Design of a Universal Antenna-Based Problem Simulation Interface with Emphasis on Geometric Data Capturing and Handling

Timothy M. Rokebrand

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.

Johannesburg, September 2020

Declaration

I declare that this dissertation is my own, unaided work, except where otherwise acknowledged. It is being submitted for 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.

Signed this ____ day of _____ 20__

_____

Timothy M. Rokebrand

Abstract

The design of a simulation framework that allows for interaction with multiple computational engines through a single interface is presented. The framework was designed, not only to combine these engines under a single interface but also to simplify the design process, this was achieved by making use of macro-based design. These macros are used to build antenna arrays by automatically defining the dimensions of base elements that make up the array using the simulation frequency as reference. For more advanced users, there is the option to make use of scripting, which allows for more optimisation driven design to take place. The framework takes the user inputs from either the graphical or scripting interfaces and, using the macros, creates input files formatted to pass into the simulation engines. The engine then performs the calculations and produces an output file that is then used to provide requested outputs. An implementation of the framework that is able to run simulations in both NEC++ and FEKO is provided. Testing entailed running simulations with dipole, Yagi-Uda and rectangular patch geometries. The results showed that the program was able to successfully capture data from the user, convert these user inputs into simulation files that could then be executed to complete simulations effectively. Results from the different computational engines were similar and these aligned with literature. This is evidence that the framework is successful in its objective of providing a universal simulation platform using multiple engines, while the geometric information is captured once.

Acknowledgements

I would like to thank Professor Alan Clark for, not only his guidance and endless support as a supervisor, but for the knowledge he shared through the years as a lecturer and for inspiring me to pursue my Masters in this field. I would also like to thank Mrs. Ingrid Menzel and her late husband for the generous donation made to assist in completing my studies, my appreciation for this gesture is boundless. Finally, my friends and family, for standing by my side giving unwaivering support from beginning to end, it would not have been possible without them.

Contents

List of Figures

List of Tables

Nomenclature

GUI Graphical User Interface

VSWR Voltage Standing Wave Ratio

MoM Method of Moments

FEM Finite Element Method

NEC Numerical Electromagnetics Code

FEKO Feldberechnung für Körper mit Beliebiger Oberfläche ("Field computations involving bodies of arbitrary shape")

Chapter 1

Introduction

In the world of Computational Electromagnetics there are a large number of simulation packages available, each with their own shortfalls. This work provides a framework for an antenna simulation program that allows a user, through a single interface, to interact with these various simulation packages; thus overcoming the shortcomings of the different computational engines. The main focus of the framework was the handling of geometric data, with the aim of simplifying the process of capturing data and retrieving the relevant electromagnetic characteristics of the proposed design.

Since the advent of Computational Electromagnetics, the understanding, analysis and design of efficient antennas has been greatly enhanced as well as the understanding of electromagnetic behaviour in general. As technology advances, so does our ability to create more detailed and accurate simulations. This is due to the development of various electromagnetic solvers. In their most basic form, these solvers simply calculate electric and magnetic fields, using Maxwell's equations as a basis, within one, two or three dimensional simulation space. These programs greatly speed up the tedious process of iterative calculations within a simulation space and they have been greatly enhanced with improvements in computing speeds especially within the realm of parallel computing [2].

These solvers are used to solve a multitude of different problems in the fields of high voltage engineering, power engineering, electromechanical engineering and communication to name a few. This research is focused on improving aspects within the programs to streamline the design and analysis of antennas by integrating existing programs and their differing simulation methods.

This project aimed to initiate the development of an electromagnetic solver and antenna simulation software framework that will focus on the manner in which geometric information is handled. It is to be developed as a completely open source package. The program must be user friendly for both expert users as well as amateur users. It must be capable of being used as a viable teaching tool for those learning about electromagnetics and antennas. This will therefore mean that minimum input should be required in order to generate an effective design. However, a more advanced user should have more freedom to adjust any parameter in the design.

There are a number of existing packages that effectively simulate specific types of structures (wire, volumetric, planar) or problems and use particular mathematical techniques in order provide accurate solutions for those types of structures. This means that there is often a limitation on the type of problem that can be accurately solved by a particular program. The proposed solution to overcome this problem of lack of diversity in terms of problem types, is to develop a framework that would allow for a single interface to be used to access any of these methods best suited to simulate an antenna design and pass the necessary information to existing engines. A challenge arises in defining the proposed antenna design in such a way that it is possible for multiple simulation techniques to be used as well as allowing for easy manipulation of antenna parameters for optimisation purposes.

There are programs that allow the user to make use of scripting, which makes it easier to run through multiple iterations of a particular design [3]. This process is

not always intuitive but is seen as a powerful tool in the design and optimisation processes. It will also provide an invaluable tool in teaching in that iterations of a particular problem can be simulated and visualisations can be provided to clearly show the effects of altering particular parameters [4]. It is therefore seen as important that the framework caters for scripting capabilities.

In order to test the efficacy of the proposed framework the scope of the research includes a basic implementation of a program designed around the framework. Being an initial and basic implementation only skeletal type antennas such as Helical, Yagi-Uda, Dipole/Monopole antennas and simple Patch Antennas are included in the implementation and testing scope.

1.1 Research Outcomes

- Identify the challenges that are currently faced and explain how the proposed framework will address these challenges and issues;
- The design of a framework for a universal geometry handler that allows for the use of multiple simulation methods and computational engines;
- The development of an open source and basic implementation of a program based on the proposed framework that allows for testing and acts as a proof of concept; and
- Show the framework's ability to seamlessly integrate multiple simulation engines using different mathematical methods;

1.2 Dissertation Overview

Chapter 2

This chapter provides background to the field of Computational Electromagnetics and will, in particular, focus on the application of antenna simulation. It will provide information on topics pertinent to the research and begin to contextualise the purpose of the research.

Chapter 3

This chapter identifies the problems that the research aims to overcome. It highlights existing solutions and provides the methodologies and approaches taken in order to achieve the defined outcomes.

Chapter 4

This chapter details the design of the framework and presents an implementation as a proof of concept. Testing of this implementation was conducted, the results of which are also provided in this chapter.

Chapter 5

This chapter provides conclusions of the findings and details future steps to follow this research in terms of further improving the implentation of the framework.

Chapter 2

Antenna Simulation

2.1 Computational Electromagnetics

Computational Electromagnetics refers to the use and study of computational methods in order to solve electromagnetic problems. It involves the solving of Maxwell's equations (shown in Section A) in order to quantify and gain understanding of the electromagnetic field behaviour for a wide range of applications [5]. The simulation of antennas is of particular interest for this research. This is necessary in order to gauge potential performance characteristics and improve physical designs through the use of optimisation to generate the best theoretical design for a particular purpose.

The manner in which these types of problems are approached may vary as there are many methods and approaches available, but the working principle behind each method is similar. In general the objects or space under investigation are bounded within a singular or multi dimensional simulation space. Due to the fact that computers are to be used to perform the calculations the simulation space needs to be bounded and discretised. This means that the entire space is divided into a set number of cells. This discretisation is performed differently between simulation

methods but once the cells are created, mathematical operations are performed to determine electromagnetic behaviour and field characteristics for each cell [2].

Maxwell's equations form the the basis of most calculations in electromagnetics. They can be represented in differential or integral forms, the use of either depends on the parameters being calculated, the type of simulation and the simulation method being employed. The differential form allows for the solving of the equations and thus the characterisation of field behaviour at specific points in space. The calculated values at these points are then related to each other in order to develop a full picture. The integral form allows for the field characteristics to be determined throughout a region and this in turn relates to characteristics along contours, through or over surfaces and volumetric objects [6]. Both the integral and differential forms of the four equations are given and explained in the next section.

These equations are then used in conjunction with each other to develop a full picture of the electromagnetic parameters of objects under investigation. In computational simulations—when using the integral forms of the equations—the size of the integral steps (dS for example) is determined by the manner in which the simulation space or object under investigation is segmented. Thus, increasing the number of cells that the simulation space is broken up into results in a higher accuracy because each integral operation can be applied to a smaller area and there is more detail given when the overall result is calculated. This also leads to larger numbers of integrals that need to be solved and the reason why computational electromagnetics is important. While these integrals can be solved manually, the shear number makes it impractical to do without computational aids [7].

2.2 Simulation Methods

Depending on various factors, there are a number of simulation methods that can be used. These methods cover both the frequency and time domain. The various methods have their own strengths and drawbacks and are used depending on a number of factors including the electrical size of the simulated objects, the frequencies being used and the scale of the simulation in terms of computational power and memory requirements[8].

This section includes information about the methods that are of particular interest within the scope of the research. The methods that will be discussed in depth are Method of Moments and the Finite Element Method. The methodologies have a wide range of applications and are not only limited to the field of electromagnetics. They can be seen as computational methods that can be used to solve a large number of repeating equations for a particular simulation space. These equations are dependent on the nature of the problem under investigation and the required properties. In the case of antenna simulations, these equations usually take the form of integral or differential equations derived from Maxwell's equations (discussed in Section 2.1). These equations in isolation are used to determine a number of properties such as current and charge distributions, impedances, gain characteristics and radiation patterns to name a few [9]. As previously discussed these calculations are to be carried out over every cell within the simulation space and these methods allow for efficient computation of the results.

2.2.1 Method of Moments (MoM)

Method of Moments is a frequency domain methodology that is widely used in antenna simulations. The discretisation in this method is achieved by breaking

only the structure under investigation into segments, not the entire simulation space. To calculate the electromagnetic properties, integral equations are used to develop information that can be used used to calculate electric and magnetic field characteristics. Calculating the electromagnetic properties and how that point is affected electromagnetically in relation to the position of a source defined by the user and ensuring that interfaces between segments are correctly accounted for to avoid inconsistencies. In antenna simulations, these segments are generally cylindrical. This makes this method particularly useful for wire type antennas and skeletal structures.

The general approach to the method of moments is to define an unknown linear function that will fit to a particular characteristic under investigation (for example, charge distribution along a wire). This is achieved by equating a known function to the integral form of the unknown function. The integral allowing for the operation to be conducted through a region or along a contour or surface. The unknown function is then represented by the sum of N basis functions each of which is multiplied by a weighting coefficient, where N relates to the number of segments that need to be solved for [9, 10].

A basis function refers to a known function that can be used to approximate the value of the unknown function for a particular region. These basis functions can be as simple as a pulse function over a single segment so the the value remains constant over a segment; or a combination of sine functions spread over multiple which when added together starts developing a picture of the actual solution. The coefficient values are responsible for setting the magnitude of their respective basis function. Thus it is important to select an appropriate basis and accurately calculate the coefficients to allow for an accurate model to be determined [9, 10].

An important part of using basis functions is setting up appropriate boundary conditions to allow for correct handling of intersections between segments and

termination points of structures. With the boundary conditions set, it is possible to solve the integral portion of the equation. The integrals are solved at each segment, this in turn leaves N unknowns in the form of the coefficients. However in order to solve for N unknowns, N separate equations are required. To generate the N equations required, the integral is solved for all segments but keeping the reference point—a point at the centre of a segment—constant. Once the integral is solved for all segments, the reference point is shifted one segment and the process repeats, thus leaving N unknown coefficients acting on N different equations. These can be written as an N×N matrix. Finally, the product of the integral and coefficient can be equated to a known value, and it is then possible to solve for the values of the coefficients and therefore find values for the original unknown function [9, 10].

The accuracy of the simulation depends on the number of segments used. Too few segments will not provide enough information for reliable results to be provided. Too many segments will result in unnecessary computing expenses and results may also be unreliable due to numerical errors. These errors could come about if the segments are so small that rounding errors occur when calculating the matrix values. These errors would be worsened during the solving of the matrix equations. A general rule is that segment length should be significantly less than the wavelength of the simulated frequency with at least ten segments per wavelength distance [9].

A powerful idea in this methodology is that of “thin-wire approximations”. This approximation can be made for wire type (cylindrical) elements when the radius of a wire is much smaller than both the length of the element and the wavelength of of the investigated. The approximation states that current will only be present on the surface of the wire, the currents will only be axial in nature (i.e. they will travel along the length of the wire) and that current is uniformly distributed across the surface. This means any currents within the wire and cross or circumferential currents are ignored. This greatly reduces complexity of the calculation process

and therefore simplifies the entire simulation process. It is important to note that applying this approximation does not ignore the radius of the wire. The radius of the wire is a necessary parameter in calculating the electromagnetic fields; without it a surface area does not exist and parameters such as charge distribution cannot be calculated accurately and Maxwell's equations also cannot be solved to give useful information [9, 11].

One major advantage of this method is that only the structure being investigated needs to be discretised, unlike other methods that require the discretisation of the entire simulation space before calculations can be carried out. This therefore means that calculations performed on electrically large structures (i.e. those that are large in relation to the wavelength of the frequency at which they are excited) are often more efficient than other methods in terms of computing requirements. It also means that because there is no simulation boundary, properties related to larger distances such far-field radiation patterns can be calculated with relative ease. These advantages combined with the thin-wire approximation make this method ideal for simulating wire type structures.

Another disadvantage of this method is, due to the manner in which the matrices are created (by solving an integral for each entry). This means that every entry has a value and this means that dense matrices are created [9]. This makes the process of solving the matrix equations computationally intensive when compared to other methods that create sparse matrices with many zero values. Another drawback to MoM simulations is that they do not inherently take dielectric and magnetic properties into account. This is overcome in some programs, such as FEKO, through the addition of certain extensions to the MoM simulation code [8].

2.2.2 Finite Element Method (FEM)

The Finite Element method presents a different approach to solving electromagnetic problems. In this frequency domain method, like in MoM, the structure is simulated by breaking the the structure into finite cells then calculations are carried out for each cell and interfaces between cells are considered, using different boundary conditions, to ensure smooth gradients between cells. The accuracy of simulations is also dependent on the scaling of the cells in relation to the simulation frequency.

This method differs in that these cells are not cylindrical segments, instead they are polygon type shapes, most commonly triangular or, in the case of 3-D simulations, tetrahedral shapes. These triangles are set up in a continuous mesh covering the structure being simulated as well as a predefined area around the structure. This entire area, including the structure, forms the simulation space and calculations must be carried out for every cell within the simulation space.

A disadvantage of this method is that, because the entire simulation space is considered, it is generally more computationally intensive than the MoM approach where only the structure is segmented. This is apparent in electrically large simulations where there would be a need for a much larger number of individual cells, thus limiting the effectiveness of this method to smaller scale problems. This is partly overcome by using different mesh sizing depending on the nature of specific points within the simulation space. Larger segments can be used on smooth surfaces or in relatively large spaces such air gaps where changes in electromagnetic field properties would be gradual from segment to segment. Smaller segments could be used around corners or sharp points and at interfaces between different materials where rapid changes in field behaviour may occur over much shorter distances.

This method has the advantage that it is able to accurately and effectively carry

out calculations for volumetric surfaces. It also allows for and requires the defining of physical properties that would affect electromagnetic responses thus making it effective in dealing with simulation spaces with materials of varying dielectric and magnetic properties.

2.3 Simulation Programs

There are a wide variety of simulation packages available from basic free-to-download packages to high-cost, computationally powerful products. These programmes make use of the aforementioned simulation methodologies in order to present the user with useful information about the performance characteristics of antennas and aid with effective design. This is achieved using either custom built or existing simulation engines in order to perform the necessary calculations. This section discusses some of these programs and the engines that they use. Particular attention will be paid to the Numerical Electromagnetics Code (NEC) simulation engine and FEKO program as these will form the focus of the discussed implementation.

2.3.1 Numerical Electromagnetics Code (NEC)

This code was originally developed in the 1970s [12]. It is a MoM solver that focuses on the simulation of wire-type structures. NEC-2, originally written in FORTRAN, was introduced in 1981 and added more functionality. Later iterations (NEC-3 and NEC-4) added further functionality around buried elements and modelling of smaller antennas. NEC-2, however, remains as the latest open source version and is therefore widely used across electromagnetic solvers. The code of the engine itself has been translated into various languages, such as nec2c which is a version written in C and nec2++ which was written in C++, thus making it more compatible with modern

coding approaches.

What has remained constant throughout the the development is the "punch card" approach to its operation. In other words, the engine will accept a non-encrypted text based input file of a particular format (".nec"). The file contains all the information about geometry, frequency, excitation voltages and ground considerations. The engine uses the information contained within this input file to carry out the simulation, once completed the results are written to an output file that can be used to compile the information into useful formats (graphs, 2D and 3D plots etc.).

A number of programs have been based on and use the NEC engine as the basis of their simulation packages, and then have their own means to interpret and present the information contained in input and output files. These products include free packages such as 4NEC-2, EZNEC (limited functionality in free version), xnec2c which provided a graphical interface for the nec2c engine and NEC Lab [13, 14]. There have also been a number of programs that do not make use of the engine itself but instead simply interpret NEC input and output files and are then able to use the information to generate plots or give visualisations of the antenna itself. An example of such a program is xnecview [15].

While NEC does make provision for patch antennas and volumetric surfaces, as a limitation of the MoM approach it uses, these surfaces are represented using wire meshes. This therefore means that certain assumptions have to be made which could affect the accuracy of simulations. As only wire type structures are considered, the effects of different dielectric materials cannot be simulated as this would involve consideration of the properties of the space between elements and while the method of moments approach can be adapted to allow for a certain degree of consideration of these parameters, NEC does not allow for this parametrisation. It would therefore be beneficial to make use of a different simulation approach that is designed to consider

these properties and is able to handle volumetric surfaces with fewer assumptions.

2.3.2 FEKO

FEKO is a proprietary product owned by Altair and is a powerful electromagnetic solver and simulation program. This program makes use of its own engine and is able to execute various types of simulations using a number of different simulation methods.

The base computation approach used by FEKO is also MoM, which will be useful for basic skeletal and wire based structures. But as was discussed in Section 2.2.1 the Method of Moments approach does not consider dielectric properties and also cannot handle volumetric surfaces without breaking them down into wire mesh structures that approximate the actual surface. FEKO however, has extensions to the normal MoM approach, giving it the ability to represent dielectric properties. This additional functionality is limited, and FEKO is able to use a different simulation method, depending on the scale of the simulation (particularly in terms of electrical size) and the types of structures and dielectric materials contained within the simulation space [8]. These alternative methods are FEM, Multilevel Fast Multipole Method (MLFMM), Physical Optics, Geometric Optics and Universal Theory of Diffraction (UTD); with each method being effective with larger and larger electrical scale problems [8]. The scope of this work is on relatively small scale problems so the main focus will be on the MoM and FEM approaches.

As with NEC, FEKO makes use of a “punch card” type approach, in that, it creates input files (“.pre” files) of a specific format that is in turn interpreted by the FEKO engine in order to calculate the required outputs. Once calculated the results are written to a binary file unique to FEKO as well as a plain text file (“.out”). FEKO has a number of interfaces for both creating inputs and setting up

simulation parameters(CADFEKO and EDITFEKO) and for viewing and interacting with various outputs (POSTFEKO). CADFEKO allows the user to create and edit antenna geometries and provides a graphical representation of the design as changes are made once completed the geometry and simulation data is written to a .pre file. EDITFEKO is an interface through which the user can create and edit .pre files, providing macros in order to easily generate the necessary input commands for parameters such as geometry types and dimensions, frequency ranges, excitation considerations and output plot characteristics to name a few.

2.3.3 Incompatible Programs

There are a number of other popular existing programs such as HFSS, CST and WipID that perform electromagnetic simulations, however they will be incompatible with the framework as described in Section 4. This is because these programs read and compile input and output files that are encrypted and unique to the particular program. Due to this, one will not be able to create plain text files as with NEC and FEKO simulations using a program external to the calculation engine.

2.4 Physical Geometry Considerations

As mentioned above, there are a number of considerations to be made in terms of the geometries and properties of different antenna types. The main reason for different types of antennas is to obtain different gain characteristics depending on the application. If we consider a single charge in an isolated system from figure A.1 we can see that the electric field is distributed evenly in all directions, this makes it an isotropic source. In the same way a theoretical antenna that, when excited, has equal gain in all directions is known as an isotropic antenna. The gain of antennas

is often measured with the unit dBi (decibels in relation to an isotropic antenna). Thus if the radiation pattern of an isotropic antenna is represented as a surface, the shape would be a sphere as the gain is equal in all directions.

Different antennas would have radiation patterns that distort the shape of this sphere which allows for higher gain to be achieved in certain directions. It is important to note that as you increase gain in a certain direction it will reduce in another because of conservation of energy laws.

The following subsections present a high level introduction to the geometric considerations of the antennas that fall within the scope of this research. These are dipoles and monopoles, Yagi-Uda, helical and patch antennas. It highlights the range of parameters relating to the antennas. Also, basic forms of the antennas will be explored. These parameters need to be well defined and incorporated into the code so that they are easily accessible and adjustable.

2.4.1 Monopole and Dipole Antennas

Certainly the most simplistic of the antennas under investigation, they are also some of the most commonly used antennas. In its basic form a dipole is simply a piece of wire excited by a source of some description most commonly sized to be a half wavelength long. It is essentially one element that is split where the source is connected (often in the centre) where one side is fed from the live of the feed and the other by the return, hence the name dipole. It is an omnidirectional antenna which means that its radiation pattern is such that the maximum gain of the antenna is in multiple directions. The maximum gain of an ideal half-wavelength antenna is approximately 2.15dBi and the radiation pattern is a toroidal shape around the element [1, p.200].

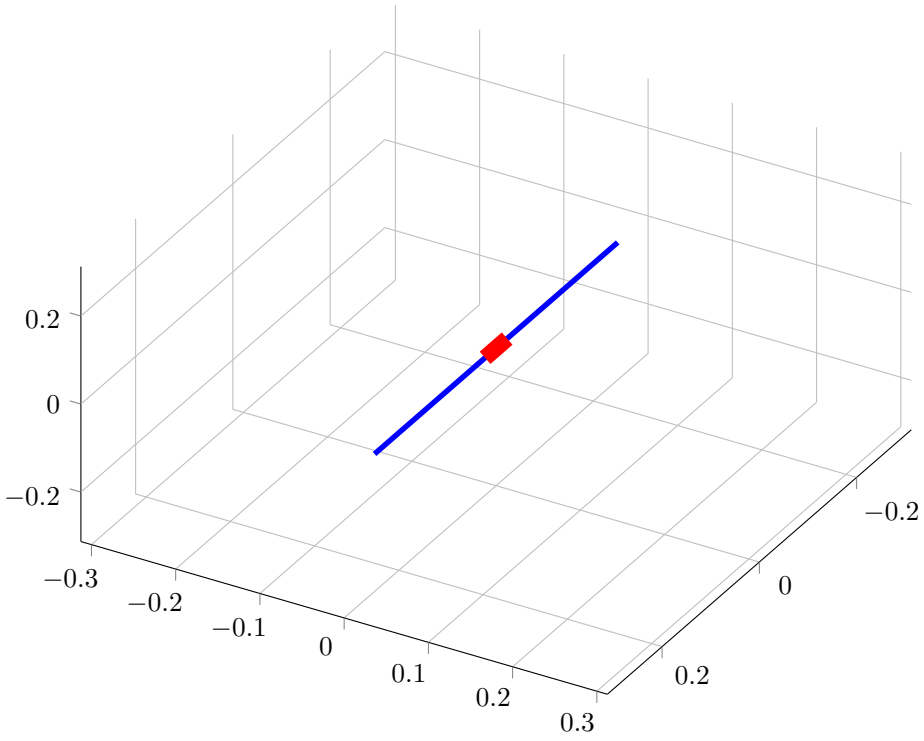

Figure 2.1: Centre-fed dipole antenna

A monopole can be seen as one half of a dipole, it will be excited by only the live with the return connected to ground. Like the dipole, it is an omnidirectional antenna but it has a maximum gain and is most commonly designed to be a quarter wavelength long.

The geometric considerations for both the monopole and dipole are as follows [1, p.200]:

- Length of element
- Location of the feed along the length of the element
- Radius of the element

2.4.2 Yagi-Uda Antennas

The Yagi-Uda antenna is a widely used high-gain or directional antenna. In its basic form it is highly directive meaning that the maximum gain of the antenna is high in a single direction and the gain is almost negligible in other directions. It is designed as a frequency specific antenna and thus has a narrow frequency bandwidth. The basic design involves a single driven element — most often a dipole — behind a number of directors that shape the main radiation beam in a particular direction. Then a reflector element is placed behind the driven element in order to limit the amount the amount of backward radiation [1, p.481-485]. Geometric properties to consider are:

- Total number of elements
- Length of driven, director and reflector elements
- Boom length of the antenna i.e. total length
- Location of the feed along the length of the driven element
- Radius of the elements
- Spacing between the driven element and the directors

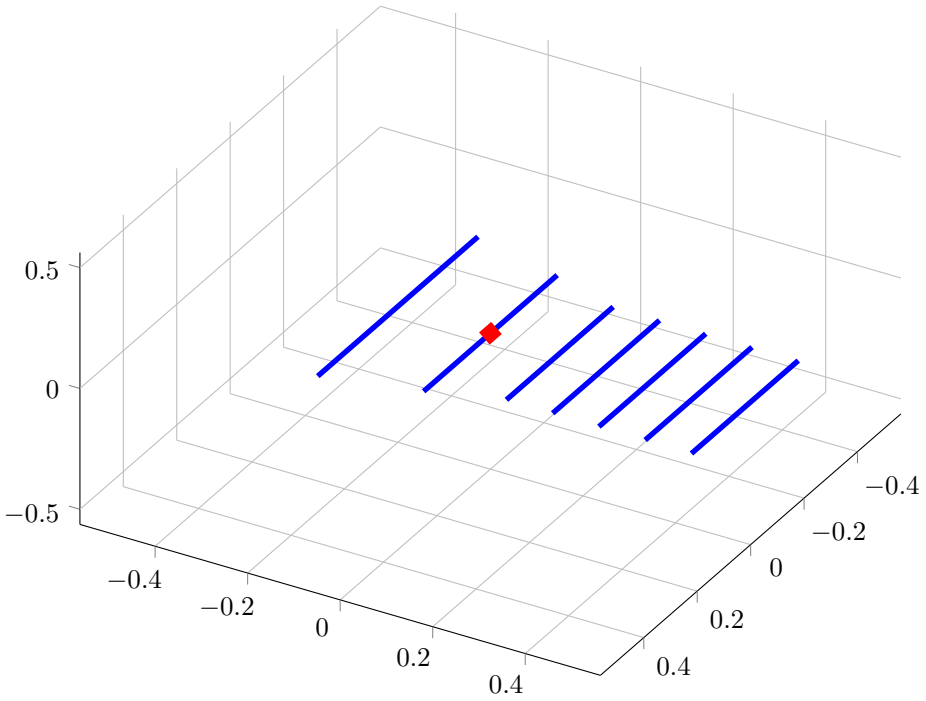

Figure 2.2: Seven-element Yagi-Uda antenna made up of (from left to right) a single reflector, a single driven element and five directors

- Spacing between the driven element and the reflector
- Spacing between the directors

2.4.3 Helical Antennas

The Helical antenna is a high-gain antenna designed by John D. Kraus. Like the Yagi-Uda antenna, they are widely used and have a small frequency bandwidth [1, p.271-274].

- Number of turns in helix
- Circumference of each turn
- Radius of the element
- Pitch angle of the helix

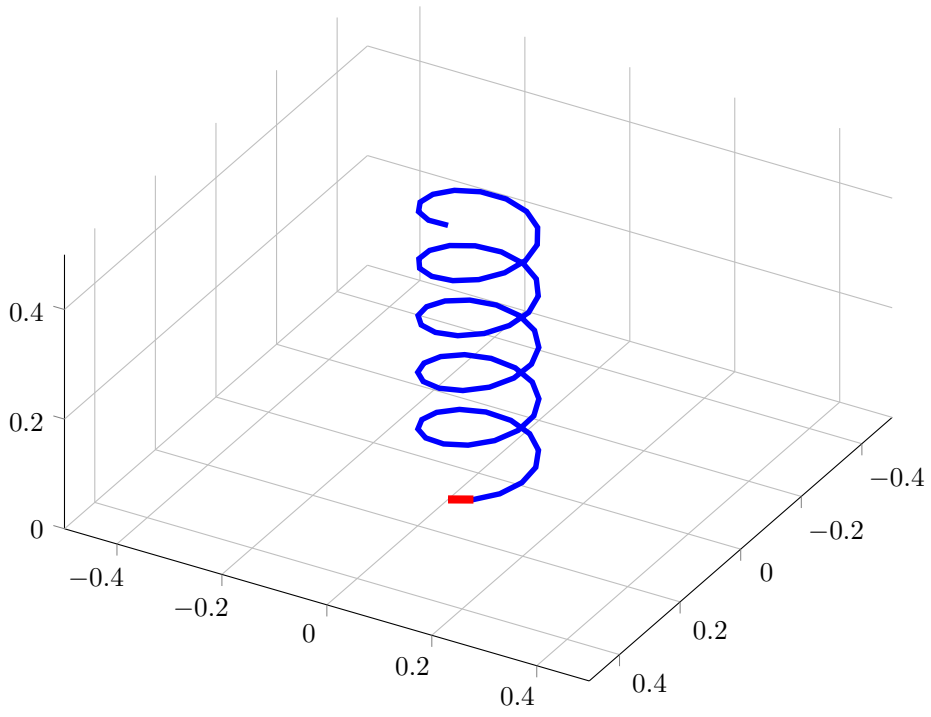

Figure 2.3: End-fed helical antenna

- Position of the voltage feed

2.4.4 Patch Antennas

Patch antennas are widely used in electronic devices as they are cheap to manufacture and their design makes them inherently compact.

In its basic form a patch antenna consists of the patch itself and a ground plane separated by a substrate or dielectric material, see Figure 2.4. The patch on top is excited by a source and this establishes an electric field between the patch and the ground plane. The patch is usually designed to have a legth equivalent to half a wavelength as this is a length at which the antenna will resonate and perform most effectively. The electric fields are strongest at the edge edge of the patch where fringing occurs, which results in radiation. The gain of a patch antenna of this type is approximately 6dBi [16].

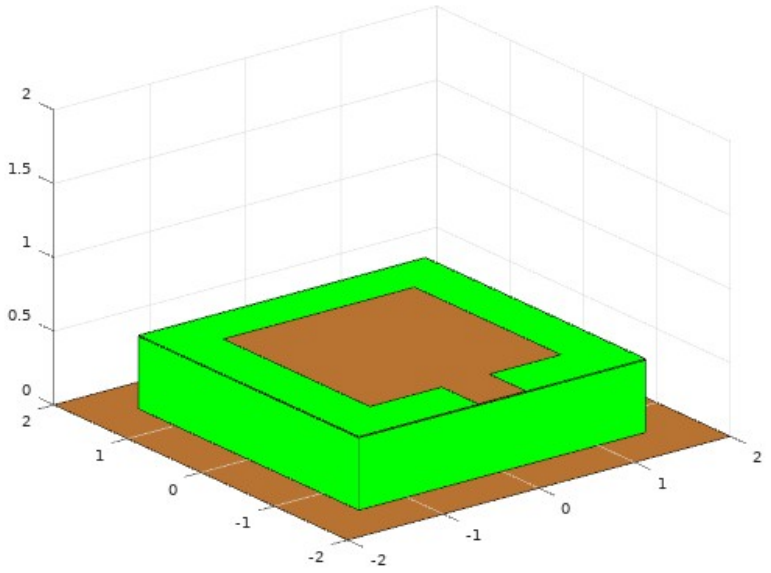

Figure 2.4: Basic centre coaxial-fed patch antenna; substrate shown in green between the active patch element and ground plane

The fact that the patch antenna contains a substrate with dielectric characteristics means that it is not directly compatible with a NEC simulation approach. Work presented within literature [17, 18] defined methods and examples of simulating patch antennas within SuperNEC with sufficient accuracy. Therefore, these approaches will be considered.

Geometric parameters and physical properties to be considered are as follows [19]:

- Length and width of the patch (if rectangular) or diameter (if circular)
- Relative permittivity (ε_r) of the substrate
- Thickness of the substrate
- Position of the feed point

2.5 User Experience Considerations

While user experience is not a direct parameter of investigation within the scope of this research, it is an important consideration in the development of the interfaces,as discussed in Chapter 3.

Antenna simulation software packages have the potential to be valuable teaching aids to assist students, particularly at undergraduate level, obtain a clear understanding of many concepts. This point has been explored through research conducted by Baltzis et al. [4] where the impact of using free electromagnetic simulation software (in the case of the research the use of FEMM was explored) on the performance of students and their understanding of concepts. The work concluded that there appeared to be improvements in the academic performance of students who were actively making use of the software. The sample sizes used in the research were relatively small but the results indicated clear improvement. There are currently several free-ware antenna simulation packages but they fall short in terms of usability, especially for amateur users. It is therefore important not only to make such programs accessible to students, but to give the best user experience possible.

User experience is an important consideration in the development of this framework, in order to define expectations in terms of user experience within the context of the research, it must be understood that there will be different types of users, each with varying requirements depending on their expertise and expectations from such a program. For the purposes of this research the different users and expected user experience for each type of user are defined as follows:

- **Amateur User**—An amateur user refers to a first time user or one with little knowledge of antenna theory and require an antenna design for studying or practical purposes. The expected user experience in this context relates to

making the GUI easy to understand and ensuring that there is logical flow within the interface in terms of acquiring inputs from the user. Another more important aspect is that it requires that the user can give minimal input and still receive a useful antenna design.

- **Advanced User**—An advanced user refers to someone that has knowledge of antennas and may want to create there own geometries and optimise design parameters. They must be able to easily create geometries and iterate through various adjustments to the geometry. This is to be achievable through either interface (GUI or by scripting algorithms). This process will be facilitated through the use of functions that will allow the user to easily add and edit geometries (discussed further in Section 4.2) to the simulation space making the implementation of algorithms as convenient as possible.

- **Developer or Researcher**—Future development and expansion of the framework is an important consideration of the research. In order to ensure the survival of the framework, the process of adding components and functionality must be made simple and clear. This will involve having set procedures and limitations on languages that can be used but also ensure there are clear guidelines to add components and integrate them with the existing framework. Researchers using the framework in order to test, for example, new geometries, a new optimisation algorithm etc. must be able to carry out these tests with minimal to no adjustments to the framework. If the researcher needs to add particular functionality to the framework, they then fall into the category of developer and the outlines provided above cover the expectations needed from the framework.

Chapter 3

Universal Geometry Handling

The following chapter identifies problems that arise from current solutions in the field of antenna simulation as discussed in Chapter 2. It also discusses current solutions available and finally presents a universal geometry handling interface as a proposed solution to the problem.

3.1 Program Limitations

As discussed in Chapter 2 there are a number of different simulation engines and programs available to users. These programs use a number of different methodologies in order to perform the electromagnetic calculations.

Many of these programs are limited in the problems that they are able to solve efficiently or have been developed for a particular type of solution. These limitations could be due to the type of simulation method that is used; for example, NEC uses MoM and is unable to handle dielectric properties [12], or simply due to incompatible design of the software. The application of certain methods depends on characteristics such as the type of structure, the electrical size of a simulation and the use of varying materials.

There is no package currently that would allow a user to pass a single geometry or design to multiple simulation engines. This is due to the fact that these programs differ in the manner in which they interpret the geometry. It would therefore be useful to handle the geometric data in such a way that it can easily be passed to any of these simulation engines for calculations. These simulation engines present the raw data of their calculations in different manners; this means that a program that can interpret these results universally and then present them to the user will be of use as it will enhance convenience and user experience. Standardising the format in which the geometric information is contained, i.e. creating a universal antenna geometry language to facilitate easier interpretation by any of these simulation engines will therefore be investigated.

3.2 User Experience

Many of the packages mentioned in Section 2.3 are not entirely intuitive and are often intimidating to inexperienced users. The issue arising from this is that if the user comes across a problem that the simulation engine they are using cannot handle effectively, they are then required to find a program that would allow them to make use of a different calculation method. This in turn involves learning how to use yet another interface. There exists a need to simplify the user experience of such packages, without sacrificing the functionality required by those with a more advanced understanding of antenna theory.

3.3 Objectives and Proposed Solution

The main objective of the research was to propose a solution to the aforementioned issues and shortcomings of antenna simulation programs. It was an important point

of the research to address the disconnect between the different simulation engines. This must be done in a way that also addresses the complexity of learning to use different interfaces

In order to overcome the challenges discussed, a universal geometry handling program framework was proposed. The objective of which would be to provide the user with a single interface to create antenna geometries. Then using the same interface this information could be passed, along with other necessary simulation information, to a number of simulation engines in order to calculate the electromagnetic and performance characteristics of the created geometry.

This means that the user would only need to familiarise themselves with and use a single interface. The limitations of certain calculation methods would be overcome as the user would, through the various simulation engines available, have access to any other appropriate method, without having to redefine the geometric parameters. It would also allow for the comparison of different calculation methods as well as the performance of different simulation engines.

The challenge that needed to be overcome was capturing the geometric data in such a way that it is easy to manipulate in order to then pass it to the different simulation engines. Each of these engines required the data to be formatted in different ways so that they were able to interpret the data and therefore carry out the calculations to produce the necessary outputs. This in turn creates another challenge as each engine presents its output data in a different format. In order for this information to be presented in a meaningful manner within a single interface the data would need to be interpreted and manipulated to suit the program's requirements to present the relevant outputs.

Scripting capabilities of the program also needed to be considered as this is a potentially powerful tool that a user could use to run through various scenarios as

quickly as possible and allow for the implementation of optimisation algorithms, all within the single environment of the proposed program structure.

In order to make the program accessible to anyone interested in simulating or designing antennas, a requirement of the program was that it would be developed as an open source tool.

3.3.1 Existing Solutions and Programs

As discussed in Section 2.3 there are a number of simulation packages available, and the proposal is to bring a form of unification between the programs and the calculation methods that they provide.

The ability to run through different iterations of antenna geometries (i.e. scripting capabilities) either for analysis or optimisation efficiently is not a feature that is available in most packages especially those that are free. This leads to different approaches to create these iterations, such as the methodology presented in [20], where a Matlab script was used to implement an optimisation algorithm by writing .nec files to pass to 4NEC2 to perform calculations for each iteration of the algorithm. A similar approach was used by Dietrich et al. [21] where they were looking to automate the design process further.

SuperNEC, a Matlab-based antenna simulation package had extensive scripting capabilities [3] but it is no longer maintained and while compatible with modern machines, it is no longer trivial to install and depends on outdated versions of Matlab (version 7 latest) in order to operate.

FEKO's approach was to unify different methodologies into a single program and provides the ability to perform optimisation routines and run through iterations of a design automatically. This makes it an extremely powerful but it is an expensive

tool and it has a moderately difficult user interface for novice antenna designers to grasp. It also does not give the option for the user to make use of a different engine apart from the one that is built in.

The aforementioned approaches and programs offer the user a wide range of features and a lot of power but there is currently no direct link between engines and there is no program that allows the user access to various engines and their calculation methods through a single interface. Scripting features are also limited in a lot of the software packages available, particularly in open source and free programs; which in turn limits the effectiveness of designs a user of these products can produce.

The methods of capturing the actual geometric parameters from the user vary between different programs. Some packages allow the user to develop through a geometry editor (such as 4NEC2 and FEKO) where the user adds elements and can reposition and resize objects in a "click and drag" manner. Most packages require the user to insert x and y coordinates (and z for 3D simulations) for the starting and end points of each particular element. The user also needs to define the placement of sources, transmission lines etc. Then depending on the simulation method the geometry and simulation space is either segmented or meshed. This approach is suitable for geometries with few elements or single element structures, but it becomes slightly cumbersome when needing to define array-type structures or a space with multiple antennas. In this regard, standard rules of design of such structures may be able to be applied in order to make the process of entering arrays and other geometries more efficient. This leads to the idea of structuring the program as somewhat of a compiler, the structure of which will be discussed further in Chapter 4.

3.4 Methodologies

The project entails the design of a piece of software, several engineering design methodologies were investigated (summarised by Adams in [22]). Most of these methods relate mostly to mechanical or physical products that required material and cost considerations. The selected methodology is to be manipulated slightly in order suit the research presented. There are also considerations around validating the proposed program framework to ensure that it will perform as expected. Understanding the objectives and bounds of the research is an important consideration, therefore scope limitations will be discussed in this section.

3.4.1 Overall Methodology

The methodology selected to approach the design and development of the proposed software framework was that presented by Pahl et al.[23]. This methodology breaks the design process into four distinct phases, listed as follows:

- Task Clarification—This phase's purpose was to identify requirements as well as investigate possible existing solutions and attempt to identify possible challenges. The identified problems are highlighted in the problem identification given earlier in this chapter along with the existing solutions;
- Conceptual Design—The purpose of this phase was to further investigate the problem in order to produce an initial design to provide a solution to the identified problem. This was achieved through the proposal process and research in order to determine a viable solution. In the context of this research this was achieved through literature review, which is presented in Chapter 2;
- Embodiment Design—The purpose of this phase was to further elaborate and complete the design. In terms of the research this involved determining the

best languages to use in order to develop the program as well as give further detail on the structure of the program as a whole. Testing and validation methods were also decided upon in this phase. Through these testing methods the framework was enhanced and altered where required;

- Detailed Design—The design will be finalised in this phase and this will involve making final changes to the design with the knowledge gained from the previous phases. The final product being the "specification of information" in the form of the documentation of the design of the proposed program that will allow for full development of the complete program to take place. The information of the full and final framework design is detailed in Chapter 4.

3.4.2 Validation and Testing Methodologies

To validate the proposed framework, an implementation will be developed with limited functionality that will align with the scope limitations given in Section 3.5. This implementation will be based on the framework as proposed in Chapter 4. It will provide a high level view of the capabilities of the framework as a whole.

The testing method was based on the IEEE 1597.2 - 2010 standard [24] which gives a set of benchmark antennas with parameters and performance attributes that they are expected to meet. This meant that successful validation would depend on whether or not the benchmark antennas' geometry could be accurately represented, simulated and match the expected performance characteristics presented in the standard. This will determine if there are any discrepancies in the handling of the geometries while passing data to the different simulation engines.

3.4.3 Coding Methodology

As all code was written by the author the decision was taken to make use of the waterfall approach with a feedback loop. This meant that a particular section of the program was coded to completion before moving on to the next section. If there were dependencies between the current and previous sections, necessary adjustments were made to any previous sections to ensure the implementation of the current section was unhindered. It was also important to be mindful of the effects that any changes could have on any of the other completed sections.

While the framework does not limit what languages can be used it was decided that, for the initial implementation, that Octave would be used. This was because Octave is completely open-source software, it has built-in GUI capabilities that were useful in simplifying the design of the interface, scripts are easily created for the Octave environment and there are significant numerical tools readily available without the addition of external libraries [25]. Another reason is that, because the educational aspect is important, MatLab and Octave are widely used and understood by undergraduate students in the engineering environment.

3.5 Scope Limitations

Due to the enormous scope that the field of Computational Electromagnetics and antenna theory presents, the research will take place within strict bounds.

- The focus of the work will be on the requirements to develop the framework, more specifically the “geometry handling” functionality that it provides. Therefore the actual computation algorithms and implementation of the simulation engines being used fall outside the scope. Only the communication links

between the engines and the proposed framework will be investigated

- Only wire type structures as presented in Section 2.4 and patch antennas will be investigated and explored within the scope of the implementation;

- Specific example problems that were presented are ones that could be expected to be found at an undergraduate level, it will then focus around the design and analysis of basic antennas and highlight a limited range of characteristics and performance attributes of the different antennas;

- The NECpp [26] engine was chosen to be the primary engine used for implementation and testing purposes so that the communication links could be developed and clear demonstrations of the framework's capabilities could be shown. In order to test the universality of the program the FEKO engine was also integrated into the implementation;

- The framework that is being developed should be expandable to allow for future improvements and development, to this end all code was made available in Github at the following link: https://github.com/Rokie01/RC-SIM;

- As it is a requirement that the program is developed as completely open source, it was decided to use Octave as the main platform for development. Octave allows for the development of GUIs, has the required scripting capabilities [25] and parallelisation of Octave is possible and is widely investigated in literature.

Chapter 4

The Framework

This chapter describes the framework designed to address the problems presented in Chapter 3. The structure of the framework is discussed in detail where different types of interfaces as well as the back-end architecture is explored. Along with the breakdown of the framework, results from the developed implementation will be analysed to demonstrate how different components of the framework were assembled. Examples of the implementations capabilities were explored and the results of these tests are presented to validate the ability of the framework to perform cross-platform simulations successfully.

4.1 Objectives of the Program

The program provides an interface that allows a user to create antenna geometries and simulate them, using a number of different methods. It allows for the use of multiple simulation engines at the back end while only a single geometry is created. It also allows for performance characteristic-based design, in other words a user would ask for a particular attribute (gain, VSWR, impedance etc.) and an appropriate geometric structure would be created.

The program comprises of two main interface options, namely, a GUI and interacting through scripting. The GUI provides a visual representation of the design as it is created and edited, it provides the user with clear instructions and set inputs relevant to respective antenna types being investigated. The scripting interface will allow the user to achieve all the functionality contained in the GUI but will allow for more freedom in terms of optimisation and making subtle changes to designs by being able to specify exact dimensions and easily running through iterations of designs. It is important to strike a balance between user experience and functionality in these interfaces. The hope being the creation of an environment in which amateur and professional users can perform simulations and analysis easily and efficiently. One of the outputs research was an early implementation of the software with much of the core functionality in place.

The focus for the research was to define the structure of the program as a whole with emphasis on the manners in which the capturing of geometric information occurred. More important will be the production of a framework along with an understanding of the requirements to consider in order to handle different antenna geometries and performance attributes. It also allows the user to create input files that would allow for multiple simulation engines to be used. All of this being contained in a single package means that there is a degree of added convenience when compared to having to learn how to use different interfaces.

The presented manner to capture the geometry from the user would be to essentially develop the rules and structures of a standardised antenna geometry language. This language would need to be adaptable and make provision for potential developments in antenna geometries in the future.

The geometry capture element of the program would be loosely based on existing software such as LaTeX and other TeX based applications where plain text is modified

using mark-up language in order to develop the desired output. It is created such that there are distinct layers, with each layer offering the user higher level functions and macros in order to achieve and create the output that they require. There is the base of the TEX language, then there are TEX macros available in order to easily define text and formatting properties, LATEX then added another layer of macros further improving ease of use and adding document classes in order to improve on the user experience [27]. This essentially means that the program would employ a Domain Specific Language (DSL) to provide a means for the user to input their desired geometry and operate as a compiler to interpret this input and provide the required output. The advantage of this is that the antennas can be broken down into their base components, allowing for freedom to define materials and electromagnetic properties at a low level while the user is exposed to higher levels of the design process. It allows a user with any new type of geometry to define it from the lowest levels up without significant changes to the code; this should, ideally, be achievable through the higher level macros.

4.2 Architecture Overview

The framework can be thought of as a "geometry handler" as it is able to take inputs from a user as any number of geometry combinations and then process this information into a form that can be passed external computational engines. It is also able, by means of scripting, to iterate through different designs and run optimisation algorithms to allow for desired performance attributes to be realised. A high level breakdown of the structure of this framework is given in Figure 4.1

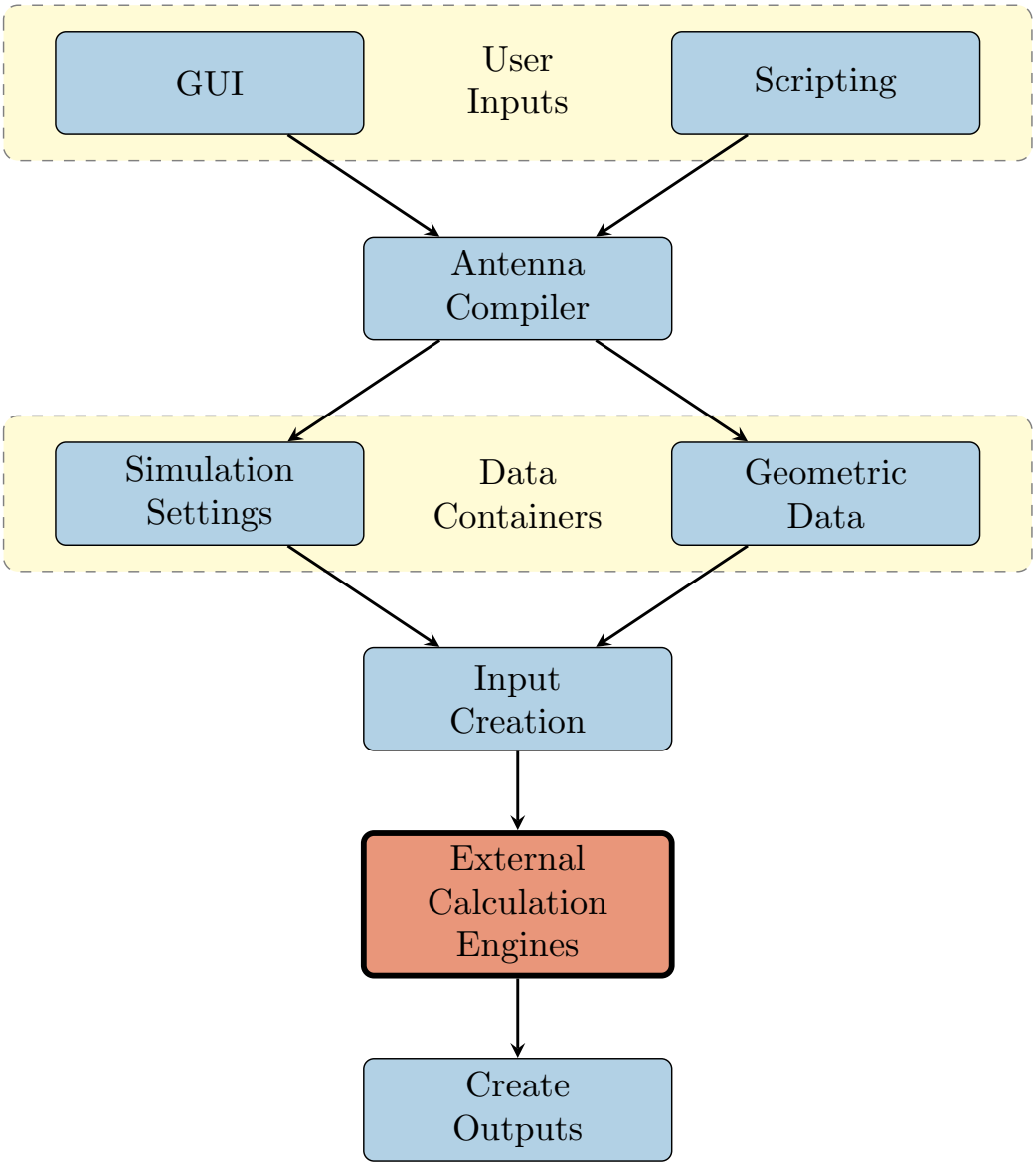

Figure 4.1: Overall architecture of proposed framework

4.3 Antenna Compiler

The compiler component is made up of the front-end or interfaces through which the user can interact to create the desired geometry. It also encompasses all the functions and back-end functionality responsible for storing all the required geometric data and simulation parameters.

When the user interacts with the interfaces they pass information on to functions

that are responsible for passing the user inputs to functions. These functions are then responsible for manipulating the data into a form that can be handled easily. In the presented implementation the information is captured and contained within two main "structs", which are containers in Octave that allow for a single struct object to be defined which can contain multiple parameters/components of differing variable types.

The main objective of the compiler portion of the program is to produce these two structures. The defined structs are created to contain all the geometric data and the simulation parameters (including output parameters such as desired plots, results etc.) respectively. The decision to store the information in this way was made as it allowed for clear division between simulation and geometric parameters, it was possible to keep the struct objects the same between the graphical and scripting interfaces and it allowed for the storage of varying element types as well as different sized objects within the same structure.

The approach to capture the geometric information was to make use of a number of base antenna components from which other antenna structures and arrays could be constructed. For example, from the base element of a wire, a Yagi-Uda antenna can be created as this is simply an array of wires. Table 4.1 gives details of the base elements that were included within the scope of this work. From this information it can be seen that there are significant differences in how these objects are defined.

The base geometries as shown in Table 4.1 were defined as classes in the implementation. This allowed for them to be treated as objects and made it possible to create arrays containing different and unique antenna components and that could be traversed easily in order to interpret the data and pass it to necessary functions.

For more complex and array type antenna geometries, such as the Yagi-Uda antenna, functions were defined with specific inputs related to parameters of the antenna.

Table 4.1: Base geometric elements properties [1]

Base Structure	**Structure Type**	**Properties**
Wire	Wire	• Total wire length • Wire Radius • Number of Segments
Helix	Wire	• Wire Length • Wire Radius • Number of turns • Radius of starting point (i.e. first turn) • Radius of ending point (i.e. last turn) • Distance between turns • Slope gradient (angle) of turns • Number of Segments
Rectagular Patch	Volumetric	• Length and Width of patch • Length and Width of ground plane • Thickness of dielectric substrate • Permittivity of substrate material

These inputs were then used to build the antenna from the base wire objects without requiring further input from the user in order to create the individual elements. This meant that the only objects created in the simulation space were the defined base elements, thus simplifying the interpretation of the geometric data as fewer object types required tracking and storing. The classes presented in the implementation are

summarised in the UML diagrams in figure 4.2.

Figure 4.2: UML Class Diagram of the implemented program

The ability to store different variable types was therefore important in respect of geometric data. As mentioned before, the struct object allowed for all this information to be stored in a single container by defining arrays of each of the different base-element class types as parameters in the struct. Similarly, the second struct was used to store the simulation parameters, including data pertaining to excitation frequencies, selection of simulation engine, frequency sweep parameters and the form that the outputs will take (i.e. radiation patterns, current distributions etc.). Unlike the geometric information, stored in array objects, these were stored as individual scalar values within the struct.

In order to execute the functionality of building complex array antenna types with their own unique properties using the properties of the base level structures; it was decided that an object oriented approach would be effective. This allows for the defining of any antenna type as a Class with its own properties and parameters. The

base elements will also have their own classes that will be used in the higher order antenna classes. This means that while the user is exposed to design process of the higher order antenna, the compiler is simply concerned with the properties of the base elements present in the simulation. This information, pertaining to the base elements, is then passed on to the conversion portion of the framework.

It is important that the necessary functions are in place in order to obtain and effectively pass on required information from the user through the compiler to the external calculation engines.

There will also be functions that would deal with the creation and manipulation of antenna arrays. These arrays could be made up of numerous geometries and considerations will also have to be made in terms of grounding, as well as transmission lines connecting antennas and the phase issues that would be introduced. The functions will be responsible for the placement of all the antennas contained within the antenna array in the simulation space in any orientation that the user requires. These functions were developed to be independent of the user interfaces. Regardless of the interface that the user interacts with the same functions are used with the same input arguments in order to develop the geometry and set up the simulation.

4.4 Execution in the Interfaces

The use of separate user interfaces was seen as key to allow for broader functionality especially in terms of optimisation with scripting. To keep the data handling process consistent, the functions used to create and consolidate the geometric information were the same between the interfaces. The main differences being the manner in which the user would interact with these functions. In the case of the GUI, the user would interact through buttons and other user interface (UI) controls that

were linked to the core functions via callback functions. In the case of scripting the functions were directly interacted with by the user who would then be responsible for manually entering required inputs to the functions. These functions were responsible for organising data into structures that would be used for plotting the structure in the home screen, preparing the geometric and simulation information for conversion and create complex geometries that are made up of the basic elements e.g. Yagi-Uda antenna.

In order to test the validity of the framework, a high level implementation was produced with the basic functionality in place. This section discusses the methods used to produce the implementation and highlights a potential approach to developing the framework. These methods encompass the design of the user interfaces as well as how functions, classes and other structures were used to achieve the desired functionality within the compiler section of the architecture to collate geometries and simulation parameters. The communication links to the external simulation engines were developed. Finally, output procedures were developed and the handling of this information was also implemented.

This basic implementation was developed within the scope and limitations mentioned in Section 3.5, this in turn means that it has limited functionality and is designed as a proof of concept for the framework.

In relation to data handling there were two critical elements. Firstly, the data structures in where different data would be stored (discussed in Section 4.3) and secondly how the information would be passed between the various functions and the different components of the framework. This second component was dependent on the interface being used. The following subsections describe the two UIs that were implemented in detail and also discuss the manner in which the different interfaces achieved the interaction with functions, ensuring that the data outputs were created

correctly.

4.4.1 Graphical User Interface

The GUI interface was designed to provide visual representations of the antenna geometry throughout the design and simulation process. It allows the user to create and specify the geometric characteristics of antenna elements and set up simulation parameters. The layout was simplistic and the input requirements are clearly stated in order for basic structures to be created.

The GUI was designed using Octave's UI controls to create graphic handles in order to display information and acquire inputs from the user. These UI controls took various forms for example push buttons, text input boxes, radio buttons and drop-down lists [25]. When interacting with some of these controls the user would execute callback functions. These callback functions are linked to specific controls and each performs a different task. These range from executing code responsible for creating structures, opening different windows, capturing data, executing simulations and creating outputs. The functionality of these controls is discussed later in the section.

In order to store the necessary information (i.e. create the structs that were discussed in the previous section) within the GUI, the approach was to use the graphics handles themselves to store the information. This was achieved by making use of the "Userdata" parameter contained within some of the UI Control options available in Octave. This parameter allowed for the storage of any information whether it was scalar information, arrays or structures. Most crucially by passing the graphics handle to a callback function it was possible to manipulate the data from within the functions, thus providing a similar effect as passing a variable by reference, which was not a standard feature of octave. The graphics objects used to store the information were the main menu button objects (see Figure 4.3), with the "Simulation Setup"

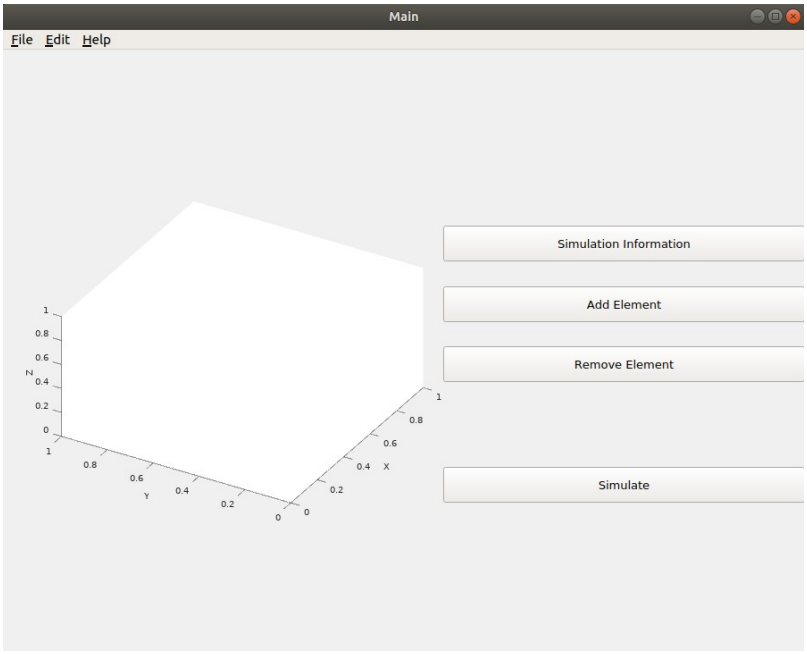

Figure 4.3: Screenshot of the main screen of the program

button containing simulation information, the "Add elements" button containing all of the geometric data and the "Simulate" button containing the callback that calls all the functions to kick off the conversion and execution portion of the framework.

The manner in which the UI control objects were implemented and the functionality contained within the GUI are discussed in the following subsections. This includes detailed explanations of the menus and options available to the user within the GUI as well as how data is stored and which controls were used to achieve this.

Main Screen

The main screen is the initial screen that the program opens to. It is from here that all the menu options are accessed in order to set up simulation parameters, add/remove elements and execute the calculation and output generation process. There is an interactive visual representation provided of the antenna geometry which is updated as changes are made.

The simulation button is the UI control that links to the capturing of simulation parameters. The parameters of this control include information of the physical appearance of the button such as position and sizing. More importantly it contains a Userdata parameter and this is where the settings struct discussed previously is contained. There is a callback function that executes when the button is pressed and links to the Simulation Set-up menu.

The "Add Element" button was set up in a similar way to the "Simulation Information" button. This button, however, contained the geometric information collected from the user through the "Add Element" menu.

Simulation Set-up Menu

Once the user presses the "Simulation Information" button in the Main Menu, the Set-up menu is presented as seen in Figure 4.4. The purpose of this menu is to allow the user to provide required simulation information. The current implementation allows for simulations at a single frequency to be carried out as well as a linear sweep through a user-defined range of frequencies. It is also possible to select which simulation engine will be used from this menu. The user will also be able to define the outputs they would like to receive once the simulation is completed. The output options are as follows:

- **Radiation Patterns -** The user has the options to request two-dimensional or three-dimensional plots of the radiation patterns. The values calculated are given as gain in measured in dBs. The user will also be able to define the resolution of the plot by defining the angular increments in both the phi and theta directions.
- **VSWR Plot -** This is an important characteristic in order to determine the

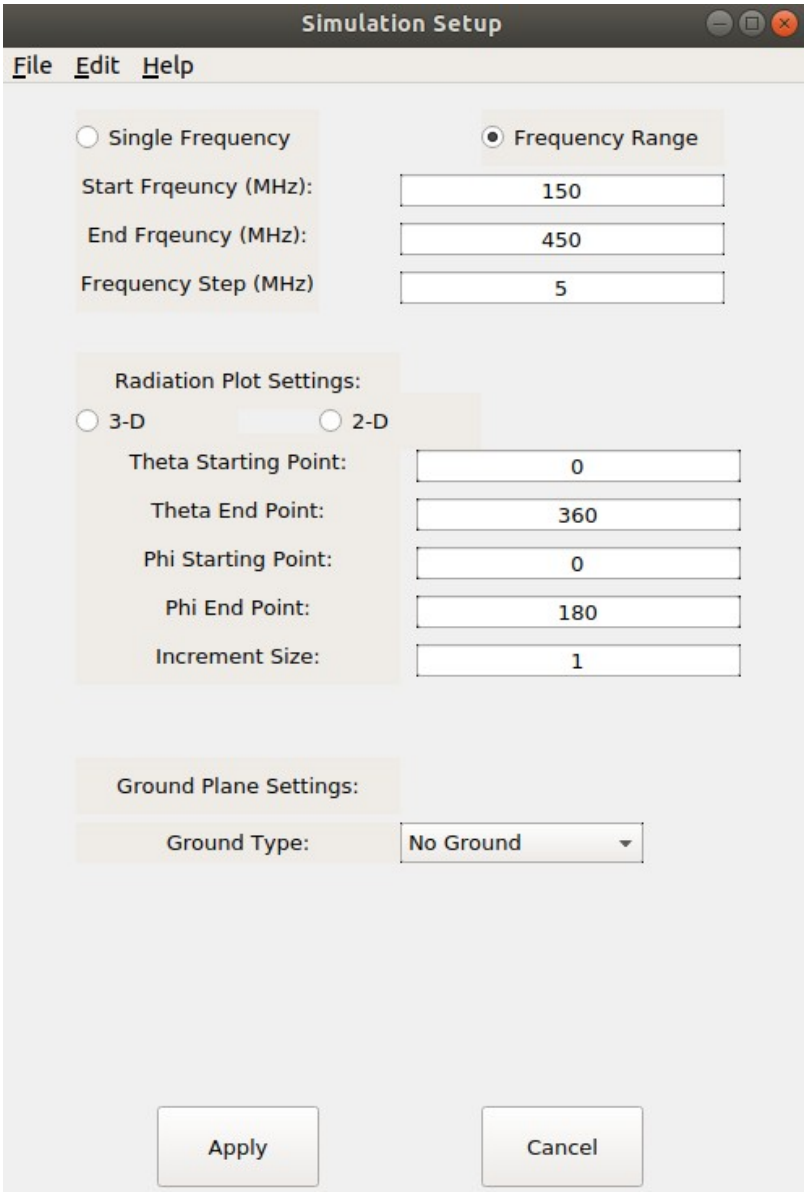

Figure 4.4: Screen grab of the simulation setup menu

expected performance and efficiency of the antenna. The user will be able to request a plot of the VSWR vs. frequency or a specific parameter that may be varied for optimisation purposes.

- **Impedance Plot -** Impedance information, like that of VSWR can be requested and plotted against frequency or another selected variable.

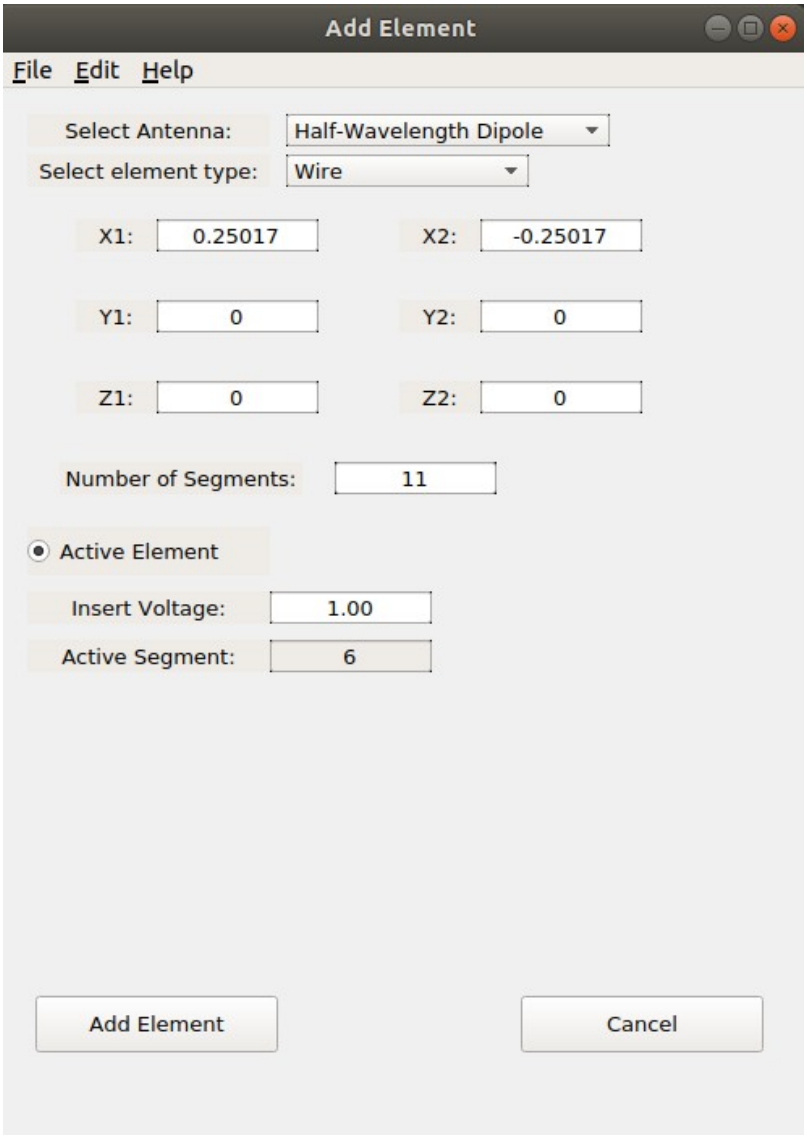

Figure 4.5: Screen grab of the element creation menu adding a predesigned 300MHz active half-wavelength dipole antenna

Add Element Menu

From this menu the user is able to add antenna elements to the simulation space. The implementation allows for the addition of straight wire elements and helical antennas as well as the creation of full Yagi-Uda antennas. The geometric properties of these elements can be defined, including the segmentation properties for simulation and the positioning of voltage sources when considering active elements.

While the properties are customisable, the user also has the option of choosing from a list of predefined geometries. These geometries are common antenna designs and their dimensions are based off of the defined simulation centre frequency's wavelength. The characteristics of the antennas were taken from tested and widely accepted designs; they can however be customised by the user.

4.4.2 Scripting Interface

There was substantial emphasis placed on the scripting capabilities of the program. The user should be able to perform all processes contained within the GUI. More importantly the scripting framework should allow the user to run through as many different iterations as they need to. This will allow the user to have greater control over the design process and allow them to implement optimisation algorithms of their own.

At the core of the scripting interface is the aim of creating the two structures as mentioned in Section 4.3, namely a Geometry Struct and a struct containing the simulation settings.

With regards to the passing of information between functions in the scripting interface the approach was relatively simplistic as the functions were developed to be independent of the interfaces. This meant that the functions could be called directly through the script by passing the appropriate values to the function. The user would have to create their geometry and set simulation parameters by passing appropriate values to the functions in order to generate the necessary objects through the class definitions and constructors and then call the necessary functions to execute the simulation

4.5 Input Creation

Once the user has entered all the necessary parameters and are satisfied with their inputs, they call the Simulate function (directly in the case of scripting and through UI control callback functions in the GUI). The process from this point onwards is identical between the two interfaces. The simulate function simply takes the two created structs as inputs.

The first task performed in this function is to create the required input file. The existing simulation engines expect inputs in different forms. Some of these inputs take the form of encrypted or binary files that are unique to and only understood by a particular program. The engines investigated in this research take their inputs as plain text files. While the formats of these inputs may differ from engine to engine, the use of plain text means that it is possible to generate these input files outside of any interface linked to the particular engine.

This meant that functions were written in order to interpret the data contained within both the geometric and simulation parameter structs. This data was then written to the appropriate plain text file — ".nec" and ".pre" formats for NEC and FEKO respectively — in preparation to run the simulations.

There were two main functions created in the implementation, one responsible for creating NEC input files and the other FEKO input files. These functions called a number of sub-functions, each one responsible for populating either the FEKO or NEC input file with specific pieces of information. A breakdown of these functions is given in Table 4.2 below.

Examples of the resulting NEC and FEKO input files created from these functions are shown in Figures 4.6 and 4.7 respectively. From these examples the differences in the two formats can be seen. These differences include:

Table 4.2: Breakdown of functions used to create input files

Main Function	**Sub-function**	**Description**
writeNEC	writeGeometryNEC	Writes straight wire geometry data to the input file
	writeHelixNEC	Writes helical structure data to the input file
	writeFrequencyNEC	Writes the frequency information to the input file
	writeExcitationNEC	Writes excitation information to the input file
writeFEKO	DP	This function writes the position of the endpoints of wires to the file
	BL	Joins the points created in the previous function
	FR	Sets up the frequency parameters as per the user's request
	HE	This feeds information about helical structures to the engine
	FF	Captures information to create far-field pattern
	IP	Responsible for segmentation data and dimensions
	A1	Contains source information like type, magnitude and position

- For wire structures the GW command requires a tag number, a 3-dimensional cartesian beginning and end point and the number of segments. FEKO reads in beginning and end points seperately as DP instructions which are joined

```
GW          1       5   0.24983            0       0   -0.24983            0       0   0.005
GE
FR          0      61       0       0     150       5
EX          0       1       3       0       1
RP          0     181     361    1000       0       0       1       1       0       0
EN
```

Figure 4.6: Example of a NEC input file generated for an active half-wavelength dipole at 300MHz frequency with instructions to generate a far-field radiation pattern

using the BL command.

- Source locations, in NEC, are defined using the active element's tag number and the number of the segment (as counted from the beginning point to the end point) on which the source is to sit. In FEKO the active segment's beginning and end points are defined separately from the entire element's end points. The implementation distinguishes the active segment end points from the entire element's ends using the prefix "S" instead of "P".

- In the NEC file the number of segments is defined simply as an integer value when defining the wire element. In FEKO the maximum segment length is set in the IP card (similar definitions are used for triangular, cubic and tetrahedral segmentations in other simulation types involving different shapes). This means that the length of a single segment forms part of the input instead of the total number of segments [8].

.

4.5.1 External Engine Calculations

This section exists outside the investigated framework. Once the geometric and simulation information has been formatted and written to the appropriate input file, the information is passed to these external engines by passing a Unix command through Octave's 'system' function. The commands used were as follows:

```
DP: P1 :  :  :  :  : 0.24983 :     0 :     0
DP: P2 :  :  :  :  : -0.24983 :     0 :     0
DP: S1 :  :  :  :  : 0.049966 :     0 :     0
DP: S2 :  :  :  :  : -0.049966 :     0 :     0
IP:  :  :  :  :  : 0.005 :  : 0.099933
BL: P1 : S1
BL: S2 : P2
BL: S1 : S2
EG:     0 :     0 :     0 :  :  :  :  :  :  :  :  :     1

A1:     0 :    -1 :  :  :  :     1 :     0 :     0 :     0 :     0
FR:    61 :     0 :  :  :  :  150000000 :    5000000
FF:     1 :   181 :   361 :     0 :     0 :     0 :     0 :     1 :     1 :  :  :  :     0
EN
```

Figure 4.7: Example of a FEKO input file generated for an active half-wavelength dipole at 300MHz frequency with instructions to generate a far-field radiation pattern

- **NEC**

  system('nec2++ -i input.nec -o output.out');

  This takes — in this example — the input.nec file and pulls it through the NEC++ engine and produces the output.out file [26].

- **FEKO**

  system(runfeko geometry.pre);

  This firstly runs PREFEKO to create a '.fek' file from the pre file and then passes the result to the solver which produces a geometry.out file containg results of the calculations [8].

These engines then perform calculations using the methods discussed in 2.2.

The results of these calculations are then written to output files that are also plain text formats. This is beneficial because, like in the case of the input formats, you are not dependent on any particular interface in order to use and interpret the results. The interpretation of the output files will be discussed in the next section.

4.5.2 Output Interpretation

The purpose of this component of the framework was to interpret and translate the calculation results into a form that could be used to present the requested information to the user. Before generating the actual plots, the first step was the extraction of the relevant information. This involves locating the required data in output text files and storing it in appropriate data structures. Both NEC and FEKO output files follow distinctive formats and make use of specific key words and identifiers that could be used as flags when traversing the file for particular information.

Once these flags were identified the required data could be extracted using text file and string manipulation commands. These extracts were then converted from strings into necessary formats and then stored in data structures such as arrays that were then used to create output plots. In the implementation these operations occur in functions that are used to generate the specific outputs, whether to plot radiation patterns, impedence plots etc. each function extracts different information to create the plots.

While the different simulation engines presented their calculation results differently; the information was extracted into the same structures. This meant from this point forward the program is independant of the engine used.

4.5.3 Generate Outputs

This component involved the creation of outputs in the form of summarised results explaining the performance characteristics of the proposed design and plots from the data structures created after the output conversion component. The outputs generated in this step depended on user requirements and their original inputs to the simulation setup.

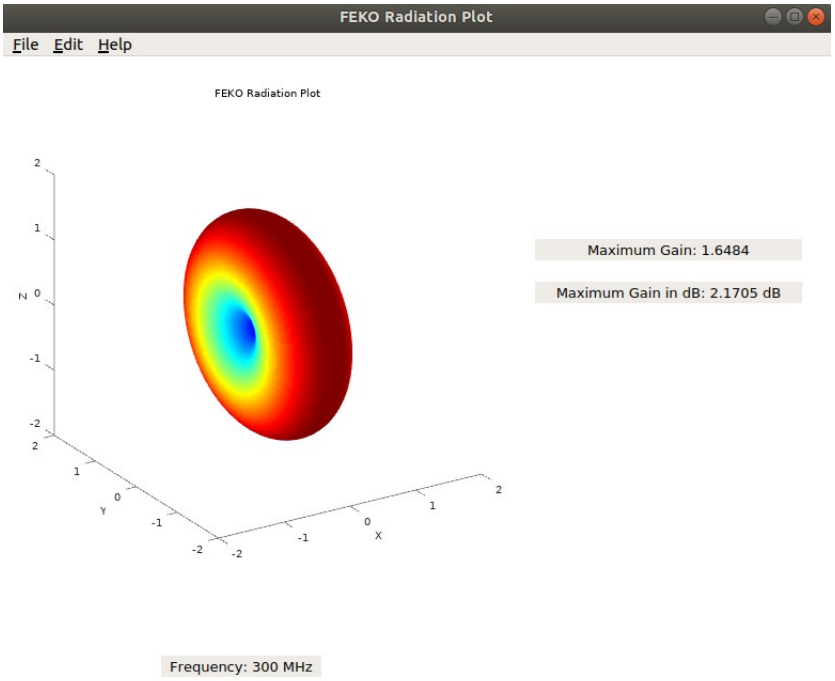

Figure 4.8: Far-field radiation pattern of a 300 MHz half-wavelength dipole

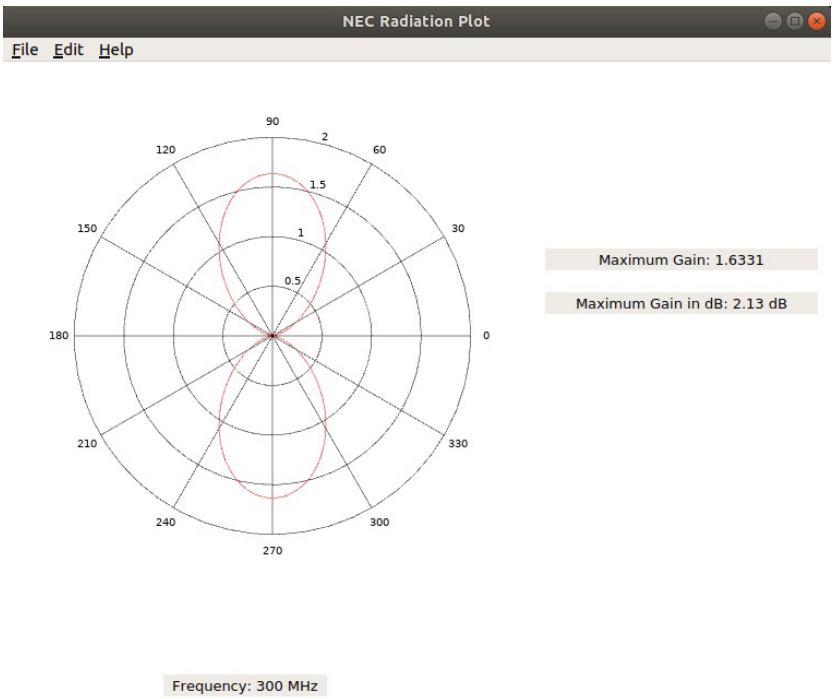

Figure 4.9: 2-D representation of far-field radiation pattern of a 300MHz half-wavelength dipole

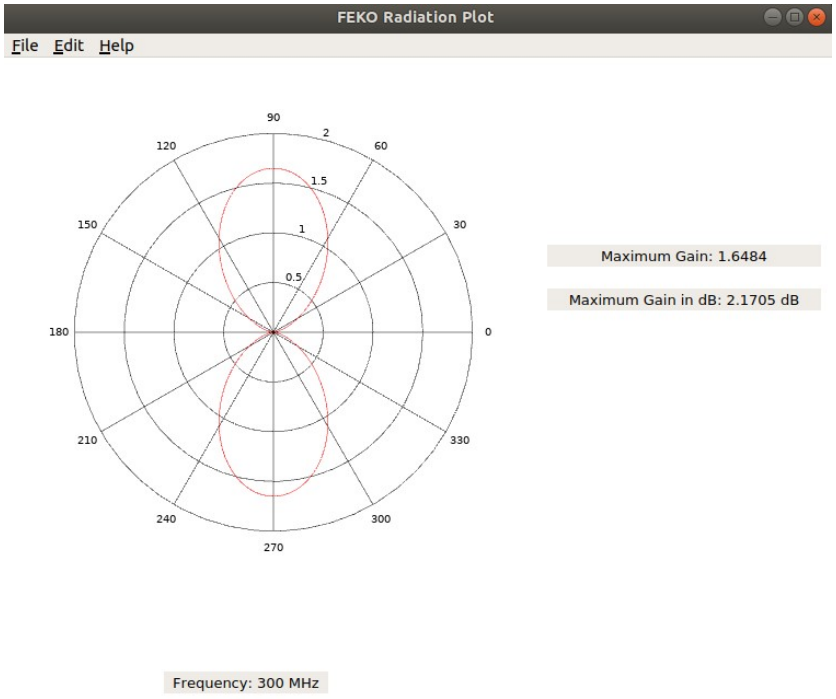

Figure 4.10: 2-D representation of far-field radiation pattern of a 300MHz half-wavelength dipole

The outputs included in the implementaition includes far-field radiation plots (both 2-D and 3-D) and impedance plots. Interaction with the outputs was also considered, for example being able to sweep through frequency ranges and immediately updating graphics to display the outputs for different excitation frequencies. These interactions should be intuitive to allow for clear understanding. An example of the outputs in implementation is shown in Figures 4.8, 4.9 and 4.11.

Comparing the results shown it can be seen that there was a small difference in gain values between the FEKO and NEC results. Both results were close to what is stated in literature at 2.15dbi. It can therefore be taken that these simulations were a success. See Appendix B for further examples that show the correlation between the two engines and literature.

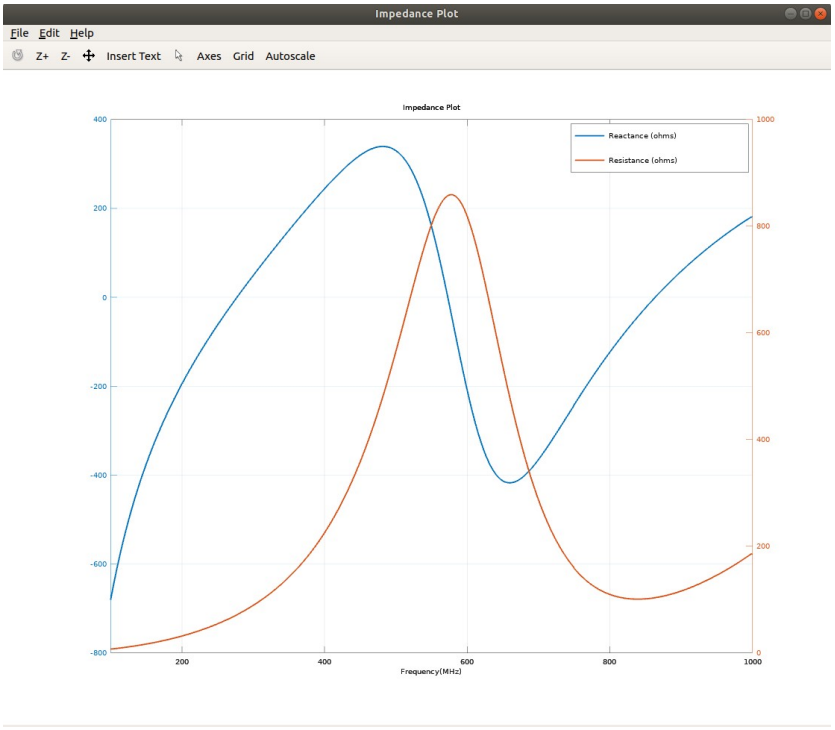

Figure 4.11: Input impedance plot of 300MHz half-wavelength dipole for 100MHz - 1GHz

```
addpath('../geometryDefs/');
addpath('../plotFunctions');
addpath('../writeNEC');
addpath('../writeFeko');

  ant = struct('wire',wire(0),'helix',helix(0), 'no_of_elements', 0);
  ant.no_of_elements = ant.no_of_elements +1;
  i = 1;
  yagi_wires = yagi(ant.no_of_elements,12,0.5,0.6,0.45,0.15,0.1,0.1,0.001,5,1,3);
  for j= 1:size(yagi_wires)(2)
    ant.wire(i) = yagi_wires(j);
    i = i+1;
  end
  plot_geometry(ant, figure);
settings.frequency_start = 300;
settings.frequency_increment = 0;
settings.frequency_steps = 1;
settings.ground_flag = 0;
settings.ground = ground_card(1 ,0, 0, 0);
settings.rad_theta_start = 0;
settings.rad_phi_start = 0;
settings.rad_resolution = 1;
settings.rad_theta_steps = 181;
settings.rad_phi_steps = 361;
settings.rad_theta_end = 180;
settings.rad_phi_end = 360;
simulate(settings, ant);
```

Figure 4.12: Script to simulate a half-wavelength dipole antenna

Figure 4.12 shows a script that generates the same antenna as shown in the simulation results above. There is more freedom to make adjustments to finer details and parameters when using the scripting interface, the user has the user has the ability to develop functions and algorithms as needed.

Chapter 5

Conclusion

The focus of this dissertation was the development of a software framework for an antenna-simulation program allowing, through a single interface, access to multiple computational engines. In so doing, the weaknesses of certain calculation methodologies were overcome by giving the user access to a more suitable method all within the same interface. This also has the added benefit of reducing the number of interfaces that a user needs to familiarise themselves with.

The framework was tested and validated using a basic, high-level implementation. It showed the capabilities of the framework to not only deal with interaction between two simulation engines (NEC and FEKO), but it also showed the overall geometric and simulation information handling capabilities. The implementation involved the defining of basic structures that could form the basis of most antenna types. Classes and functions were then used within the implementation to create macros that allowed for more complex structures to be derived from the previously defined base elements. There were also predefined antennas added to the GUI that were dependent only on the centre frequency to define the dimensions of the antennas. This simplistic approach to adding antenna geometries to the simulation space means that the user does not require extensive knowledge of antennas but can still generate

a number of antenna designs.

Another useful concept contained in the framework, is the use of macros to make the designing of more complex arrays easier. This was achieved by defining base antenna components as classes and making them the building blocks for the larger arrays. Using functions to manipulate the size and position of elements it is possible to build a complex array in a matter of seconds. Literature was used to determine the ideal dimensions of these antennas.

Test cases showed that the implementation was able to transform the user inputs into either NEC or FEKO text-based input files that were successfully used to carry out simulations in either of the available engines. Results between the NEC and FEKO simulations were similar and there was alignment with literature in terms of the expected performance characteristics. The comparison of the two engines is a useful tool in itself as it gives a clearer picture of what actual performance could be.

The program presented in the research was basic in its implementation but it shows that the basis of the framework is valid. This has wider implications because, with further development, this could become a useful open-source tool that could be used for educational purposes as a design tool for hobbyists or professionals alike. Another benefit to the approach presented is that for any new compatible engine or any enhancements made to existing engines, the process is well defined to incorporate these into the program. This allows for the user to reap these benefits without any significant changes to the interface they engage with.

Octave was selected as the language to develop the initial implementation. This was due to the fact that it is completely open source and it contains all the necessary tools required to develop the both graphical and scripting interfaces along with the availability of significant numerical capabilities. The developed framework, however, is not limited to specific languages and should the need arise in the future to develop

using an alternative language, the process and structure of the program as laid out in the presented framework should remain constant.

In terms of future development, there is a wide range of functionality that can be added. This includes the addition of other computational engines, the integration of built in optimisation tools and algorithms, and the inclusion of more pre-designed antennas. There is essentially unlimited scope to what features can be added to the implementation of the program.

The base principles and structure of the framework will remain constant, in that the geometry and simulation settings will be captured from the user once; this information will then be stored in two main containers—one containing geometric data and the other holding simulation information and the triggers to select the appropriate engine. It is then necessary to implement the required processes to convert the information contained in these two structures into an appropriate input format for the selected engine to execute the simulation. The data will then be presented to the user in the requested format.

References

[1] J. Kraus. *Antennas*. New York City: McGraw Hill, 2nd ed., 1988.

[2] T. Rylander, A. Bondeson, and P. Ingelstrom. *Computational Electromagnetics*. New York: Springer, 2nd ed., 2013.

[3] A. Fourie and D. Nitch. "SuperNEC: Antenna and indoor-propagation simulation program." *IEEE Antennas and Propagation Magazine*, vol. 42, no. 3, pp. 31–48, 2000.

[4] K. B. Baltzis. "The finite element method magnetics (FEMM) freeware package: May it serve as an educational tool in teaching electromagnetics?" *Education and Information Technologies*, vol. 15, no. 1, pp. 19–36, 2010.

[5] D. K. Cheng. *Field and Wave Electromagnetics*. New York: Addison Wesley, 2nd ed., 1989.

[6] N. N. Rao. "Maxwells Equations in Integral Form and Maxwells Equations in Differential Form." In *Fundamentals of Electromagnetics for Electrical and Computer Engineering*, chap. 2–3, pp. 38–105. Pearson Education, 2009.

[7] M. Clemens and T. Weiland. "Discrete Electromagnetics: Maxwell's Equations Tailored to Numerical Simulations." Tech. rep., Darmstadt University of Technology, Darmstadt.

[8] Altair Hyperworks. *User Manual for FEKO 14.0*. Altair Engineering Inc., 2015.

[9] W. C. Gibson. "The Method of Moments." In *The Method of Moments in Electromagnetics*, pp. 33–62. Boca Raton: Chapman & Hall/CRC, 2008.

[10] L. Sevgi and E. Arvas. "A Tutorial on the Method of Moments." *IEEE Antennas and Propagation Magazine*, vol. 54, no. June 2012, pp. 260–275, 2012.

[11] B. M. Kolundžija and A. R. Djordjević. "Introduction." In *Electromagnetic Modeling of Composite Metallic and Dielectric Structures*, Artech House antennas and propagation library, chap. 1. Boston, MA: Artech House, 2002. URL `https://books.google.co.za/books?id=zdR47nZ7oK8C`.

[12] G. J. Burke and A. J. Poggio. *Numerical Electromagnetics Code (NEC) - Method of Moments*. Naval Ocean Systems Center, 1981.

[13] A. Voors. "4nec2 - NEC based antenna modeler and optimizer.", 2020. URL `https://www.qsl.net/4nec2/`.

[14] W7EL. "EZNEC." URL `https://www.eznec.com/eznec.htm`.

[15] PA3FWM. "Xnecview - A program for visualizing NEC2 input and output data.", 2020. URL `https://www.pa3fwm.nl/software/xnecview/`.

[16] R. Garg, P. Bhartia, I. Bahl, and A. Ittipiboon. "Microstrip Antenna Design Handbook." In *Electromagnetic Modeling of Composite Metallic and Dielectric Structures*, Antennas and Propagation Library, chap. 1, pp. 1–4. Artech House, 2001. URL `https://books.google.co.za/books?id=_er1LO5pEnUC`.

[17] G. L. Shaw. "Patch Antenna Design." , no. April, p. 73, 2005.

[18] M. E. Mathekga. "Formulations for Analysis of Probe-Fed Printed Antennas in SuperNEC." 2008.

[19] J. Volakis. *Antenna Engineering Handbook*. New York City: McGraw Hill, 4th ed., 2007.

[20] P. Kadlec and Z. Raida. "Multi-objective self-organizing migrating algorithm applied to the design of electromagnetic components." *IEEE Antennas and Propagation Magazine*, vol. 55, no. 6, pp. 50–68, 2013. URL `http://ieeexplore.ieee.org/document/6781705/`.

[21] J. Dietrich and A. Sebak. "Automating N E C ˜ With Matlab @'." In *Canadian Conference on Electrical & Computer Engineering*, pp. 342–346. 2002.

[22] K. M. Adams. "Nonfunctional Requirements in Systems Analysis and Design." vol. 28, 2015. URL `http://link.springer.com/10.1007/978-3-319-18344-2`.

[23] G. Pahl, W. Beitz, J. Feldhusen, and K. Grote. *Engineering Design - A systematic Approach*. London: Springer, 3rd ed., 2007.

[24] IEEE Standards. *IEEE Recommended Practice for Validation of Computational Electromagnetics Computer Modeling and Simulations IEEE Electromagnetic Compatibility Society*. February. 2011.

[25] J. W. Eaton, R. Wehbring, S. Hauberg, and D. Bateman. *GNU Octave*. February. 5 ed., 2019.

[26] T. C. A. Molteno. "NEC2++: An NEC-2 compatible Numerical Electromagnetics Code." *Electronics Technical Reports*, vol. 3, 2014.

[27] LATEX3 Project Team. "Latex for authors." *World Wide Web Internet And Web Information Systems*, pp. 1–33, 2001.

[28] D. Fleisch. "The Ampere-Maxwell Law." In *A Student's Guide toMaxwell's Equations*, chap. 4. New York: Cambridge University Press, 2008.

[29] C. A. Chen. "Optimum Element Spacings for Yagi-Uda Arrays." , no. 5, pp. 615–623, 1973.

Appendix A

Maxwell's Equations

A.1 Gauss' Law for Electric Fields

This law describes the relationship between the electric field and the charge distribution over a defined surface area. It indicates that the electric field around a surface is dependent on the the amount of charge and the charge density over a particular surface [6]. This law also describes the relationship between electric field strength and the distance from the charge source. If you have a point charge of Q the charge density and therefore the electric field is at its strongest at this point. Gauss' law then describes that if you enclose this point charge in a sphere the total charge on the surface of the sphere is equal to the charge contained within the sphere. In other words the charge Q is distributed over the surface of the sphere. This means that the charge density decreases as the sphere's radius increases; meaning that the electric field strength decreases the further away you move from the point charge [5, 28].

Figure A.1 below shows a positive charge with electric field lines (displayed in red) radiating out in all directions and equipotential lines (shown in blue). Equipotential lines are an indicator of field strength, the closer the lines are together the stronger

the field. It can be seen that as one moves further from the charge the distance between the equipotential lines increases thus the field weakens. It can also be noted that the same number of red field lines cross through each equipotential line but they spread out the further out they radiate. This is analogous to the previously discussed charge enclosed in a sphere and it can be seen how charge density would decrease with an increase in radius.

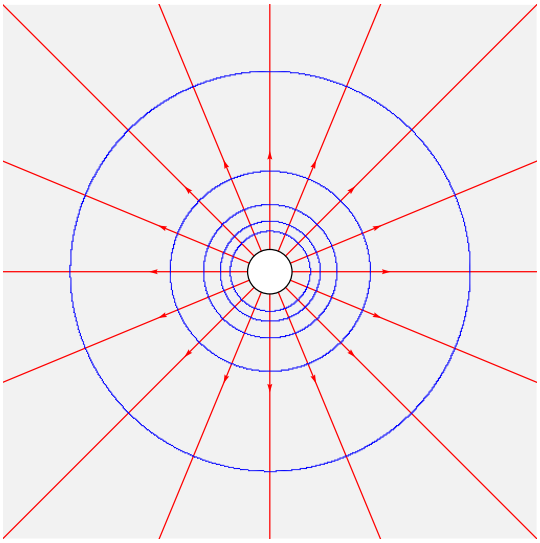

Figure A.1: Positive point charge with electric field lines (in red) and equipotentials (in blue) displayed

Integral Form:

$$\oint_S \vec{E} \cdot dS = \frac{q}{\epsilon_o} \tag{A.1}$$

Where:

- $\vec{E}$ = Electric field vector
- q = Amount of charge (Coulombs)
- ϵ_o = Permittivity of free space

Differential Form:

$$\nabla \cdot \vec{E} = \frac{\rho}{\epsilon_o} \tag{A.2}$$

A.2 Gauss' Law for Magnetic Fields

This law indicates that—regardless of magnetic field strength—the net magnetic flux passing through a closed surface is zero. Another way of viewing this is by using a graphical representation of a magnetic field around an object; noting that the number of field lines entering an object must equal the number of lines exiting the object as depicted in Figure A.2 below. This would be the same for any closed surface regardless of its shape or size. As with the electric field, the distance between the field lines indicates field strength and the figure shows that the field strength is strongest around the poles of the object [6, 28].

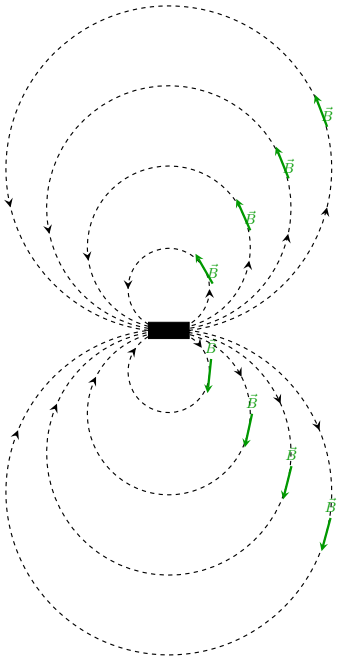

Figure A.2: Example of magnetic field around an object

Integral Form:

$$\oint_S \vec{B} \cdot dS = 0 \tag{A.3}$$

Where:

- $\vec{B}$ = Magnetic field vector
- $\int_S \vec{B} \cdot dS$ = Total magnetic flux

Differential Form:

$$\nabla \cdot \vec{B} = 0 \tag{A.4}$$

A.3 Faraday's Law of Electromagnetic Induction

In 1831 Michael Faraday found that time-varying magnetic fields result in electric fields being formed. He measured a current in a closed loop of wire that was placed in a changing magnetic field, this in turn meant that a voltage or what became known as an electromotive force (emf) was induced around the loop. In order for this current to form it meant that there was an electric field forming around the wire as the magnetic flux passing through the closed loop changed [6, 28]. This relationship is shown in Equations A.5 and A.6 below.

Looking at equation A.5, the electric field around a closed path, C, is related to the change in magnetic flux over time through a surface enclosed by C. The negative symbol is used to maintain a convention of clockwise current flow within a closed loop with respect to the direction of the magnetic flux through the closed loop or the motion of the loop through the magnetic field. This is where the "right-hand-rule" convention arises [6].

Integral Form:

$$\oint_C \vec{E} \cdot dl = -\frac{d}{dt} \int \vec{B} \cdot dS \tag{A.5}$$

Differential Form:

$$\nabla \times \vec{E} = -\frac{\delta \vec{B}}{\delta t} \tag{A.6}$$

A.4 Ampére's Circuital Law:

This law is based the experimental findings by Oersted which were confirmed and refined mathematically by Ampére that electric currents generate magnetic fields. Maxwell further added to this with the idea that changing electric fields would result in the formation of magnetic fields. This addition by Maxwell was crucial to understanding electromagnetic wave propagation without which the development of antennas and wireless communication systems as we know them would not have been possible [6]. Equation A.7 states that the mmf is equal to the sum of the total current as a result of the flow of charge and displacement current on any surface enclosed by the closed path C. The displacement current term came about as an analogous term to Faraday's law in that a changing electric field is responsible for generating a magnetic field. Although it it is not a current in the sense of flowing charge [28]. It is simply a mathematically derived term (see below for a further definition) that results in the same units ($Charge/time$) as current and accounts for magnetic fields created by changing electric fields.

Integral Form:

$$\oint \frac{\vec{B}}{\mu_o} \cdot d\vec{l} = \int_S \vec{J} \cdot d\vec{S} + \frac{d}{dt} \int_S \epsilon_o \vec{E} \cdot d\vec{S} \tag{A.7}$$

or as:

$$\oint_C \vec{H} \cdot d\vec{l} = \int_S \vec{J} \cdot d\vec{S} + \frac{d}{dt} \int_S \vec{D} \cdot d\vec{S} \tag{A.8}$$

Where:

- $\vec{H}$ = Magnetic Field intensity vector (A/m)
- $\oint_C \vec{H} \cdot d\vec{l}$ = Magnetomotive Force (mmf)
- $\vec{J}$ = Current density vector or the sum of currents due to the flow of charge across a surface (A/m^2)
- $\frac{d}{dt} \int_S \epsilon_o \vec{E} \cdot d\vec{S}$ = Displacement Current which accounts for changes in "Electric Flux" and is a measure of the rate of change of this flux which relates to change in the charge density over a defined surface (Q/s)
- $\vec{D}$ = Displacement flux density vector (C/m^2)
- μ_o = Permeability of free space

Differential Form:

$$\nabla \times \vec{E} = -\frac{\delta \vec{B}}{\delta t} \tag{A.9}$$

Appendix B

Further Simulation Examples

B.1 Yagi-Uda Antenna

The following example shows the simulation of a six-element Yagi-Uda antenna as presented in [1, p.481-485]. The simulation makes use of the predesigned Yagi-Uda antenna option built into the program. The literature states that an antenna built with these dimensions should have a gain of approximately 12dBi [1, 29].

Figure B.1 shows the "Add Element" menu for the Yagi-Uda antenna. The "Standard Yagi-Uda Antenna" option was selected, from here the parameters are automatically populated using dimensions determined using fixed ratios with repect the centre frequency's (in this case 300MHz) wavelength, these ratios, according to [1, 29], are as follows:

- Driven element length = 0.46λ;
- Reflector element length = 0.475λ;
- Director length = 0.44λ;
- Driven-Reflector spacing = 0.25λ;

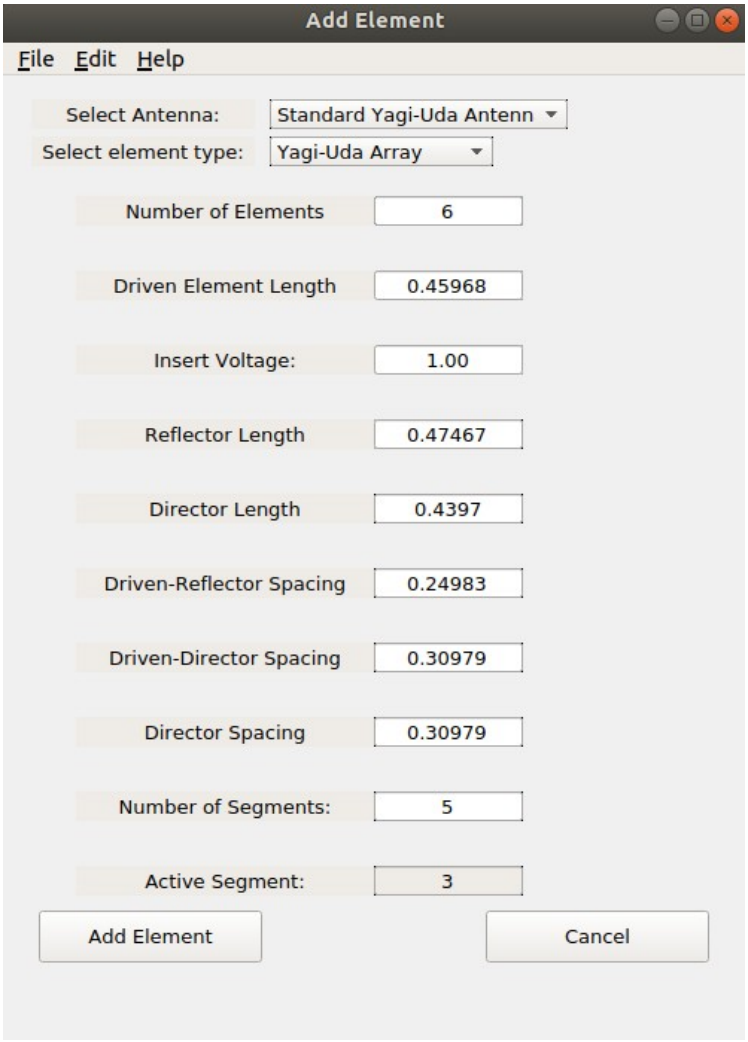

Figure B.1: Add element menu populated using the predefined antenna option

- Driven-Director Spacing = 0.31λ; and
- Director-Director spacing = 0.31λ.

The user can enter the number of elements and can also adjust any of the predetermined values if they wish to. The simulation was then set-up to provide 2-D representations of the far-field patterns produced by the antenna in both FEKO and NEC. Figure B.2 shows a graphical representation of the antenna.

Figures B.3 and B.4 show the input files generated by the program for FEKO and NEC respectively. These files are then passed to the appropriate simulation engines

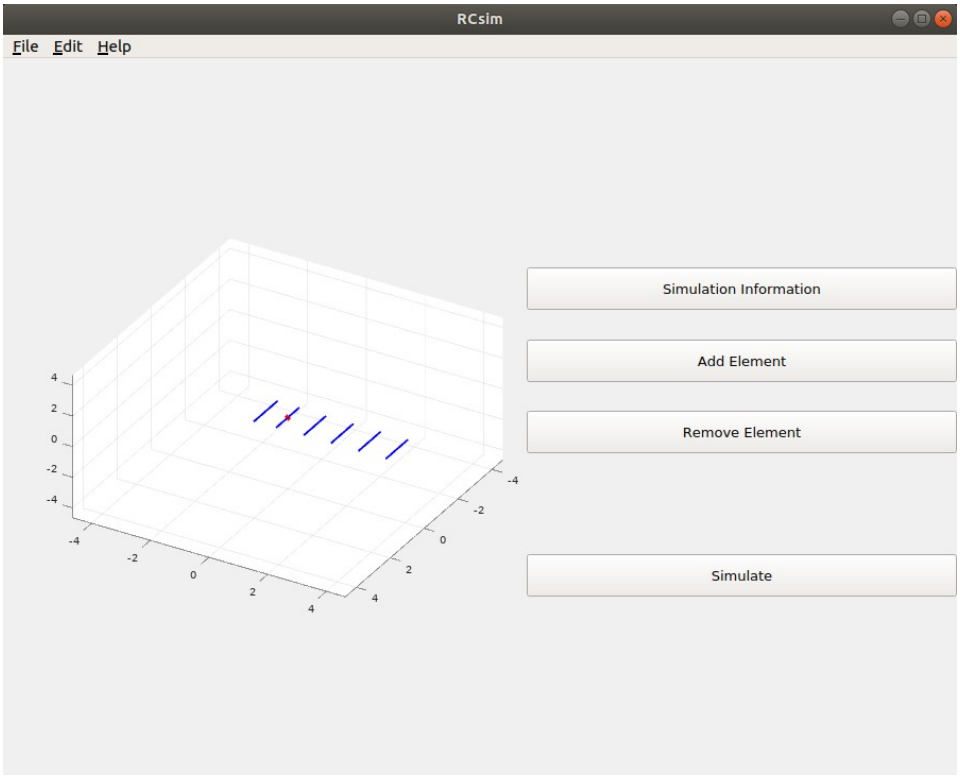

Figure B.2: Graphical representation of 6-element Yagi-Uda antenna

```
GW          1       5   -0.22984           0       0   0.22984           0       0   0.001
GW          2       5   -0.237335       -0.24983           0   0.237335       -0.24983           0   0.001
GW          3       5   -0.21985        0.30979            0   0.21985        0.30979            0   0.001
GW          4       5   -0.21985        0.61958            0   0.21985        0.61958            0   0.001
GW          5       5   -0.21985        0.92937            0   0.21985        0.92937            0   0.001
GW          6       5   -0.21985        1.23916            0   0.21985        1.23916            0   0.001
GE
FR          0       1       0       0     300       0
EX          0       1       3       0       1
RP          0       1     361    1000      90       0       1       1       0       0
EN
```

Figure B.3: NEC input file generated to simulate 6-element yagi uda antenna

and the results of these calculations are used to generate Figures B.5 and B.6. From the results show a small difference in the gain value between the FEKO and NEC simulations but both are marginally less the 12dBi which means that the results are as expected.

```
DP: P1 :  :  :  :  : -0.22984 :     0 :     0
DP: P2 :  :  :  :  : 0.22984 :     0 :     0
DP: S1 :  :  :  :  : -0.045968 :     0 :     0
DP: S2 :  :  :  :  : 0.045968 :     0 :     0
DP: P3 :  :  :  :  : -0.237335 : -0.24983 :     0
DP: P4 :  :  :  :  : 0.237335 : -0.24983 :     0
DP: P5 :  :  :  :  : -0.21985 : 0.30979 :     0
DP: P6 :  :  :  :  : 0.21985 : 0.30979 :     0
DP: P7 :  :  :  :  : -0.21985 : 0.61958 :     0
DP: P8 :  :  :  :  : 0.21985 : 0.61958 :     0
DP: P9 :  :  :  :  : -0.21985 : 0.92937 :     0
DP: P10 :  :  :  :  : 0.21985 : 0.92937 :     0
DP: P11 :  :  :  :  : -0.21985 : 1.23916 :     0
DP: P12 :  :  :  :  : 0.21985 : 1.23916 :     0
IP:  :  :  :  :  : 0.001 :  : 0.091937
BL: P1 : S1
BL: S2 : P2
BL: S1 : S2
IP:  :  :  :  :  : 0.001 :  : 0.094935
BL: P3 : P4
IP:  :  :  :  :  : 0.001 :  : 0.087941
BL: P5 : P6
IP:  :  :  :  :  : 0.001 :  : 0.087941
BL: P7 : P8
IP:  :  :  :  :  : 0.001 :  : 0.087941
BL: P9 : P10
IP:  :  :  :  :  : 0.001 :  : 0.087941
BL: P11 : P12
EG:     0 :     0 :     0 :  :  :  :  :  :  :  :  :     1

A1:     0 :    -1 :  :  :  :     1 :     0 :     0 :     0 :     0
FR:     1 :     0 :  :  :  :  300000000 :          0
FF:     1 :     1 :   361 :     0 :     0 :    90 :     0 :     1 :     1 :  :  :  :     0
EN
```

Figure B.4: FEKO input file generated to simulate 6-element yagi uda antenna

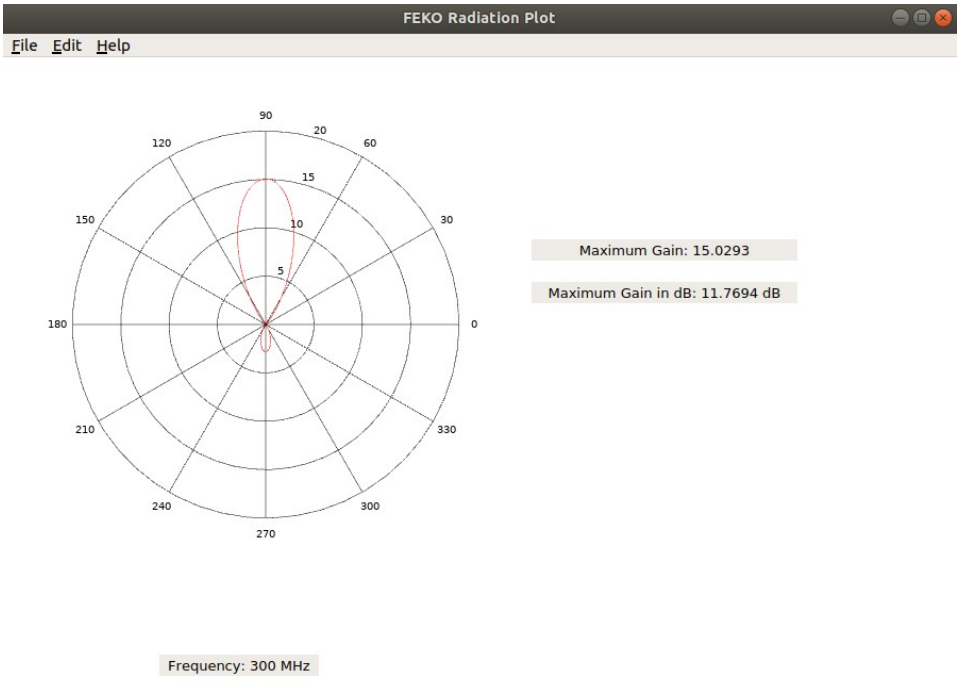

Figure B.5: 2-D far-field pattern generated using FEKO simualtion results for 6-element Yagi-Uda antenna

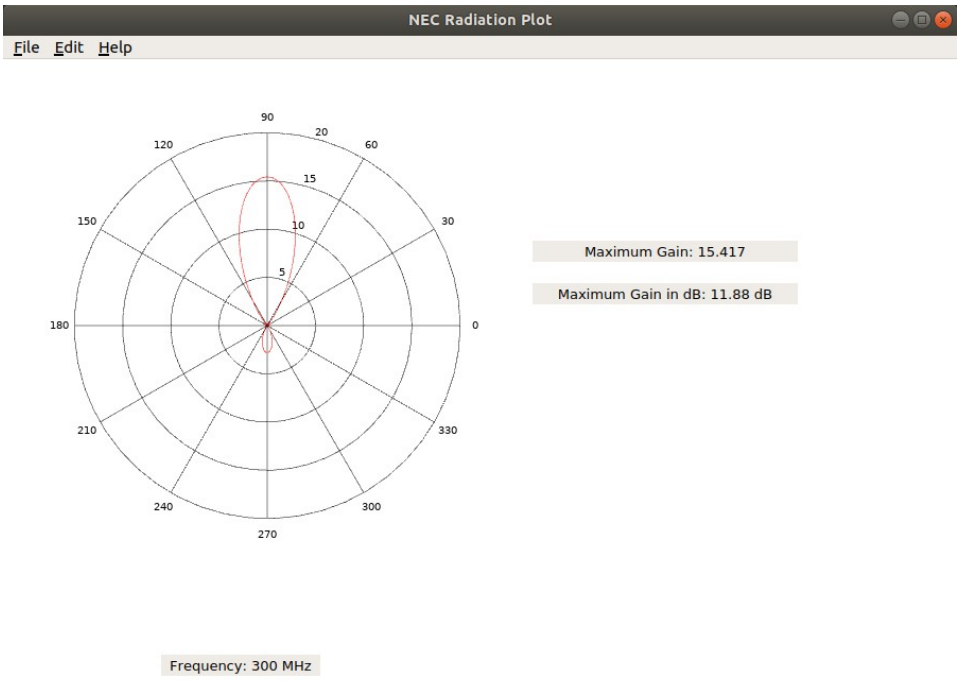

Figure B.6: 2-D far-field pattern generated using NEC simualtion results for 6-element Yagi-Uda antenna

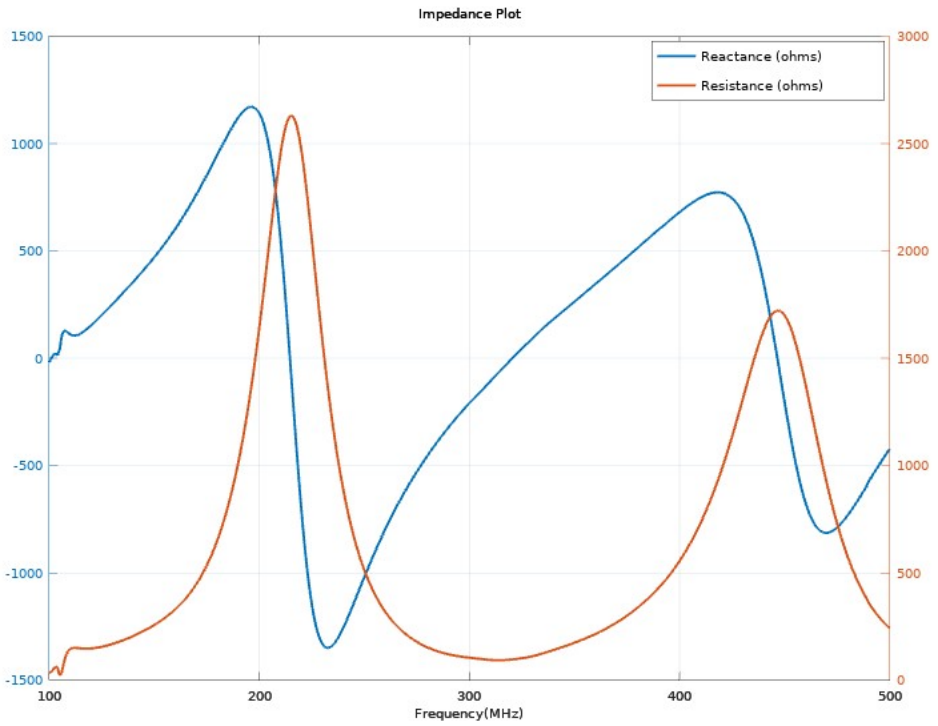

Figure B.7: Impedance plot of 6-element Yagi-Uda antenna designed for 300MHz centre frequency

B.2 Patch Antenna

The following example demonstarates how patch antennas are handled in the framework to produce the FEKO input files required for simulation. The shows that the patch that can be created is a basic, edge-fed, rectangular patch antenna. The example is for a 1GHz application.

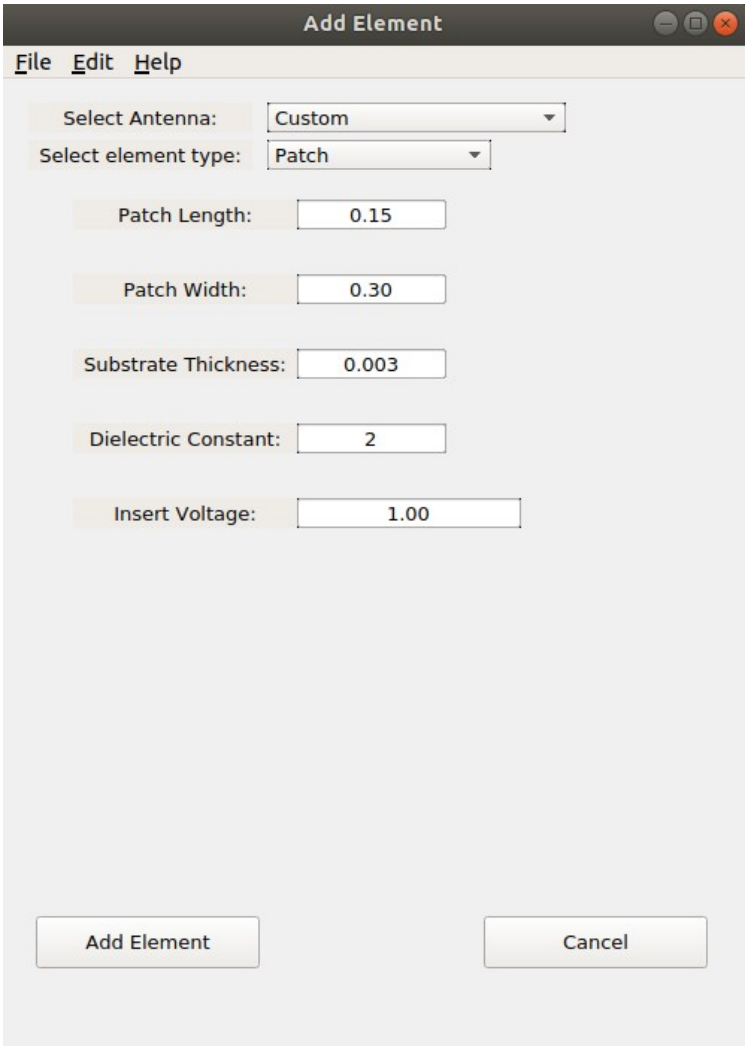

Figure B.8: "Add Element" menu with patch geometry fields populated

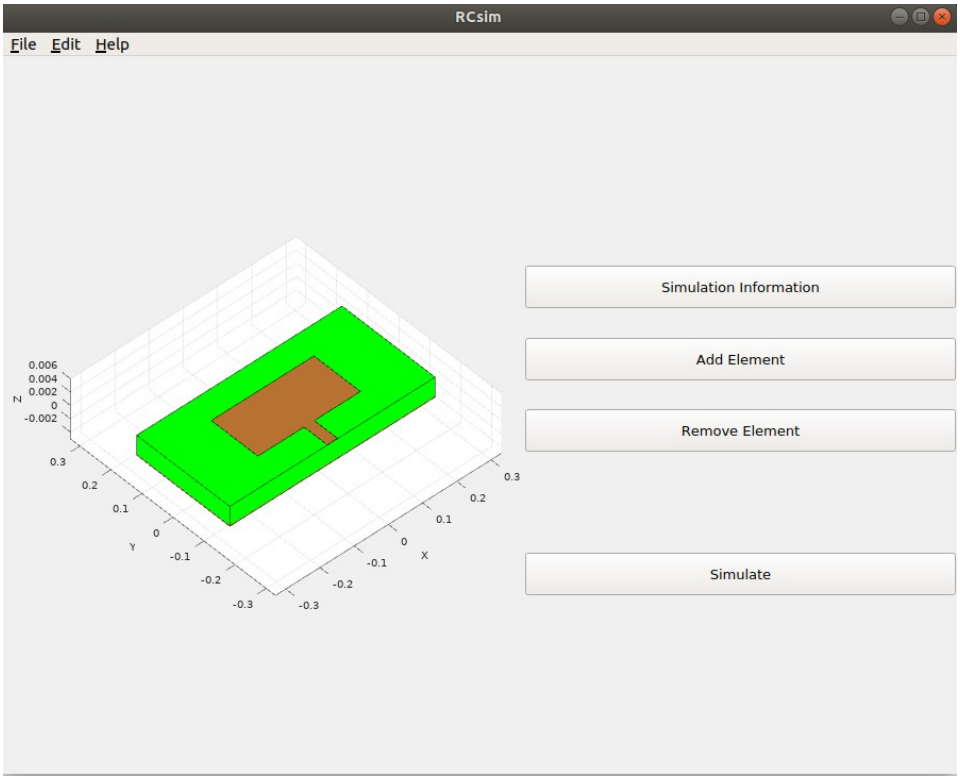

Figure B.9: Graphical representation of the patch antenna

```
DP: P1 :  :  :  :  : -0.15 : -0.075 : 0.003
DP: P2 :  :  :  :  : -0.045 : -0.075 : 0.003
DP: P3 :  :  :  :  : -0.045 : -0.1515 : 0.003
DP: P4 :  :  :  :  : 0.045 : -0.1515 : 0.003
DP: P5 :  :  :  :  : 0.045 : -0.075 : 0.003
DP: P6 :  :  :  :  :  0.15 : -0.075 : 0.003
DP: P7 :  :  :  :  :  0.15 : 0.075 : 0.003
DP: P8 :  :  :  :  : -0.15 : 0.075 : 0.003
DP: P9 :  :  :  :  :  -0.3 : -0.15 :     0
DP: P10 :  :  :  :  : -0.045 : -0.15 :     0
DP: P11 :  :  :  :  : -0.045 : -0.1515 :     0
DP: P12 :  :  :  :  : 0.045 : -0.1515 :     0
DP: P13 :  :  :  :  : 0.045 : -0.15 :     0
DP: P14 :  :  :  :  :   0.3 : -0.15 :     0
DP: P15 :  :  :  :  :   0.3 :  0.15 :     0
DP: P16 :  :  :  :  :  -0.3 :  0.15 :     0
DP: P17 :  :  :  :  : -0.045 : 0.1515 : 0.0015
DP: P18 :  :  :  :  : 0.045 : 0.1515 : 0.0015
DP: P19 :  :  :  :  :  -0.3 : -0.15 : 0.003
DP: P20 :  :  :  :  :  -0.3 :  0.15 : 0.003
DP: P21 :  :  :  :  :   0.3 : -0.15 : 0.003
DP: P22 :  :  :  :  :  -0.3 : -0.15 :     0
IP:  :  :  :  :  : 0.03
PM: P1 : P2 : P3 : P4 : : P5 : P6 : P7 : P8
PM: P9 : P10 : P11 : P12 : : P13 : P14 : P15 : P16
LA: 1
PM: P3 : P4 : P18 : P17
LA: 2
PM: P11 : P12 : P18 : P17
ME: test :  :  : 0
IP:  :  :  :  :  :  :  : 0.03
LA: 3
QU: P19 : P20 : P21 : P22 : 1
EG: 0 : 0 : 0 :  :  :  :  :  :  :  : 1
DI: test : 0 : -1 :  :  : 2 :  :  :  : 0
AE: 0 : 2 : 1 : 0 : 1 : 1 : 0 : 50
FR:     1 :     0 :  :  :  :  300000000 :          0
FF:     1 :     5 :     9 :     0 :     0 :     0 :     0 :     5 :     5 :  :  :  :     0
EN
```

Figure B.10: Input generated to be simulated in FEKO

As can be seen in Figures B.8, B.9 and B.10 the program is able to handle the geometric data of the patch antenna and use it to create the FEKO input file for simulation.