 0;
}

// This function is passed a file name string and extracts the required simulation variables
// from the file and assigns them to the relevant variables of the class.
void SINCELSph2D::setSimulationParameters(string settingsFile){

    ifstream inputFromFile(settingsFile, ios::in);

    if(!inputFromFile){
      cerr << "File could not be opened!" << endl;
      exit(1);
    }

    string UnitName, Value, Units;
```

```
67
68      while(getline(inputFromFile, UnitName, ',')){
69
70        // This next line of code removes the newline character from my settings file when it
71        // is read into the variables. This was only an issue with the UnitName variable.
72        UnitName.erase(std::remove(UnitName.begin(), UnitName.end(), '\n'), UnitName.end());
73        getline(inputFromFile, Value, ',');
74        getline(inputFromFile, Units, ',');     // This variable is not used.
75
76        if(UnitName == "vp"){ vp = std::stof( Value ); }
77        else if(UnitName == "vchan"){ vchan = std::stof( Value ); }
78        else if(UnitName == "deltaR"){ deltaR = std::stof( Value ); }
79        else if(UnitName == "deltaThetaDeg"){deltaThetaDeg = std::stof( Value );}
80        else if(UnitName == "deltaT"){deltaT = std::stof( Value ) * pow(10,-9);}
81        else if(UnitName == "useDTfromfile"){useDTfromfile = std::stoi(Value);}
82        else if(UnitName == "simRunTime"){ simRunTime = std::stof( Value) * pow(10,-9); }
83        else if(UnitName == "tau1"){ tau1 = std::stof( Value) * pow(10,-6); }
84        else if(UnitName == "tau2"){ tau2 = std::stof( Value) * pow(10,-6); }
85        else if(UnitName == "tau3"){ tau3 = std::stof( Value) * pow(10,-6); }
86        else if(UnitName == "tau4"){ tau4 = std::stof( Value) * pow(10,-6); }
87        else if(UnitName == "lamdac"){ lamdac = std::stof( Value); }
88        else if(UnitName == "currentDecay"){ currentDecay = std::stof( Value); }
89        else if(UnitName == "I1"){ I1 = std::stof( Value); }
90        else if(UnitName == "I2"){ I2 = std::stof( Value); }
91        else if(UnitName == "I3"){ I3 = std::stof( Value); }
92        else if(UnitName == "eta"){ eta = std::stof( Value); }
93        else if(UnitName == "Na"){ Na = std::stof( Value); }
94        else if(UnitName == "rObs"){ rObs = std::stof( Value); }
95        else if(UnitName=="integrationSteps"){integrationSteps=std::stoi(Value);}
96        else if(UnitName == "observationAngleDeg"){ observationAngleDeg = std::stof( Value); }
97        else{ cout << "Variable "<< UnitName << " : " << Value <<" was not assigned." << endl;}
98      }
99  }
100
101 // This function prints the minimal Sim variables for batch script simulations.
102 void SINCELSph2D::PrintMinimalSimParameters(){
103
104     cout << "*****************************************************" << endl;
105     cout << "Simulation Variables:" << endl;
106     cout << "requested rObs: " << rObs << ". Observation point angle: "  << ↩
            observationAngleDeg << endl;
107     cout << "deltaR deltaTheta intSteps -deltaT -simTime" << endl;
108     cout << deltaR << "    " << deltaThetaDeg << "    " << integrationSteps << "   " <<  ↩
            deltaT << "    " << simRunTime << endl;
109 }
110
111 // This is the main workhorse of the class. Using the parameters given by the settings file
112 // this function creates a field object for the Magnetic PHI fields surrounding the electric
113 // fields of interest. Each of these fields are then evaluated over a time period and the
114 // resulting field waveforms are written to a .curve. What is really cool about this
115 // function is that each of the fields is calculated in its own thread, which effectively
116 // means that the fields are evaluated simultaneously.
117 void SINCELSph2D::EvaluateSingleCells(){
118
119     // Create field objects:
120       BfieldP simBfieldP1(vchan, c, lamdac, I3, I2, eta, tau2, tau3, tau4, omega0, Na, ↩
              currentDecay);
121       BfieldP simBfieldP2(vchan, c, lamdac, I3, I2, eta, tau2, tau3, tau4, omega0, Na, ↩
              currentDecay);
122       BfieldP simBfieldP3(vchan, c, lamdac, I3, I2, eta, tau2, tau3, tau4, omega0, Na, ↩
              currentDecay);
123
124     // Open the output data field files:
125       ofstream observationPointBpDataFile(BpFieldcurvename);
126       ofstream observationPointErDataFile(ErFieldcurvename);
127       ofstream observationPointEtDataFile(EtFieldcurvename);
128       ofstream observationPointExDataFile(ExFieldcurvename);
129       ofstream observationPointEzDataFile(EzFieldcurvename);
```

```
130     // Write the output file headers:
131       observationPointBpDataFile << "#SinCell_BfieldP" << endl;
132       observationPointErDataFile << "#SinCell_EfieldR" << endl;
133       observationPointEtDataFile << "#SinCell_EfieldT" << endl;
134       observationPointExDataFile << "#SinCell_EfieldX" << endl;
135       observationPointEzDataFile << "#SinCell_EfieldZ" << endl;
136
137     cout << "*************************************************" << endl;
138     cout << "Starting Simulation Time Steps: " << endl;
139
140     // This is where I iterate over all the time steps:
141     for(int n=(timeToReachObservationPoint/deltaT)-10; n != nSteps; n++){
142
143       // Progress indicator:
144       cout << "Progress " << ((n*deltaT)/simRunTime)*100.00 << "\%  \r" ;
145       std::cout.flush();
146
147       // Create a thread for each magnetic field.
148       future< double > BpFieldThread1 = async(std::launch::async, &BfieldP::integrateZ, &↩
              simBfieldP1, n*deltaT + deltaT/2.0 , theta - deltaTheta/2.0, rObs + deltaR/2.0, ↩
              integrationSteps);
149       future< double > BpFieldThread2 = async(std::launch::async, &BfieldP::integrateZ, &↩
              simBfieldP2, n*deltaT + deltaT/2.0 , theta + deltaTheta/2.0, rObs + deltaR/2.0, ↩
              integrationSteps);
150       future< double > BpFieldThread3 = async(std::launch::async, &BfieldP::integrateZ, &↩
              simBfieldP3, n*deltaT + deltaT/2.0 , theta + deltaTheta/2.0, rObs - deltaR/2.0, ↩
              integrationSteps);
151
152       // Wait for threads to complete:
153       BpFieldHolder1 = BpFieldThread1.get();
154       BpFieldHolder2 = BpFieldThread2.get();
155       BpFieldHolder3 = BpFieldThread3.get();
156
157       // Electric Field Update Equations:
158       ErFieldHolder = prevErFieldHolder + ErConstant*(BpFieldHolder2*sinPlusConstant - ↩
              BpFieldHolder1*sinMinusConstant);
159       EtFieldHolder = prevEtFieldHolder + EthetaConstant*(BpFieldHolder2*rPlusConstant - ↩
              BpFieldHolder3*rMinusConstant);
160
161       // Values for next time Step:
162       prevErFieldHolder = ErFieldHolder;
163       prevEtFieldHolder = EtFieldHolder;
164
165       // Write field data to output files.
166       if(n*deltaT >= timeToReachObservationPoint){
167         observationPointBpDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) << ↩
                " " << BpFieldHolder2 << endl;
168         observationPointErDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) << ↩
                " " << ErFieldHolder << endl;
169         observationPointEtDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) << ↩
                " " << EtFieldHolder << endl;
170         observationPointExDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) << ↩
                " " << (cos(theta)*EtFieldHolder + sin(theta)*ErFieldHolder) << endl;
171         observationPointEzDataFile << (n*deltaT - timeToReachObservationPoint)*pow(10,+9) << ↩
                " " << (-1*sin(theta)*EtFieldHolder + cos(theta)*ErFieldHolder) << endl;
172       }
173     }
174
175     cout << endl;
176     cout << "Simulation Complete." << endl;
177     cout << "*************************************************" << endl;
178     cout << endl;
179
180     // Close Data file:
181     observationPointBpDataFile.close();
182     observationPointEtDataFile.close(); observationPointErDataFile.close();
183     observationPointExDataFile.close(); observationPointEzDataFile.close();
184 }
```

Listing F.4: BfieldP.h

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;

class BfieldP{
public:
  BfieldP( double Tvchan, double Tvp, double Tlamdac, double TI3, double TI2, double Teta, double Ttau2, double Ttau3, double Ttau4, double Tomega0, int TNa, int TcurrentDecay = 0 );
  double BpDifferential( double z, double t, double theta, double rObs );
  double integrateZUpper( double t, double theta, double rObs, int N );
  double integrateZLower( double t, double theta, double rObs, int N );
  double integrateZ( double t, double theta, double rObs, int N );

private:
  double vchan;
  double vp;
  double lamdac;
  double I3;
  double I2;
  double eta;
  double tau2;
  double tau3;
  double tau4;
  double omega0;
  int Na;
  int factorial[13];
  int currentDecay;

  // Fixed Constants:
  const double pi = 4.0 * atan( 1.0 );
  const double c = 299792458;
  const double mu = 4 * pi * pow( 10.0, -7.0 );
  const double epsilon = 8.8541878176 * pow( 10.0, -12.0 );
};
```

Listing F.5: BfieldP.cpp

```
#include "BfieldP.h"
// Note that this is the same class used in the FInite Antenna Method.

// This is the constructor class for the BfieldP object. The Magnetic (B) field in the Phi
// co-ordinate. WRT Cylindrical co-ordinates. This constructor takes in all the variables
// needed to later calculate the B field at instants in Time.
BfieldP::BfieldP( double Tvchan, double Tvp, double Tlamdac, double TI3, double TI2, double Teta, double Ttau2, double Ttau3, double Ttau4, double Tomega0, int TNa, int TcurrentDecay ){

  // Fixed Constants:
  factorial[0] = 1;
  factorial[1] = 1;
  factorial[2] = 2;
  factorial[3] = 6;
  factorial[4] = 24;
  factorial[5] = 120;
  factorial[6] = 720;
  factorial[7] = 5040;
  factorial[8] = 40320;
  factorial[9] = 362880;
  factorial[10] = 3628800;
  factorial[11] = 39916800;
```

```
22    factorial[12] = 479001600;
23
24    // Terespolsky Function Constants:
25    I3 = TI3;
26    I2 = TI2;
27    eta = Teta;
28    tau2 = Ttau2;
29    tau3 = Ttau3;
30    tau4 = Ttau4;
31    omega0 = Tomega0;
32    Na = TNa;
33
34    // RS constants:
35    vchan = Tvchan;
36    vp = Tvp;
37    lamdac = Tlamdac;
38    currentDecay = TcurrentDecay;
39  }
40
41  // This is the main function of the class. It calculates what I like to call the "Field
42  // Strength". This is not the correct terminology but it works for me. What this actually is,
43  // is the magnetic field differential, at a point in space and time, due to an element at
44  // height Z on the lightning channel. In simple terms it is the function that returns the
45  // value of the term inside the integral component of the equation. Later functions will use
46  // this to calculate integral value. refer to the chapter on Finite Antenna Method for the
47  // mathematics and equations. The parameters it takes are the observation point Time (t),
48  // observation point angle (theta) and observation point radial distance (rObs). It also
49  // takes the height component value of the channel (z). The returned value is used in the
50  // integration equation/function.
51  double BfieldP::BpDifferential( double z, double t, double theta, double rObs ){
52
53      double singleSumTerm = 0;
54
55      double timeShiftTerm = t -abs(z)/vchan -sqrt(pow(rObs,2.0) + pow(z,2.0) -2.0*rObs*z*cos(↩
            theta))/c;
56
57      double Rterm = sqrt(pow(rObs, 2.0) +pow(z,2.0) -2.0*rObs*z*cos(theta));
58
59      if(timeShiftTerm < 0){
60        return 0;
61      }
62      else{
63        for ( int j = 0; j <= Na; j++ ){
64          singleSumTerm += pow( omega0 * ( timeShiftTerm), j ) / factorial[ j ];
65        }
66
67        // Note that these terms have been hard coded with the Terespolsky version of the
68        // Nucci Current impulse, as discussed in the main literature.
69        double BpInduction = ( exp( (-1.0* abs(z) * currentDecay) / lamdac ) / ( 4.0 * pi * ↩
              epsilon * pow( c, 2.0 ) ) ) * ( rObs * sin( theta ) / ( pow( Rterm,3.0 ) ) ) * ( (↩
               I3 / eta * exp( -1.0 * ( timeShiftTerm) / tau2 ) ) * ( 1.0 - exp( -1.0 * omega0 *↩
               ( timeShiftTerm) ) * singleSumTerm ) + I2 * ( exp( -1.0 * ( timeShiftTerm) / tau3↩
               ) - exp( -1.0 * ( timeShiftTerm) / tau4 ) ) );
70
71        double BpRadiation = ( exp( (-1.0* abs(z) * currentDecay) / lamdac ) / ( 4.0 * pi * ↩
              epsilon * pow( c, 3.0 ) ) ) * ( rObs * sin( theta ) / ( pow( rObs, 2.0) + pow( z, ↩
              2.0 ) - 2.0 * rObs * z * cos( theta ) ) ) * ( ( I3 / eta * exp( -1.0 * ( ↩
              timeShiftTerm) / tau2 ) ) * ( exp( -1.0 * omega0 * ( timeShiftTerm) ) * pow( ↩
              omega0, Na + 1.0 ) * pow( timeShiftTerm, Na ) / factorial[ Na ] - 1.0 / tau2 * ( ↩
              1.0 - exp( -1.0 * omega0 * ( timeShiftTerm) ) * singleSumTerm ) ) + I2 * ( -1.0 / ↩
              tau3 * exp( -1.0 * ( timeShiftTerm) / tau3 ) + 1.0 / tau4 * exp( -1.0 * ( ↩
              timeShiftTerm) / tau4 ) ) );
72
73        double BfieldP = BpRadiation + BpInduction;
74        return BfieldP;
75      }
76  }
77
```

```
78 // This is the function that integrates the Bfield strength of the +ve Channel.
79 double BfieldP::integrateZUpper( double t, double theta, double rObs, int N ){
80
81   double upperZ = (rObs / (1.0 - pow( vchan / c, 2.0 ) ))*( -1.0 * pow( vchan / c, 2.0 ) * ↩
         cos( theta ) + ( vchan * t ) / rObs - vchan / c * sqrt( 1.0 - pow( vchan / c , 2.0 ) +↩
          pow( vchan * t / rObs, 2.0 ) + pow( vchan * cos( theta ) / c , 2.0 ) - 2.0 * vchan * ↩
         t * cos( theta ) / rObs ) );
82
83   // Trapezoidal Integration
84   double result = BpDifferential( 0.0, t, theta, rObs) + BpDifferential( upperZ, t, theta, ↩
         rObs );
85   double dz = ( upperZ ) / ( ( double ) N );
86
87   for ( double z = dz; z < upperZ; z+=dz ){
88     result += 2 * BpDifferential( z, t, theta, rObs);
89   }
90
91   result *= upperZ / ( 2.0 * ( double ) N );
92   return result;
93 }
94
95 // This integrates the Bfield strength of the lightning channel image (below ground).
96 double BfieldP::integrateZLower( double t, double theta, double rObs, int N ){
97
98   double lowerZ = (rObs / ( 1.0 - pow( vchan / c, 2.0 ) ))*( -1.0 * pow( vchan / c, 2.0 ) * ↩
         cos( pi - theta ) + ( vchan * t ) / rObs - vchan / c * sqrt( 1.0 - pow( vchan / c , ↩
         2.0 ) + pow( vchan * t / rObs, 2.0 ) + pow( vchan * cos( pi - theta ) / c , 2.0 ) - ↩
         2.0 * vchan * t * cos( pi - theta ) / rObs ) );
99
100  // Trapezoidal Integration
101  double result = BpDifferential( 0.0, t, theta, rObs) + BpDifferential( -1.0*lowerZ, t, ↩
         theta, rObs );
102  double dz = ( lowerZ ) / ( ( double ) N );
103
104  for ( double z = dz; z < lowerZ; z+=dz ){
105    result += 2 * BpDifferential( -1.0*z, t, theta, rObs);
106  }
107
108  result *= lowerZ / ( 2.0 * ( double ) N );
109  return result;
110 }
111
112 // This function performs an integral over the +ve and -ve lightning channels. It performs
113 // this integral at a specific time points over the entire applicable lightning channel. The
114 // applicable lightning channel length is dependant on the time taken for current to travel
115 // in the channel. The returned result is the field value at the time instant. The input
116 // parameters are the time instant of interest (t), observation point angle (theta),
117 // observation point radial distance (rObs), and N is the  amount of integration steps in
118 // each integral stage, regardless of channel length.
119 double BfieldP::integrateZ( double t, double theta, double rObs, int N ){
120   return integrateZUpper(t,theta,rObs,N) + integrateZLower(t,theta,rObs,N);
121 }
```

Listing F.6: Makefile

```
CC=g++
CFLAGS=-c -Wall -std=c++11 -fpermissive -pthread

all:
  $(CC) $(CFLAGS) SINCELSph2D.cpp -o ./build/SINCELSph2D.o
  $(CC) $(CFLAGS) BfieldP.cpp -o ./build/BfieldP.o
  $(CC) $(CFLAGS) main_Sph2D.cpp -o ./build/main_Sph2D.o
  $(CC) -pthread -std=c++11 ./build/SINCELSph2D.o ./build/BfieldP.o ./build/main_Sph2D.o -o ↩
       SingleCellFDTD

clean:
  rm ./build/*.o SingleCellFDTD
```

Listing F.7: 2D_SINCELL_Settings.csv

```
tau1,0.072,us,
tau2,5,us,
tau3,100,us,
tau4,6,us,
lamdac,2000,nucciconst,
currentDecay,0,trueorfalse,
I1,9900,A,
I2,7500,A,
I3,9550,A,
eta,0.845,number,
Na,3,teresnumber,
vp,299792458,m/s,
vchan,150000000,m/s,
rObs,50,m,
observationAngleDeg,90,deg,
integrationSteps,1000,int,
deltaR,10,m,
deltaThetaDeg,10,deg,
integrationSteps,2000,count,
useDTfromfile,0,trueorfalse,
deltaT,0.05,ns,
simRunTime,5167,ns,
```