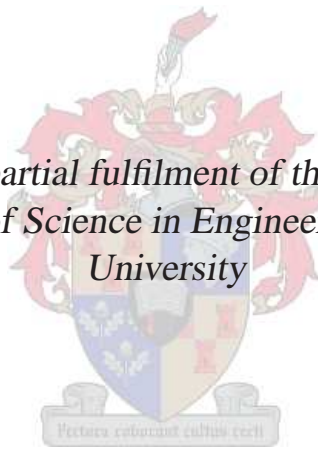


Efficient Numerical Analysis of Focal Plane Antennas for the SKA and the MeerKAT

by

Danie Ludick

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering at Stellenbosch University



Supervisor: Prof. David Bruce Davidson
Department of Electrical and Electronic Engineering

March 2010

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2010

Abstract

The use of Focal Plane Arrays (FPAs) as suitable feed-structures for the *Parabolic Dish Reflector antennas* that are intended to form a large part of the Square Kilometre Array (SKA) is currently the topic of conversation in various SKA research groups. The simulation of these structures however, relies on intensive computational resources, which can result in very long simulation runtimes - a serious problem for antenna designers. It was the purpose of the research to investigate efficient simulation techniques, based on the *Method of Moments* (MoM). In this thesis, the reader will be introduced to ways of improving FPA design by using resources such as High Performance Clusters, developing efficient MoM formulations for FPAs such as the Vivaldi antenna array and by developing efficient solution techniques for the resulting MoM equations by using techniques such as the *Characteristic Basis Function Method* (CBFM). In addition to the above mentioned methods, the concept of distributed computing is explored as a way to further aid the antenna designer in obtaining desired results in a reasonable time and with sufficient accuracy.

Opsomming

Die gebruik van Fokus Punt Samestellings (FPS) vir die voer van Paraboliese Skottel Antennas in die Square Kilometer Array (SKA), geniet tans baie aandag in verkeie navorsing-sirkels. Die analise van hierdie samestellings vereis egter intensiewe berekenings-infrastrukture, wat tot lang simulasies kan lei - 'n ernstige probleem vir antenna ontwerpers. Die doel van die skrywer se navorsing was om effektiewe simulatie metodes te ondersoek, gebaseer op die *Moment Metode*. In hierdie tesis, sal die leser bekendgestel word aan verskeie metodes om die ontwerp van Fokus Punt Samestellings doeltreffend te verrig; nl. die gebruik van Parallel Rekenaar Klusters, die ontwikkeling van effektiewe Moment Metode kode vir samestellings soos die Vivaldi antenna konfigurasie, asook die ontwikkeling van effektiewe oplos-metodes vir die matrikse wat deur die Moment Metode gelewer word, deur die sogenaamde *Karakteristieke Basis Funksie Metode* (KBFM) te gebruik. Hierby ingesluit word die konsep van *verspreide numeriese berekening* ondersoek, as 'n manier waarop die antenna ontwerper resultate binne 'n aanvaarbare tyd en akkuraatheid kan verkry.

Acknowledgements

I would like to express my sincerest gratitude toward the following people and organisations for their contribution to the success of this project.

- A big thank you to all my family and friends, especially my wife Sunel, for her support and patience and for drawing many of the illustrations included in this thesis.
- Prof. D.B. Davidson for introducing me to this exciting field of Computational Electromagnetics and for his excellent academic guidance throughout my University career
- The Centre for High Performance Computing (CHPC) in Cape Town, for providing me with the *iQudu* IBM cluster for the simulations.
- Kevin Colville from the CHPC for his advice regarding the application of distributed computing to numerical problems.
- EM Systems and Software (Pty) Ltd. for all the evaluation licenses we acquired for running FEKO in parallel on the iQudu.
- I would like to thank the FEKO support team (especially Mel van Rooyen) for their prompt replies on my queries regarding the installation and use of FEKO at the CHPC.
- The other CEMAGG members: A. Young and E. Lezar for their support and insights during my research
- The South African SKA Project Office for the financial support granted to me
- The University of Stellenbosch and specifically the Department of Electrical and Electronic Engineering for the outstanding quality of education they offered me

Contents

Declaration	i
Abstract	ii
Opsomming	iii
Acknowledgements	iv
Contents	v
List of Figures	viii
List of Tables	xi
Nomenclature	xii
1 Introduction	1
2 Numerical analysis of large electromagnetic structures	3
2.1 Overview of CEM techniques	3
2.2 High Performance Computing	4
2.3 The Fast Multipole Method (FMM)	8
2.4 Macro Domain Basis Function (MBF) Techniques	11
2.5 Conclusion	16
3 Using CEM Tools in a Parallel Computing Environment	19
3.1 Overview of the MoM algorithm	19
3.2 Parallelisation of the MoM algorithm in FEKO	21
3.3 The <i>iQudu</i> - a High Performance Computing (HPC) infrastructure	22
3.4 Runtime predictions	23
3.5 The simulated antenna-model	25
3.6 Benchmarking criteria	25
3.7 The effect of interconnects on simulation runtimes	26
3.8 Benchmarking results using the Infiniband interconnect	27
3.8.1 Speed-up versus the number of nodes	27
3.8.2 Efficiency versus the number of nodes	30
3.8.3 Efficiency versus <i>Grain-size</i>	31
3.8.4 Memory usage	31
3.9 Conclusions	32

4	A Method of Moments Formulation in 3D	34
4.1	The Rao-Wilton-Glisson (RWG) MoM formulation for arbitrary 3D scatterers and radiators	34
4.1.1	The electric field integral equation (EFIE)	35
4.1.2	The RWG basis-functions	36
4.1.3	The Galerkin testing procedure	38
4.1.4	Derivation of the MoM matrix equation	39
4.1.5	Numerical evaluation of the MoM matrix elements	40
4.2	A practical implementation of the RWG MoM formulation, viz. <i>GMoM</i>	40
4.2.1	Programming considerations	40
4.2.2	Geometry setup	42
4.2.3	Matrix equation setup	44
4.2.4	Solving the MoM matrix equation	46
4.2.5	Post-processing	48
4.3	Figure of merit used for evaluating <i>GMoM</i>	50
4.4	Applying <i>GMoM</i> to a PEC scatterer	51
4.5	Applying <i>GMoM</i> to a single Vivaldi radiator	53
4.5.1	Modelling the antenna feed	53
4.5.2	Results	55
4.6	Applying <i>GMoM</i> to an interconnected dipole antenna array	57
4.6.1	Modelling element interactions	57
4.6.2	Results	59
4.7	Conclusions	60
5	The CBFM approach for solving the MoM matrix equation	62
5.1	Applying the CBFM to an interconnected FPA	62
5.1.1	Overview of the CBFM approach	63
5.1.2	Geometry setup	64
5.1.3	Generating "primary" CBFs	64
5.1.4	Generating "secondary" CBFs	67
5.1.5	Generating and solving the reduced matrix equation	68
5.2	Numerical Results	68
5.2.1	A linearly polarised 3×1 Vivaldi array	68
5.2.2	A linearly polarised 7×1 Vivaldi array	70
5.3	Parallelisation of the CBFM	72
5.3.1	Geometry Setup - from a domain decomposition point of view	72
5.3.2	Calculating the "primary" CBFs	73
5.3.3	Calculating the "secondary" CBFs	73
5.3.4	Generating the reduced matrix equation	74
5.3.5	CBFM parallelisation results	75
5.4	Conclusions	76
6	General Conclusions	77
A	Appendix A - Thin Strip MoM model of a Dipole antenna	79
A.1	The antenna model	79
A.2	Simulation results	80
A.3	Conclusion	81

- B Appendix B - Code Listing for the Ping-pong test** **82**
- B.0.1 C implementation of the Ping-pong test 82
- B.0.2 Octave implementation of the Ping-pong test 83
- B.0.3 Python implementation of the Ping-pong test 84

- References** **85**

List of Figures

2.1	Run-times associates with the LU-decomposition algorithm simulated on HPC architectures capable of sustaining 1 megaflop and 1 petaflop respectively (adapted from [1]).	6
2.2	A large electromagnetic scatterer discretized with RWG basis functions, included as an example model in the FEKO package. (Reprinted with permission from EM Systems and Software SA (Pty) Ltd.).	8
2.3	A comparison between the runtime associated with LU-decomposition, a fast-converging iterative solver and the MLFMA (adapted from [1]).	10
2.4	A comparison between the memory requirement associated with the conventional MoM and the MLFMA respectively (adapted from [1]).	10
2.5	A PEC scatterer consisting of 4 PEC plates, discretized with RWG basis functions.	12
2.6	A comparison between the runtime associated with LU-decomposition, an iterative solver, the MLFMA and the CBFM formulation (adapted from [1]).	17
2.7	A comparison between the memory requirement associated with the conventional MoM, the MLFMA and the CBFM respectively (adapted from [1]).	17
3.1	Example of an electromagnetic scatterer, with a discretized surface. The figure was obtained from POSTFEKO, part of the FEKO package.	20
3.2	Schematic overview of the sequential MoM algorithm	20
3.3	The iQudu Cluster with 160 computing nodes, housed by the CHPC in Cape Town.	23
3.4	Graphical illustration of the iQudu infrastructure.	23
3.5	Approximate runtimes for LU decomposition on the iQudu cluster with 160 nodes and 1 node respectively.	24
3.6	Example of a single Vivaldi element and a 9 element Vivaldi array modelled with FEKO.	25
3.7	Absolute run-times of a 32 element Vivaldi Array (17,472 unknowns) simulated on various number of nodes by using a 1 Gbit Ethernet and a 10 GBit Infiniband interconnect.	27
3.8	Run-time speedup vs. the number of nodes, as measured on the iQudu cluster when using the Infiniband interconnect	28
3.9	The average transmission time in seconds, associated with messages of arbitrary size during a Ping-pong test implemented with various MPI bindings using a GBit Ethernet interconnect.	29
3.10	Run-time efficiency vs. the number of nodes, as measured on the iQudu cluster when using the Infiniband interconnect	30
3.11	Run-time efficiency vs. the number of unknowns divided by the number of nodes (<i>grain-size</i>), as measured on the iQudu cluster when using the Infiniband interconnect	31

3.12	Calculated- and recorded memory usage for various number of unknowns associated with the MoM implementation in FEKO.	32
4.1	Example of an (a) open and (b) closed PEC structure, modelled with triangular patches	34
4.2	The Rao-Wilton-Glisson (RWG) basis function [2]. The diagram illustrates a triangle pair forming a surface that shares an internal (i.e. non-boundary) edge.	37
4.3	Schematic overview of the general GMoM algorithm, a practical implementation of the MoM that incorporates the use of RWG basis functions.	41
4.4	Example of triangulation schemes obtained by (a) Matlab and (b) Gmsh respectively	42
4.5	Simplex coordinates and edges illustrated on a triangular subdomain	45
4.6	Simplex coordinates illustrated on a triangular subdomain for various rotations. The centre point of the triangle is included for reference (depicted by the square).	46
4.7	Illustration of a short-dipole modelled between the centroids of two triangles, T^+ and T^- . respectively	48
4.8	Digitization scheme of the pattern in spherical coordinates	51
4.9	A square $\lambda \times \lambda$ PEC plate, discretized with triangular patches conforming to a mesh density of roughly $\lambda/16$. The two principal cuts are illustrated as $(A - A')$ and $(B - B')$ respectively.	52
4.10	The distribution of the dominant current components on a square $\lambda \times \lambda$ PEC plate	52
4.11	Surface current distribution calculated on a $\lambda \times \lambda$ PEC plate with (a) GMoM and (b) FEKO respectively.	53
4.12	The feeding edge model associated with a driving edge, l_m	54
4.13	(a) A Vivaldi radiator, discretized with triangular patches of mesh-densities varying between $\lambda/20$ in the feed-region, to $\lambda/10$ at the outer edges of the structure. (b) The feeding edge location	55
4.14	Comparing the input reflection for a Vivaldi antenna over a frequency range of 1 GHz to 3 GHz, calculated with GMoM and FEKO respectively	56
4.15	Comparing the E and H-plane directivity patterns for a Vivaldi antenna at a frequency of 2 GHz, calculated with GMoM and FEKO respectively	57
4.16	Comparing the normalised surface current distribution for a Vivaldi antenna at a frequency of 2 GHz, calculated with (a) GMoM and (b) FEKO respectively.	57
4.17	An extension of the feeding-edge model between two triangles, that includes a source impedance, Z_g	58
4.18	Comparing the geometry of a 2×2 dipole array configuration created in FEKO and GMoM respectively.	60
4.19	Comparing the S-parameters for a 2×2 dipole array configuration simulated with FEKO and GMoM respectively	60
4.20	Comparing the directivity patterns for a 2×2 dipole array configuration simulated with FEKO and GMoM respectively. All the ports are excited with $V_s = 1$ V	61
5.1	A 3×1 Vivaldi array depicted in terms of CBF subdomains p and q respectively.	62
5.2	An overview of the general CBFM approach	64
5.3	The discretization of a 7×1 Vivaldi array constructed by copying and translating one meshed element to its location in the array.	65
5.4	The extraction of sub-arrays from a 7×1 Vivaldi array for the construction of "primary" CBFs.	65
5.5	The windowing of the CBFM sub-arrays for a linearly polarised Vivaldi array.	66

5.6	Mapping the windowed "primary" CBFs to the corresponding element positions in a linearly polarised Vivaldi array.	67
5.7	Generating the "secondary" CBFs for sub-array q by considering the primary CBFs on domain p for a linearly polarised Vivaldi array.	67
5.8	The surface current distribution on a 3×1 Vivaldi array calculated with (a) a direct solution technique and (b) the CBFM approach	69
5.9	Comparing the E and H-plane directivity patterns for a 3×1 Vivaldi antenna at a frequency of 2 GHz, calculated with the CBFM and a direct solver.	69
5.10	The surface current distribution on a 7×1 Vivaldi array calculated with (a) a direct solution technique and (b) the CBFM approach.	70
5.11	Comparing the E and H-plane directivity patterns for a 7×1 Vivaldi antenna at a frequency of 2 GHz, calculated with the CBFM and a direct solver.	71
5.12	Comparing the runtime speed-up and efficiency obtained by mpi4py and MPITB when applied to a general CBFM formulation	76
A.1	The geometry of a half-wavelength dipole antenna modelled with (a) a thin strip discretized with RWG basis functions constructed in GMoM and (b) a wire-model constructed in FEKO.	79
A.2	Magnitude of the input reflection coefficient (S_{11}) in dB calculated with the RWG model (GMoM) and FEKO respectively.	80
A.3	Phase of the input reflection coefficient (S_{11}) in degrees calculated with the RWG model (GMoM) and FEKO respectively.	81
A.4	Directivity pattern in dB, calculated with the RWG model (GMoM) (-) and FEKO (- -) respectively, at $f_c = 2$ GHz.	81

List of Tables

3.1	Number of unknowns associated with each of the simulated Vivaldi antenna arrays.	25
3.2	Measured bandwidth and latency for various MPI implementations.	29
4.1	The number of sampling points, N , for a symmetric Gaussian quadrature rule of degree P	46
4.2	The normalised error percentage obtained by GMoM when compared to FEKO for various quantities calculated for a Vivaldi antenna	56
4.3	The normalised error percentage obtained by GMoM when compared to FEKO for various parameters calculated for a 2×2 dipole antenna array	61
5.1	The normalised error percentage obtained by GMoM when calculating various quantities for a 3×1 Vivaldi antenna with the CBFM and direct solution techniques.	70
5.2	The normalised error percentage obtained by GMoM when calculating various quantities for a 7×1 Vivaldi antenna with the CBFM and direct solution techniques.	71
5.3	The solution run-times obtained by the CBFM and direct solution techniques when calculating the surface current distribution for a 7×1 Vivaldi antenna at a centre frequency of 2 GHz. The results are generated on an Intel Core2 Duo 2.8 GHz processor equipped with 4 GBytes of RAM.	71
A.1	Dimensions associated with the half-wavelength dipole models	80

Nomenclature

Acronyms

SKA	Square Kilometre Array
FPA	Focal Plane Array
CSIR	Centre for Scientific and Industrial Research
CBFM	Characteristic Basis Function Method
CBFs	Characteristic Basis Functions
KAT	Karoo Array Telescope
CPU	Central Processing Unit
MoM	The Method of Moments
HPC	High Performance Computing
FLOP	Floating Point Operations
FLOPS	Floating Point Operations per second
RAM	Random Access Memory
PEC	Perfect Electric Conductor
RF	Radio Frequency
CG	Conjugate Gradient Method
GMRES	Generalised Minimal Residual Method
EFIE	Electric Field Integral Equation
RWG	Rao-Wilton-Glisson
RFI	Radio Frequency Interference

Chapter 1

Introduction

It is a commonly-held viewpoint that for any science to mature, the power of its instrumentation must improve almost exponentially with time [3]. *Radio Astronomy* is such a scientific discipline that is currently being limited by the available technology. Many of the operating radio-telescopes are between 10 and 30 years old, and this limits the number of discoveries made in this science. In 1991 the concept of the Square Kilometre Array (SKA) was born to address these issues.

It is intended that the SKA¹ will provide Radio Astronomers with a receiving aperture of more than a million square meters. It will consist of thousands of antennas that will be able to combine their individual images² to form a single big radio image of the universe. It is envisaged that the SKA will focus on various key science cases, such as exploring the formation of planets and stars when the universe was still in a gaseous form (the so called *dark-ages*).

To ensure the success of the various science cases that is to be conducted with the SKA, its core needs to be situated in a remote location. The remote location will ensure that the receivers are largely isolated from *Radio Frequency Interference* (RFI) such as unwanted cell-phone and television emissions. In 2006 it was decided that only two countries will be short-listed to host this enormous Radio Telescope, viz. Australia and South Africa. With the final decision to be made provisionally at the end of 2011, both countries set out to build SKA technology demonstrators - *ASKAP* [4] in Australia and *MeerKAT* in South Africa [3].

Many of the antennas that will form part of the SKA (and also the MeerKAT) will be of the *Parabolic Dish Reflector* type. Briefly, this type of antenna reflects incoming power to a spot known as the *focal plane*. A secondary antenna, known as the *feed-structure* then collects this energy from where it is transported to the *receiver*. The receiver then amplifies, digitises and transforms the incoming power density into a radio "image".

Two types of focal plane feed-structures are currently being researched in the SKA project, namely that of *sparse multiple feed-clusters* (such as horn clusters) and also *focal plane phased array techniques* (referred to as focal plane arrays hereafter). Of the two, the former is most certainly the more trusted and tested technology and is therefore frequently encountered as the feed-elements of large dish reflector antennas. A disadvantage of this type of feed-structure, however, is that the cluster elements are quite large. Only a small number of elements can therefore be placed in the focal region (to avoid the blocking of incoming signals). The effect of this, is that the field of view illuminated by the dish reflector is not sampled very efficiently and directly contributes to very long survey times.

Phased array feed-structures on the other hand, consist of multiple, densely packed (and

¹The information in this section is based on that found at [3]

²In Radio Astronomy, this is termed *Interferometry*

somewhat smaller) antenna-elements. Numerous beams (that can be scanned in any given direction) therefore sample the focal plane much more efficiently than that formed by sparse feed-clusters³. A major challenge related to this technology however is that the design of these systems depends heavily on iterative software simulations. The necessary computational power is frequently a limiting factor, depending on the complexity of the electromagnetic structure such as the antenna-element (especially the *electrical* size thereof) that is used in the phased array designs. In a study conducted by the author in 2007 [6], it was found that electrically large phased array simulations consisting of *Vivaldi* antennas can take up to several hours or even days to complete, thereby severely limiting the problems that can be investigated.

The purpose of the author's MScEng research was to investigate and develop efficient simulation techniques for electrically large FPA structures that may be used as feed-structures for parabolic reflector antennas in the SKA and the MeerKAT.

The research includes the use of a *high performance parallel computing* (HPC) cluster housed at the Centre for High Performance Computing (CHPC) in Cape Town. In this part of the study, the commercially available electromagnetic software package, FEKO⁴, was used to simulate various sized Vivaldi FPAs. The simulation data, such as runtimes, memory usage, etc. was then benchmarked and used to determine the performance of the parallel *Method of Moment* (MoM) solver implemented in FEKO. With the capabilities of FEKO and the CHPC infrastructure recorded, the author set out to develop his own *Computational Electromagnetic* (CEM) formulations for FPA structures, such as the well-known Vivaldi antenna array. The formulation is based on the MoM and incorporates the *Characteristic Basis Function Method* (CBFM) to solve the dense matrix equation resulting from the MoM formulation. The author will refer to his solver as *GMoM* in the text.

The thesis is structured as follows: In Chapter 2, the reader is presented with a discussion on the various well-known computational electromagnetic techniques available for the simulation of electrically large problems. In Chapter 3, the results of a detailed investigation of one of these techniques, viz. that of applying parallelization techniques to numerical algorithms in a high performance computing (HPC) environment, will be presented. In Chapter 4, the focus shifts towards the author's MoM formulation (GMoM) for 3-dimensional electromagnetic structures, as the first step towards *developing and improving* the simulation tools needed for FPA research. In Chapter 5, the author will discuss the *Characteristic Basis Function Method* (CBFM), as a way of solving matrix equations efficiently and broadening the *scope* of FPA problems that can be considered when using the MoM. The thesis is then concluded in Chapter 6 with a summary and recommendations for future research.

³For a more detailed comparison between the two FPA architectures, the reader is referred to [5]

⁴FEKO is the flagship product of EM Systems and Software South-Africa Pty Ltd. For more information regarding FEKO, please visit [7]

Chapter 2

Numerical analysis of large electromagnetic structures

This Chapter presents an overview of the CEM simulation techniques that are suited to the analysis of large electromagnetic structures. Emphasis will be placed on various methods that extend the capabilities of the MoM formulation such as the use of high performance clusters (i.e. parallel computing), the revolutionary fast multipole method (FMM) and its extension, the multilevel fast multipole algorithm (MLFMA), as well as macro-domain basis function techniques such as the characteristic basis function method (CBFM). The author will also compare these techniques in a quantitative manner by using suitable examples and results obtained from the literature.

Before focussing on the details of each of the above approaches, a general overview of well-known CEM techniques is first presented in the following Section.

2.1 Overview of CEM techniques

Computational electromagnetics (CEM), i.e. the numerical approximation of Maxwell's equations has provided a powerful basis for the development of various disciplines, such as radio-frequency, microwave, and wireless engineering. CEM is a multi-disciplinary field with its underlying principles being electromagnetic theory, numerical methods, geometric modelling, computer science and algorithms. The application of CEM is far-reaching and includes antennas, wireless communication, radar as well as providing means to investigate biological EM effects [1].

Briefly, in the field of CEM, one can distinguish between so called *full-wave* methods, also known as *low-frequency* methods, and asymptotically *high-frequency* methods. The full-wave CEM techniques include well-known methods such as finite difference time domain (FDTD) method and frequency domain-methods such as the method of moments (MoM) and the finite element method (FEM)¹. The accuracy of each of the aforementioned methods depends somewhat on the problem at hand. The high frequency methods, i.e. physical optics (PO), geometrical optics (GO) and the uniform theory of diffraction (UTD) all require fundamental approximations in the Maxwell equations, the validity of which increases asymptotically with frequency. These techniques are thereby limited to a very specific group of problems. In this thesis the emphasis is placed on full-wave methods; the reason for which is expressed in [1],

¹It is to be noted that time-domain formulations does exist for the MoM and the FEM, although it is mainly used for specialised applications.

namely that the limitations of these methods are continuously being extended with technological developments especially in the high performance computing sector. The limitations in the high-frequency techniques are fundamental and as mentioned above severely problem dependent. The remainder of this section is concerned with some of the widely-used full-wave techniques in computational electromagnetics.

Central to the full-wave techniques is the concept of discretizing an unknown quantity. In the case of the MoM, this is the surface current. For the standard FEM, the unknown quantity is the E-field² and for the FDTD, the E and H-field. This discretization-process is also known as *meshing* and entails that the structure's geometry be divided into a number of small elements. The elements that are used may vary from one-dimensional segments, two-dimensional triangular surface elements, three-dimensional tetrahedral elements or a three-dimensional grid, depending on the geometry being modelled and the applied formulation. A so-called *basis-function* is then related to each³ of the elements, that defines a simple approximation for the spatial variation of the unknown quantity [1]. It is therefore intuitive that the accuracy by which the unknown quantity can be modelled, is related to the level of discretization. In general, a finer discretization leads to a better accuracy. This however poses a problem as a finer mesh size places a bigger burden on the computational resource in terms of storage and numerical computation. This is also a critical problem related to electrically large structures. Consider for example the computational cost associated with the widely used MoM formulation:

If we let the number of basis-functions⁴ be N_{RWG} the memory requirements of the algorithm is of $\mathcal{O}(N_{RWG}^2)$ and the computational runtime of $\mathcal{O}(N_{RWG}^3)$. The MoM is typically requires a discretization size ranging from $\lambda/10$ to $\lambda/20$ when using the RWG type basis functions for a reasonable accuracy. As the discretization becomes finer, or alternatively the structure becomes electrically large, this cost can have a severe impact on the analysis if the architecture used for the simulation is insufficient in terms of on-board memory and processing power. One way of addressing this issue, is by using a high performance computing infrastructure, i.e. a parallel computing environment. This is typically a *divide and conquer* strategy in which the problem is subdivided into a number of smaller sub-problems, each of which is distributed to one of a number of compute elements so that they can be solved concurrently [8]. Subdividing the problem in this manner then also distributes the computational burden of the algorithm in terms of both runtime and memory usage, thereby allowing one to consider larger problem domains. Other means of addressing the issues related to the numerical analysis of electrically large and complex structures are based on approximation techniques such as the MLFMA and macro-domain basis function methods such as the CBFM. Before investigating each of these, it is worth discussing the important role that the high performance computing sector is playing with regards to CEM modelling and analysis.

2.2 High Performance Computing

The general computing sector underwent a significant change in terms of hardware in the mid 1960s when there was a wide-spread conversion from vacuum tube to solid state transistors. Research in this sector eventually led to the development of the integrated circuit in the late 1950s and the design of the microprocessor by Intel in the 1970s. As the transistor density on

²In some cases, the discretization of the H-field is also used

³In most cases the basis function is related to a common quantity between interconnected of elements, such as the shared edge between two triangular patches in the case of the MoM.

⁴The *RWG* subscript refer to the conventional Rao-Wilton-Glisson basis functions [2]. A more detailed discussion of the MoM will be presented in Chapter 4.

an integrated circuit eventually increased, closely following Moore's law⁵, the processing-speed (and on-board memory) associated with computers increased quite rapidly. A technique known as *frequency scaling* [10] was the driving factor behind the improvement in the processing power of processors between the mid 1980s and 2004. Frequency scaling or ramping is based on the observation that a program's runtime is a function of the number of instructions that need to be performed multiplied by the tempo at which these instructions can be executed, i.e. the number of instructions per clock-cycle. By increasing the clock-frequency, one reduces the cycle time and ultimately the average runtime required to execute the program.

Unfortunately the average power consumption of an integrated circuit containing fundamentally CMOS inverter blocks [11] can be expressed as,

$$P_{av} \approx CV^2f \quad (2.1)$$

where C is the switching capacitance, V the applied voltage and f the frequency.

It is evident that as the frequency increases, so does the average power consumption of the processor. This presented a significant problem in terms of cooling, cost and the ongoing increase in processing power. The solution to this problem presented itself in the form of parallel processing, i.e. instead of increasing the rate at which a processor can process data, simply do more operations at the same time [1]. This is the philosophy that contributed to the arrival of multi-core technology.

Historically, parallel processing has been applied in a variety of ways since the 1970s such as *pipelining* taken by early vector super-computers such as the CRAY machines. Pipelining works on the basis that certain sections of an operation can be overlapped in time, and thereby performed concurrently [1]. In the early 1970s, a classification system for computer architectures called *Flynn's taxonomy* [12] was devised. The two widely-used architecture classifications in this taxonomy applicable to parallel computing are that of single instruction multiple data (SIMD) and multiple instruction multiple data (MIMD) systems⁶. A SIMD architecture described a computing system where the same operation is performed on multiple data. An example of such a computing architecture is a graphics processing unit (GPU) typically used in the gaming industry⁷. MIMD machines described computer systems consisting of a number of nodes, each with at least a processing element, operating independently on its own local instruction stream and data [1]. These MIMD-SIMD classifications had recently made way for more general classifications in the HPC sector (specifically due to a combination of the distributed and shared memory architectures). This taxonomy includes classifications such as symmetric multiprocessors (SMPs), massively parallel processors (MPPs) and distributed processing environments. In the SMP systems, a number of processors essentially share the same address space and are connected with extremely fast on-board interconnects with caching enhancing the performance of memory access. In an MPP architecture, a large number of processors typically access distributed memory. This is similar to that of a distributed processing environment with the difference lying in the fact that the latter typically consist of heterogeneous nodes connected by slower interconnects. It is however to be noted that modern day HPC systems such as the Cray XT5 combines SMP and MPP paradigms, since it also contains a globally addressable

⁵Moore's law is attributed to Intel co-founder Gordon. E. Moore. This law is an empirical observation that the transistor density on a micro-processor doubles roughly every 18 to 24 months [9].

⁶Flynn's taxonomy also included two other architecture classifications, viz. single instruction single data (SISD), and multiple instruction single data (MISD). These are however not frequently used in the parallel computing sector.

⁷At the time of writing the use of GPUs in the scientific sector, especially that of CEM, was also starting to draw much attention [13].

memory subsystem [1].

The MPP architecture, frequently incorporating SMP nodes, is a very typical HPC architecture encountered in both the industry and research institutions. An additional factor that impacts the performance of such a system is the interconnect being used. In the Chapter 3 simulation results obtained from using a GBit Ethernet and a 10 GBit Infiniband interconnect respectively will be compared. There it will be shown that the lower bandwidth associated with the slower interconnect can degrade the runtime performance of a simulation.

At this point, it will be useful to illustrate the improvement in runtime⁸ one can obtain on some of the HPC architectures for a frequently encountered algorithm in the field of CEM, namely that of *LU-decomposition*. If on-board memory permits, LU-decomposition can be used to solve the matrix equation that is the result of applying CEM techniques such as the MoM to an electromagnetic problem. The operation count associated with this factorization technique is approximately $O(N_{RWG}^3)$ when considering that the matrix entries are stored as complex valued numbers [14]. In the 1980s the maximum performance of the available HPC systems was in the order of a megaflop [1]⁹. As cluster computing became more popular, advancements in this field has attributed to the petaflop barrier being reached by the IBM BladeCenter QS22/LS21 Cluster (known as the Roadrunner) by the end of 2008 [15]. In Figure (2.1) we consider the runtimes for the LU-decomposition algorithm on a 1 megaflop and 1 petaflop architecture respectively. Comparing the runtimes associated with the two architectures for a problem size of 100,000 unknowns, we see that the runtime has decreased from roughly a century to a few seconds - quite a significant improvement.

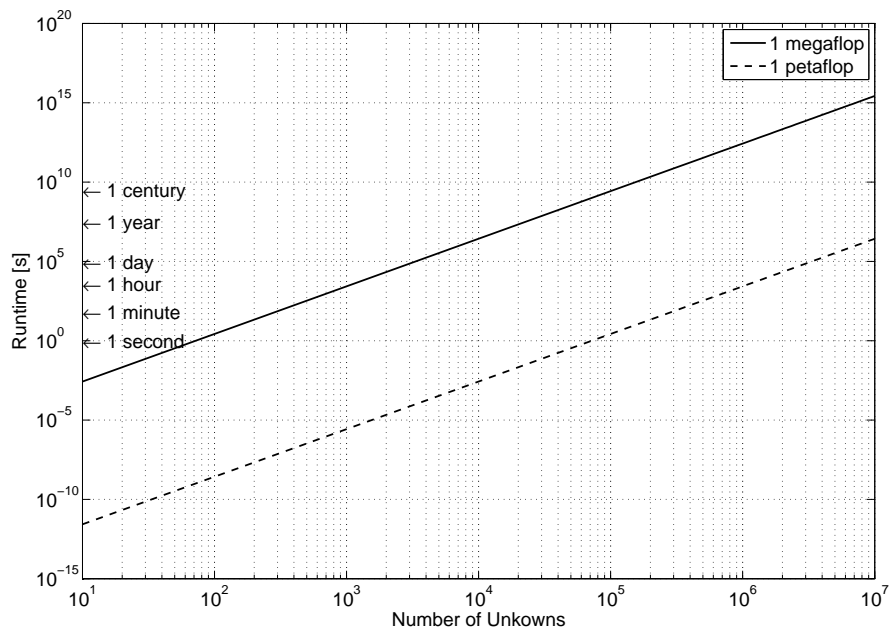


Figure 2.1: Run-times associates with the LU-decomposition algorithm simulated on HPC architectures capable of sustaining 1 megaflop and 1 petaflop respectively (adapted from [1]).

⁸In this context *runtime* actually refers to CPU-time, i.e. the time that the algorithm was allocated to execute on the processing unit's CPU.

⁹In computer terms, *flops* is an acronym for *floating point operations per second*. This is a measure of the computer- or computer-cluster's performance, that makes heavy use of floating point calculations. When working with high performance computer-architecture it is however necessary to introduce larger units than the flops, such as the teraflop that is equal to one trillion- or 10^{12} flops

With the fundamentals of the HPC sector covered in terms of general classifications and processing capabilities, it is important to overview the tools that are available for developing algorithms in a parallel computing environment, namely that of *distributed* and *shared memory* programming models. The distributed programming models are characterised by the fact that each process accesses an address-space associated only with that process. Data-communication between processes is realised through message-passing schemes. Incorporating such a message-passing paradigm into programs proved a significant challenge in the early days of parallel computing, the result being that everyone used their own hardware dependent method of ensuring interprocess communication. In the early 1990s standards such as the message-passing interface (MPI) and parallel virtual machine (PVM) emerged which provided standardized high-level communication libraries to establish interprocess communication. Various bindings or implementations of these standard interfaces exist and are typically written in compiled languages such as FORTRAN and C/C++. However, as the prototyping of complex algorithms becomes more demanding in terms of user input, debugging and error checking, these interfaces are also becoming available for interpreted programming languages such as Python, Matlab and Octave. In Chapter 3, Section 3.8.1, the author will illustrate that the speed-up and efficiency obtained by these bindings closely resemble those of the C implementation.

The other programming model for *shared memory* architectures, such as SMPs, consists of multiple *threads* of the algorithm accessing a shared memory bank. Little or no interprocess communication is required between the threads as all the data is essentially shared. This programming model thereby does not suffer from the high latency associated with bandwidth-limited interconnects as is the case with message-passing paradigms. A disadvantage of this technique however, is that all the processes require access to the same memory bank thereby limiting the data-sets that can be evaluated. As with MPI and PVM, the shared memory model has also been standardised with interfaces such as OpenMP and Pthreads¹⁰.

Recently, *hybrid* techniques are emerging that combine the advantages of both the distributed and shared memory programming models. The reason behind this is that HPC clusters mainly consist of various multi-processor compute nodes such as SMPs which are connected by some sort of high-speed interconnect. In essence, hybrid techniques entails that shared memory models be used on the SMP architectures. When interprocess communication is required between processes residing on physically separated compute nodes, the data will be passed with message-passing interface routines. Techniques such as those mentioned in this and the previous paragraphs are ensuring that the parallelisation of the sequential implementation of an algorithm is a less formidable task than it was only a decade ago.

Although HPC is becoming more freely available to the scientific sector, it is still necessary to improve the CEM techniques by applying efficient approximations to the underlying full-wave formulations with the goal of further reducing the computational cost associated with large electromagnetic structures. This will be the focus of the following two sections. It is to be noted that the following techniques are extensions to the MoM formulation. The reason for this being that the MoM formulation is specifically well suited to radiating or scattering problems consisting of highly conducting material as it explicitly incorporates the so-called "radiation condition" - i.e. the correct behaviour of the field far from the source [1] - an important characteristic in the context of the SKA and the MeerKAT project.

¹⁰It is to be noted that shared memory programming with threads is typically done with compiler directives and is largely associated with languages such as FORTRAN, C and C++

2.3 The Fast Multipole Method (FMM)

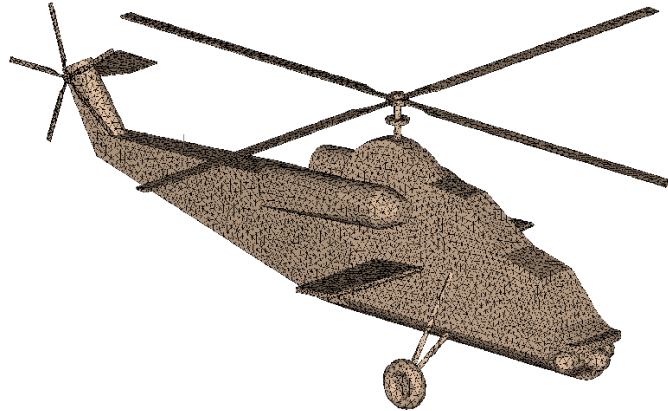


Figure 2.2: A large electromagnetic scatterer discretized with RWG basis functions, included as an example model in the FEKO package. (Reprinted with permission from EM Systems and Software SA (Pty) Ltd.).

Although HPC provides a means of conducting numerous numerical tasks in a fraction of the time due to the distribution of the problem amongst various compute nodes, HPC resources was not always as accessible in the past as they are today. This was a contributing factor to the development of so called *fast techniques* that were formulated to enable the simulation of large CEM problems consisting of thousands of unknowns, such as the model illustrated in Figure 2.2. One of these methods is the so called *fast multipole method* (FMM) formulated in the early 1990s by Rokhlin *et al* [16] which was rated as one of the top-ten algorithms of the 20th century [17]. Before presenting a brief overview of the FMM and its extension the *multilevel fast multipole algorithm* (MLFMA) introduced in 1997 by Song *et al.*, it is necessary to investigate the concept of *iterative solvers* for obtaining the solution pertaining to the MoM formulation.

In the previous Section, the concept of LU-decomposition was introduced to illustrate one method by which a solution to the MoM formulation can be obtained. In this context the solution refers to the unknown discretized current distribution on the scatterer. When applying the MoM technique to a scatterer such as that presented in Figure 2.2 one obtains the following matrix equation,

$$[Z_{RWG}] \{I_{RWG}\} = \{V_{RWG}\} \quad (2.2)$$

the derivation of which will be presented in detail in Chapter 4. For now consider the following qualitative description for each of the elements in Eq. (2.2): The matrix, $[Z_{RWG}]$, is an $N_{RWG} \times N_{RWG}$ matrix that accounts for the self and mutual *interactions* between all the basis-functions that discretize the structure. The vector, $\{I_{RWG}\}$, represents the unknown surface current distribution, ie. the solution to the problem, and the vector $\{V_{RWG}\}$ is related to the applied excitation, such as an incoming plane wave for instance.

When solving Eq. (2.2) by means of LU-decomposition it was noted in the previous section that the computational cost associated with the method scales as $O(N_{RWG}^3)$ - a cost that may

become unacceptably high as N_{RWG} becomes large. In the 1980s, iterative solvers such as the conjugate gradient (CG) algorithm started to attract much attention in the CEM community as a means of circumventing the cost associated with direct methods such as LU-decomposition [1]. Briefly, the underlying concept of these solvers is to iterate through a number of solution vectors, $\{I_{RWG,n}\}$, with $n = 1, 2, \dots, N_{iter}$ to obtain a solution estimate that minimizes the following residual vector [18],

$$\{r_n\} = [Z_{RWG}] \{I_{RWG,n}\} - \{V_{RWG}\} \quad (2.3)$$

where the solution estimate, $\{I_{RWG,n}\}$, can be expressed in the following form,

$$\{I_{RWG,n}\} = \{I_{RWG,n-1}\} + \alpha_n P_n \quad (2.4)$$

with P_n the so called "direction vector" that determines the *direction* in the N -dimensional space in which the algorithm moves to correct the estimate of the solution vector, $\{I_{RWG,n}\}$. The scalar coefficient, α_n , determines how *far* the algorithm moves in the P_n direction.

If one supposes that a sufficiently accurate solution estimate is obtained after N_{iter} -iterations, then the computational cost associated with the iterative solution technique is of $\mathcal{O}(N_{iter} \times N_{RWG}^2)$ as a complex matrix-vector multiplication is at the heart of each iteration as evident from Eq. (2.3). Unfortunately, these methods have a few drawbacks: firstly, research over the years have indicated that it is very difficult to predict N_{iter} for arbitrary problems and secondly that it does nothing to reduce the $\mathcal{O}(N_{RWG}^2)$ memory requirement associated with the storage of the impedance matrix [1]. To improve on the computational advantage of using iterative solvers for the MoM matrix equation, they are therefore mainly used together with other techniques such as the FMM, which in its most powerful multilevel form reduces the computational cost from $\mathcal{O}(N_{iter} \times N_{RWG}^2)$ to $\mathcal{O}(N_{iter} \times N_{RWG} \log N_{RWG})$ while simultaneously addressing the storage issue pertaining to large impedance matrices.

Briefly, the FMM in its original form [16; 19] was focussed on *grouping* together basis-functions that are physically separated from each other. Otherwise stated these groups of basis functions essentially reside in each other's *far-fields* or *far-zones* and hence certain approximations can be made regarding the interaction between these groups. Similarly, groups of unknowns that are in close proximity to each other are said to be in the *near-zone* region. The near-zone interactions are carried out by an explicit matrix-vector multiplication in the iterative solution phase. The far-zone interactions are replaced by a more efficient calculation of the matrix-vector multiplication which is derived by making far-field approximations. When the basis-functions are grouped according to a certain optimal *box-size* of $\sqrt{N_{RWG}}$, the resulting overall computational complexity is of $\mathcal{O}(N_{iter} \times N_{RWG}^{3/2})$ [1; 18]. By introducing a recursive hierarchy of groups where the same far-zone approximations are made within an aggregate of basis-functions, the computational complexity can be reduced further to $\mathcal{O}(N_{iter} \times N_{RWG} \log N_{RWG})$. This multilevel approach is at the core of the MLFMA.

To review the computational efficiency of the techniques presented thus far, it is worth comparing the computational cost obtained by LU-decomposition, $\mathcal{O}(N_{RWG}^3)$, an iterative solver such as CG, $\mathcal{O}(N_{iter} \times N_{RWG}^2)$ and the MLFMA, $\mathcal{O}(N_{iter} \times N_{RWG} \log N_{RWG})$. In the case of the iterative solver (CG), let $N_{iter} = 100$, and for the MLFMA $N_{iter} = 1000$. These figures were obtained from [1] as pertaining to a very rapidly converging iterative solution and a very well optimized FMM implementation respectively and is presented in Figure 2.3. The results are related to a system that is capable of sustaining 1 teraflop. From the results it is evident that in the optimistic case where convergence is obtained in a limited number of iterations, the MLFMA performs much better than the iterative and direct solution methods.

Another important factor that needs to be considered is that of memory usage. From [1] the storage requirements of an MLFMA implementation was obtained as $\mathcal{O}(10 \times N_{RWG} \log N_{RWG})$

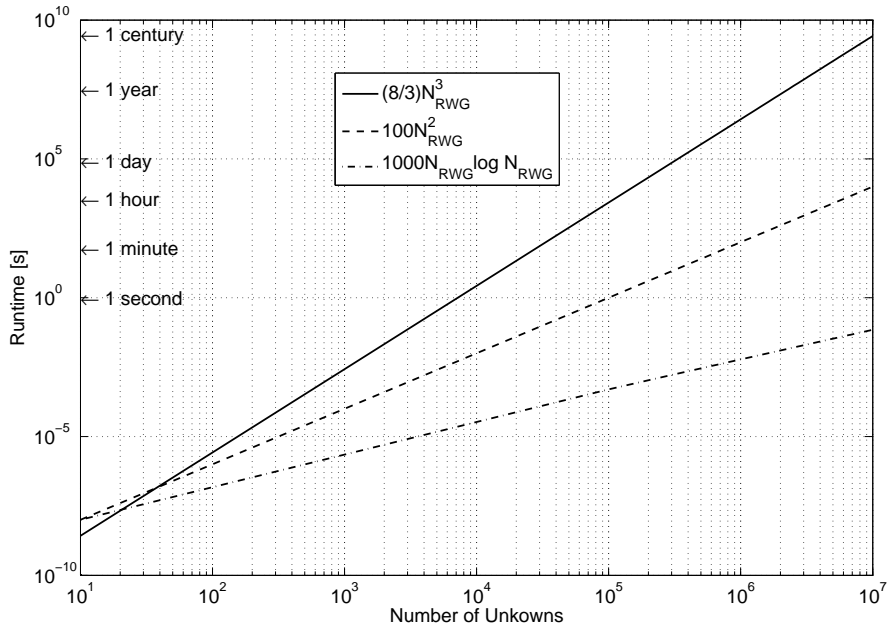


Figure 2.3: A comparison between the runtime associated with LU-decomposition, a fast-converging iterative solver and the MLFMA (adapted from [1]).

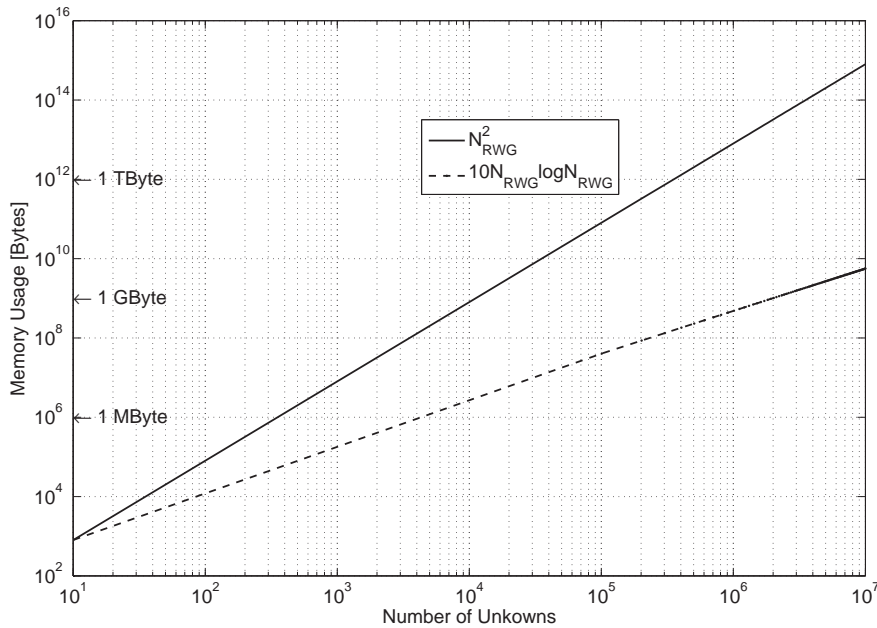


Figure 2.4: A comparison between the memory requirement associated with the conventional MoM and the MLFMA respectively (adapted from [1]).

due to the fact that only near-field interactions are stored. In Fig. (2.4), the memory usage related to the MLFMA is compared to that of the conventional MoM technique that is dominated by the storage of the impedance matrix, i.e. $O(N_{RWG}^2)$. From the results it is noted that a significant improvement is obtained in terms of memory utilisation.

Although the results related to the MLFMA illustrated in Fig. (2.3) and Fig. (2.4) present

quite an improvement above that of the direct solutions when the number of unknowns increase, it is still inherently limited by the convergence rate of the iterative solver, i.e. the value of N_{iter} . The previous runtimes associated with the MLFMA are associated with iterative solvers that converge rather (optimistically) quickly. In the following section, approximate techniques that are based on using direct methods for the solution phase, such as LU-decomposition, are presented as a means of overcoming this problem.

2.4 Macro Domain Basis Function (MBF) Techniques

To summarise, the direct techniques presented thus far, such as LU-decomposition, operates on the full MoM matrix equation that is of $O(N_{RWG} \times N_{RWG})$ which may result in unacceptable computational issues when the number of unknowns, N_{RWG} , increases. The MLFMA aims to overcome this problem by making *far-field* approximations for the interactions between sufficiently separated groups of basis functions which reduces the cost to $O(N_{iter} \times N_{RWG} \log N_{RWG})$. Unfortunately, this method is still subject to the use of iterative solvers that can suffer from convergence issues if the matrix equation is ill-conditioned.

In this section, the focus shifts to iteration-free techniques that are associated with the aggregation of *low-level* basis functions from which *high-level* or *macro-domain* basis-functions (MBFs) are constructed. These techniques focus on obtaining a reduced matrix equation that can be solved directly by using LU-decomposition, i.e. without resorting to the use of an iterative solver.

Numerous macro-domain function approaches have become available in the last decade, some of which are somewhat limited to periodic structures, such as the sub-entire-domain basis function method (SED) [20] or the sub-domain multilevel approach (SMA) that is mostly suited to planar antenna structures [21]. Other MBF-techniques such as the synthetic function approach (SFX) [22] and the characteristic basis function method (CBFM) [23] can be applied to more general EM structures. In the context of the SKA project, specifically where FPA feedstructures are of concern, significant attention has been devoted to the CBFM approach¹¹. The reason for this is primarily due to its versatility in modelling complex antenna structures by aggregating subsectional basis such as the RWG functions that present a high degree of geometrical flexibility [24]. When compared to the SFX, although similar in many respects, the CBFM does not introduce additional unknowns at the junction between the different macro-domains also referred to as sub-domains. In the chapters to follow, the CBFM therefore forms a critical part of the author's research to develop efficient means of analyzing large FPA structures in a limited time and cost framework. Before we proceed, it is however necessary to illustrate the underlying concept of how, by applying macro-domain methods such as the CBFM to the MoM formulation, one can solve large CEM structures efficiently. The theory behind the concepts discussed hereafter is primarily obtained from [23].

Consider a simple PEC plate configuration as depicted in Figure 2.5 that is discretized with a large number of RWG basis-functions, in the order of a few thousands. To avoid working with the high-order MoM matrix equation associated with the problem as a whole, the first step in the CBFM is to partition the problem-domain into smaller, more manageable sub-problems. In the case of the scatterer presented in Figure 2.5 let each of the plates constitute one sub-domain, two of which are illustrated as p and q respectively. Suppose now that the plate configuration is

¹¹Much of the research conducted in this regard is undertaken at the Netherlands institute for radio astronomy, ASTRON

illuminated by a normally incident plane wave¹². Our goal is to determine the induced surface current, \mathbf{J}_{RWG} , on the plate and from this other quantities can then be calculated such as the radar cross section (RCS) of this scatterer.

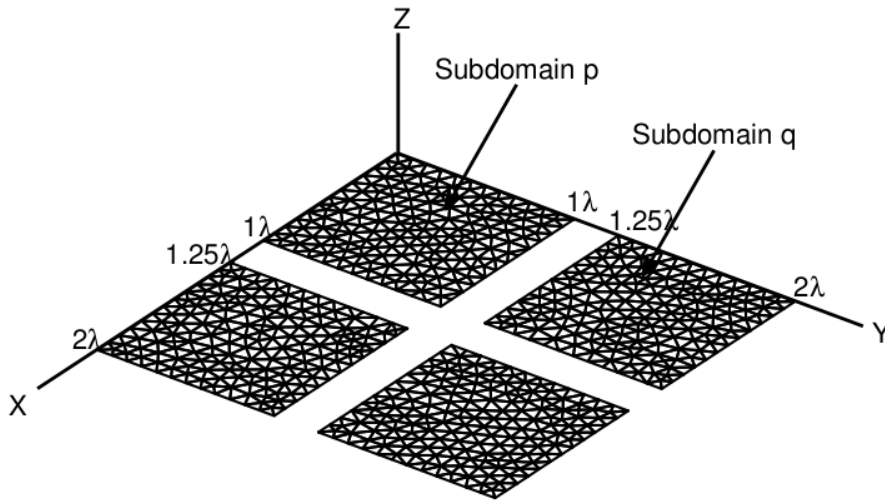


Figure 2.5: A PEC scatterer consisting of 4 PEC plates, discretized with RWG basis functions.

The MoM matrix equation formulated to obtain the unknown surface-current density, \mathbf{J}_{RWG} , is that of Eq. (2.2). This equation can however be presented in a segmented-form according to the sub-domains that the problem has been subdivided into, i.e. $1, \dots, 4$,

$$\begin{bmatrix} [Z_{RWG}^{(1,1)}] & [Z_{RWG}^{(1,2)}] & [Z_{RWG}^{(1,3)}] & [Z_{RWG}^{(1,4)}] \\ [Z_{RWG}^{(2,1)}] & [Z_{RWG}^{(2,2)}] & [Z_{RWG}^{(2,3)}] & [Z_{RWG}^{(2,4)}] \\ [Z_{RWG}^{(3,1)}] & [Z_{RWG}^{(3,2)}] & [Z_{RWG}^{(3,3)}] & [Z_{RWG}^{(3,4)}] \\ [Z_{RWG}^{(4,1)}] & [Z_{RWG}^{(4,2)}] & [Z_{RWG}^{(4,3)}] & [Z_{RWG}^{(4,4)}] \end{bmatrix} \begin{bmatrix} \{I_{RWG}^{(1)}\} \\ \{I_{RWG}^{(2)}\} \\ \{I_{RWG}^{(3)}\} \\ \{I_{RWG}^{(4)}\} \end{bmatrix} = \begin{bmatrix} \{V_{RWG}^{(1)}\} \\ \{V_{RWG}^{(2)}\} \\ \{V_{RWG}^{(3)}\} \\ \{V_{RWG}^{(4)}\} \end{bmatrix} \quad (2.5)$$

In this equation, the matrix $[Z_{RWG}^{(p,q)}]$, represents either the *self-interaction* part of the MoM formulation pertaining to a subdomain p with $p = q$, or the *mutual-coupling* terms between subdomains p and q when $p \neq q$. When considering the mutual-coupling matrix $[Z_{RWG}^{(p,q)}]$ with $p \neq q$, the source is located on domain p while the observation points is located on domain q . The vector $\{I_{RWG}^{(p)}\}$ represents the complex expansion coefficients¹³ that are used to calculate the resulting surface current density on domain p . The vector $\{V_{RWG}^{(p)}\}$ represents the excitation associated with subdomain p . Suppose further that there are $N_{RWG}^{(p)}$ unknowns associated with the p th subdomain and $N_{RWG}^{(q)}$ unknowns associated with the q th subdomain and that the number of low-level unknowns are more or less equal, i.e. $N_{RWG}^{(p)} \simeq N_{RWG}^{(q)}$ ¹⁴.

¹²Note however that any other type of excitation could also have been used

¹³These are merely complex coefficients that define the normal component of the current flowing across an internal edge between two triangles. This concept will be explained in more detail in Chapter 4 when RWG basis-functions are introduced.

¹⁴Strictly speaking the number of unknowns that are contained in each of the CBFM sub-domains do not have to be equal. In Chapter 5 however it will be explained that when considering a parallelized CBFM algorithm, keeping

The incident field local to sub-domain $p = 1$ for example can then be related via the segmented impedance matrices to the induced surface current on each of the other domains as follows,

$$\left[Z_{RWG}^{(1,1)} \right] \{ I_{RWG}^{(1)} \} + \left[Z_{RWG}^{(1,2)} \right] \{ I_{RWG}^{(2)} \} + \left[Z_{RWG}^{(1,3)} \right] \{ I_{RWG}^{(3)} \} + \left[Z_{RWG}^{(1,4)} \right] \{ I_{RWG}^{(4)} \} = \{ V_{RWG}^{(1)} \} \quad (2.6)$$

The surface current, $\{ I_{RWG}^{(p)} \}$, that will be induced on each of the plates primarily results from two types of sources: (a) the currents induced by the incident electric field¹⁵ and (b) non-local excitations that correspond to the field-coupling between the different plates [25]. Mathematically, the surface current induced on the p th sub-domain can then be expressed as follows,

$$\{ I_{RWG}^{(p)} \} \simeq \{ I_{RWG,prim}^{(p)} \} + \{ I_{RWG,sec}^{(p)} \} \quad (2.7)$$

Where $\{ I_{RWG,prim}^{(p)} \}$ represent the "primary" current-components due to the incident electric field, i.e. the current distribution on domain p in the absence of all the other domains. The vector $\{ I_{RWG,sec}^{(p)} \}$ represents the currents resulting from the mutual coupling between domain p and the other subdomains, i.e. the so called "secondary" currents. By substituting this expression for the current in terms of its "primary" and "secondary" components into Eq. (2.6), we obtain the following approximate expression relating the induced current to the incident field,

$$\begin{aligned} & \left[Z_{RWG}^{(1,1)} \right] \{ I_{RWG,prim}^{(1)} \} + \left[Z_{RWG}^{(1,1)} \right] \{ I_{RWG,sec}^{(1)} \} + \\ & \left[Z_{RWG}^{(1,2)} \right] \{ I_{RWG,prim}^{(2)} \} + \left[Z_{RWG}^{(1,2)} \right] \{ I_{RWG,sec}^{(2)} \} + \\ & \left[Z_{RWG}^{(1,3)} \right] \{ I_{RWG,prim}^{(3)} \} + \left[Z_{RWG}^{(1,3)} \right] \{ I_{RWG,sec}^{(3)} \} + \\ & \left[Z_{RWG}^{(1,4)} \right] \{ I_{RWG,prim}^{(4)} \} + \left[Z_{RWG}^{(1,4)} \right] \{ I_{RWG,sec}^{(4)} \} \simeq \{ V_{RWG}^{(1)} \} \end{aligned} \quad (2.8)$$

which may be rewritten in terms of "primary" and "secondary" components as,

$$\left[Z_{RWG}^{(1,1)} \right] \{ I_{RWG,prim}^{(1)} \} + \left[Z_{RWG}^{(1,1)} \right] \{ I_{RWG,sec}^{(1)} \} \simeq \{ V_{RWG}^{(1)} \} - \{ V_{coupling}^{(1)} \} \quad (2.9)$$

where $\{ V_{coupling}^{(1)} \}$ represents the incident fields when the induced currents on the other subdomains are used as distant sources and following Eq. (2.8) can be expressed as,

$$\begin{aligned} \{ V_{coupling}^{(1)} \} = & \left[Z_{RWG}^{(1,2)} \right] \{ I_{RWG,prim}^{(2)} \} + \left[Z_{RWG}^{(1,2)} \right] \{ I_{RWG,sec}^{(2)} \} + \\ & \left[Z_{RWG}^{(1,3)} \right] \{ I_{RWG,prim}^{(3)} \} + \left[Z_{RWG}^{(1,3)} \right] \{ I_{RWG,sec}^{(3)} \} + \\ & \left[Z_{RWG}^{(1,4)} \right] \{ I_{RWG,prim}^{(4)} \} + \left[Z_{RWG}^{(1,4)} \right] \{ I_{RWG,sec}^{(4)} \} \end{aligned} \quad (2.10)$$

We can proceed even further to say that the induced current from the "secondary" current-distribution on domain $q \neq 1$ will have very little contribution to the term $\{ V_{coupling}^{(1)} \}$, or otherwise stated we only account for first-order mutual coupling effects. In this case, Eq. (2.10) can be simplified as,

$$\{ V_{coupling}^{(1)} \} \simeq \left[Z_{RWG}^{(1,2)} \right] \{ I_{RWG,prim}^{(2)} \} + \left[Z_{RWG}^{(1,3)} \right] \{ I_{RWG,prim}^{(3)} \} + \left[Z_{RWG}^{(1,4)} \right] \{ I_{RWG,prim}^{(4)} \} \quad (2.11)$$

where only the coupling from "primary" currents on domains $q = 2 \dots 4$ have been taken into account.

$N_{RWG}^{(p)}$ more or less equal between the sub-domains, is recommended to achieve efficient load-balancing.

¹⁵In the case of radiators, the surface currents will be due to local excitations at the input-ports

The physical approximations that is discussed in the previous paragraph is at the core of the CBFM, namely isolating the effect of the primary incident fields and that of the scattered fields that contribute toward the mutual coupling between the sub-domains. A more general approach for the CBFM approach associated with scatterers can be applied as follows: consider M sub-domains (where $M = 4$ in the case of Figure 2.5). The CBFM formulation attempts to model the "primary" current distribution on each of the M sub-domains by solving the following matrix equation,

$$\left[Z_{RWG}^{(i,i)} \right] \{ J_{CBFM,prim}^{(i)} \} = \{ V_{RWG}^{(i)} \} \text{ for } i = 1, \dots, M \quad (2.12)$$

The vector $\{ J_{CBFM,prim}^{(i)} \}$ represents the $N_{RWG}^{(i)}$ complex expansion coefficients that are associated with the characteristic basis function (CBF), $\mathcal{J}_{CBFM,prim}^{(i)}$, on the i th sub-domain.

The computational cost of Eq. (2.12) is of $\mathcal{O}(N_{RWG}^{(i)} \times N_{RWG}^{(i)})$ with $N_{RWG}^{(i)}$ typically ranging from a few hundred to a few thousand unknowns. Eq. (2.12) can therefore be solved efficiently by using direct methods such as LU-decomposition to obtain the primary CBFs.

To improve the accuracy of the analysis, we can include the effect of mutual coupling between the various sub-domains. If only first-order effects are considered¹⁶, i.e. coupling currents induced on sub-domain i will include only those due to the *primary* currents on the other subdomains, we can generate a set of "secondary currents" for each domain as,

$$\left[Z_{RWG}^{(i,i)} \right] \{ J_{CBFM,sec}^{(i,j)} \} = - \left[Z_{RWG}^{(i,j)} \right] \{ J_{CBFM,prim}^{(j)} \} \text{ for } j = 1, 2, \dots, i-1, i+1, \dots, M \quad (2.13)$$

where $\{ J_{CBFM,sec}^{(i,j)} \}$ are the $N_{RWG}^{(i)}$ complex expansion coefficients associated with the "secondary" CBF, $\mathcal{J}_{CBFM,sec}^{(i)}$, on the i th sub-domain due to the "primary" CBF of the j th subdomain. The $N_{RWG}^{(i)} \times N_{RWG}^{(j)}$ matrix, $\left[Z_{RWG}^{(i,j)} \right]$, represents the coupling matrix between the i th and j th subdomain.

Each sub-domain, i , therefore hosts the following set of $N_{RWG}^{(i)}$ complex expansion coefficients pertaining to the CBFs local to that region which may be written as a column augmented matrix as,

$$\left[J_{CBFM}^{(i)} \right] = \left[\{ J_{CBFM,prim}^{(i)} \}, \dots, \{ J_{CBFM,sec}^{(i,j)} \} \right] \text{ for } j = 1, 2, \dots, i-1, i+1, \dots, M \quad (2.14)$$

where $\{ J_{CBFM,sec}^{(i,j)} \}$ is the secondary CBF induced on the i th subdomain, due to the j th primary CBF on all subdomains $j \neq i$. For each of the sub-domains there will therefore be M characteristic basis functions.

The solution to the entire problem, that is the expansion coefficient entries in the vector $\{ I_{RWG} \}$ of Eq. (2.2), can then be expressed as a linear superposition of the "primary" and "secondary" CBFs, each of which is weighted by an unknown complex coefficient, α , as follows,

$$\begin{aligned} \{ I_{RWG} \}_{N_{RWG} \times 1} &= \sum_{m=1}^M \alpha_m^{(1)} \{ J_{CBFM,m}^{(1)} \} + \sum_{m=1}^M \alpha_m^{(2)} \{ J_{CBFM,m}^{(2)} \} \\ &+ \dots + \sum_{m=1}^M \alpha_m^{(M)} \{ J_{CBFM,m}^{(M)} \} \end{aligned} \quad (2.15)$$

where $\{ J_{CBFM,m}^{(i)} \}$ is the m th CBF of the i th sub-domain, weighted by the complex coefficient, $\alpha_m^{(i)}$. In this case, the vector $\{ J_{CBFM,m}^{(i)} \}$, is mapped to its *global* position in terms of the RWG discretization. It will therefore consist of zero entries for all other subdomains, except for those related to the indices on the i th domain.

¹⁶Extending the method to incorporate tertiary basis functions can be achieved by using the "secondary" CBFs as sources, as discussed in [26]

If the solution vector expressed as in Eq. (2.15) is substituted into the original matrix equation presented in Eq. (2.2), we can obtain a reduced equation that is of the following form,

$$[Z_{CBFM}] \{I_{CBFM}\} = \{V_{CBFM}\} \quad (2.16)$$

where $[Z_{CBFM}]$ represents the reduced impedance matrix of size $M^2 \times M^2$. The vectors $\{I_{CBFM}\}$ and $\{V_{CBFM}\}$ represents the $M^2 \times 1$ solution vector and excitation vector respectively. The solution vector $\{I_{CBFM}\}$ contains the complex α coefficients introduced in Eq (2.15).

The reduced equation entries may be expressed as $M \times M$ matrices of the form, $[Z_{CBFM}^{(p,q)}]$ with $\{p, q\} = 1 \dots M$, by employing a Galerkin testing scheme¹⁷, where the source and testing functions are the column augmented CBF expansion coefficients of Eq. (2.14). The sub-matrix, $[Z_{CBFM}^{(p,q)}]$, can then be computed as,

$$[Z_{CBFM}^{(p,q)}] = \langle [J_{CBFM}^{(p)}]_{GLOB}^T, [Z_{RWG}] [J_{CBFM}^{(q)}]_{GLOB} \rangle \quad (2.17)$$

where T is the transposition operator and $\langle a, b \rangle$ denotes the inner product between vectors (or matrices) a and b . In addition, $[J_{CBFM}^{(i)}]_{GLOB}$ is the *globally mapped* column-augmented CBFs generated on sub-domains $i = p$ and $i = q$ respectively. $[Z_{RWG}]$ is the $N_{RWG} \times N_{RWG}$ impedance matrix expressed in Eq. (2.2).

By noting that in the calculation of Eq. (2.17), the matrix $[J_{CBFM}^{(i)}]_{GLOB}$, contains zero entries for the RWG indices not related to subdomains p and q respectively, the calculation can be simplified to

$$[Z_{CBFM}^{(p,q)}] = \langle [J_{CBFM}^{(p)}]_{LOC}^T, [Z_{RWG}^{(p,q)}] [J_{CBFM}^{(q)}]_{LOC} \rangle \quad (2.18)$$

In this case, $[Z_{RWG}^{(p,q)}]$ is the $N_{RWG}^{(p)} \times N_{RWG}^{(q)}$ sub-matrix whose entries represent the mutual reaction between the RWGs in domain p and q and $[J_{CBFM}^{(i)}]_{LOC}$ contains only the non-zero entries of the CBF expansion coefficients related to these subdomains respectively.

Similarly, the CBF excitation vector entries, $\{V_{CBFM}^{(p)}\}$, can be obtained as follows,

$$\{V_{CBFM}^{(p)}\} = \langle [J_{CBFM}^{(p)}]_{LOC}^T, \{V_{RWG}^{(p)}\} \rangle \quad (2.19)$$

where $\{V_{RWG}^{(p)}\}$, are the entries of the primary excitation vector, $\{V_{RWG}\}$, of Eq. (2.2), corresponding to the RWGs on subdomain p .

The reduced matrix equation presented in Eq. (2.16) is significantly smaller than the original MoM matrix equation presented in Eq. (2.2) and can be solved efficiently using direct methods. Additionally, each of the CBFs are also generated by solving a small matrix equation that is directly related to the number of unknowns contained in the CBFM sub-domain. The equations, Eq. (2.12) and Eq. (2.13), can therefore be solved using an LU-decomposition of $\mathcal{O}((N_{RWG}^{(i)})^3)$ with $N_{RWG}^{(i)} \ll N_{RWG}$. It is also worth noting that the LU-factors generated when the self-interactions on each sub-domain (the "primary" CBFs) are calculated, can be stored and recycled in the calculation of the "secondary" CBFs in Eq. (2.13).

The total operational count for the CBFM formulation presented in this section can be expressed as follows,

$$\frac{8M}{3} (N_{RWG}^{(i)})^3 + M^3 (N_{RWG}^{(i)})^2 \quad (2.20)$$

¹⁷This concept of *weighting* will be explained in Chapter 4.1.3 on page 38

where the first term accounts for the LU-decomposition associated with the generation of the "primary" and "secondary" CBFs in Eq. (2.12) and Eq. (2.13) and the second term for the M^3 complex matrix-vector multiplications during the reduced matrix equation calculation, Eq. (2.18). The operational count expressed by Eq. (2.20) assumes that the number of unknowns contained in each of the sub-domains are more or less equal, i.e. $N_{RWG}^{(i)} \simeq N_{RWG}^{(j)}$.

Eq. (2.20) can be rewritten in terms of the global RWG unknowns by noting that $N_{RWG}^{(i)} = N_{RWG}/M$ as,

$$\frac{8}{3M^2}(N_{RWG})^3 + M(N_{RWG})^2 \quad (2.21)$$

The growth of Eq. (2.21) with respect to N_{RWG} can be minimized¹⁸ by calculating the derivative of Eq. (2.21) with respect to M and setting it equal to zero. In that case, the value for M that minimizes the total CBFM operational count is calculated as,

$$M = (3/4)^{\frac{1}{4}}(N_{RWG})^{\frac{1}{4}} \quad (2.22)$$

when considering that $M = N_{RWG}/N_{RWG}^{(i)}$. The total operational count associated with the CBFM formulation can then be expressed as,

$$\frac{8}{3K}(N_{RWG})^{2\frac{1}{2}} + K(N_{RWG})^{2\frac{1}{4}} \text{ with } K = (3/4)^{\frac{1}{4}} \quad (2.23)$$

Asymptotically, the computational cost associated with the CBFM scales well when compared to direct methods, i.e. $\mathcal{O}((N_{RWG})^{2\frac{1}{2}})$ as opposed to $\mathcal{O}(N_{RWG})^3$. When compared to methods based on iterative solution techniques, such as the MLFMA, the comparison is more difficult as it depends on the convergence rate of the iterative solution. In Figure 2.6, the CBFM runtime is compared to an iterative solver and MLFMA implementation that converges after $N_{iter} = 500$ and $N_{iter} = 10,000$ respectively. The results associated with LU-decomposition are included for comparison. The runtimes are based on an architecture that is capable of sustaining 1 teraflop of processing power. The CBFM sub-domain size is based on that obtained in Eq. (2.22).

The memory requirements associated with the CBFM¹⁹, i.e. $\mathcal{O}(M(N_{RWG}^{(i)})^2)$, scales more gradually compared to direct solution techniques, which is $\mathcal{O}((N_{RWG})^2)$. The memory usage associated with a direct solver, an efficient MLFMA implementation and the CBFM are compared in Figure 2.7. From the results it is evident that the CBFM and MLFMA perform significantly better than the direct solver. For this particular example, it can also be noted that the MLFMA performs better than the CBFM formulation. One should however keep in mind that a real FMM implementation is unlikely to be this memory efficient, as is stated in [1] and also that convergence cannot always be guaranteed for iterative solution techniques such as the MLFMA.

2.5 Conclusion

In this Chapter, various well-known CEM techniques that are suited to the numerical analysis of large electromagnetic structures, have been reviewed. These techniques included the use of

¹⁸This *optimal* value for the number of sub-domains, M , is specific to this type of problem from a *runtime* point of view. Various choices for M exists, depending on the problem domain, the number of CBFs generated on each domain, the level of accuracy, etc. The value calculated in the example, is only used for illustration purposes and should not be regarded as a fixed guideline.

¹⁹In this context, the memory requirement of the CBFM is associated with only the data corresponding to the sub-domain under consideration loaded into memory, as is done in [23].

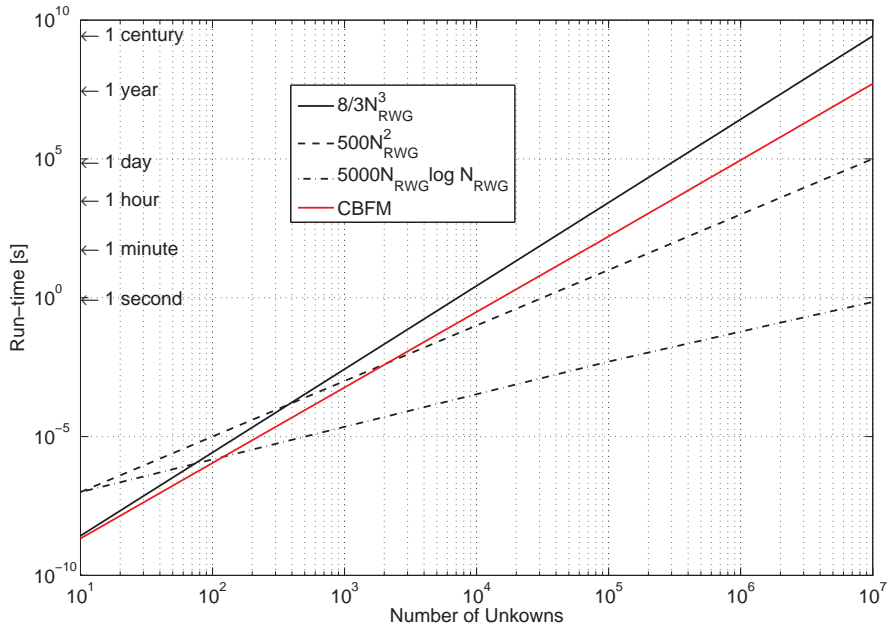


Figure 2.6: A comparison between the runtime associated with LU-decomposition, an iterative solver, the MLFMA and the CBFM formulation (adapted from [1]).

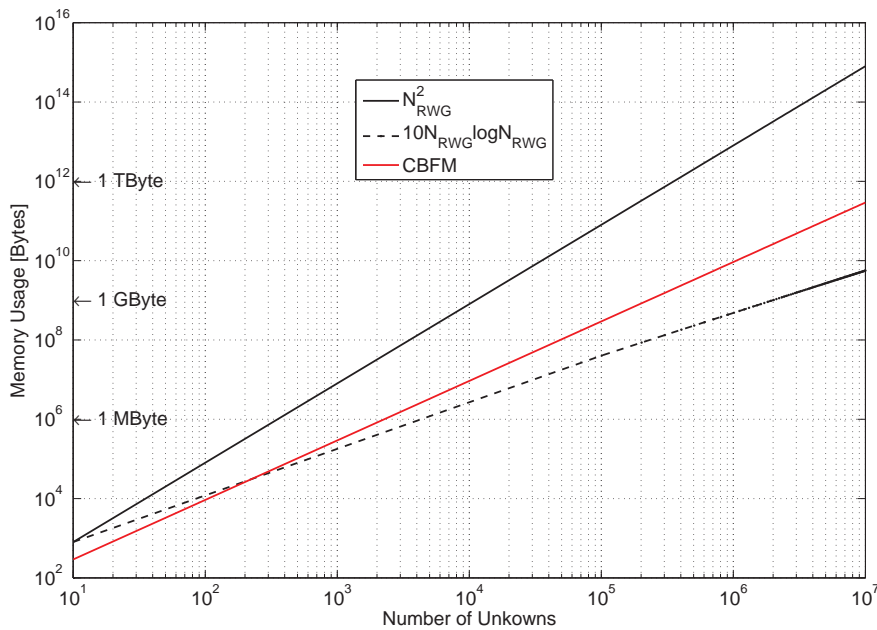


Figure 2.7: A comparison between the memory requirement associated with the conventional MoM, the MLFMA and the CBFM respectively (adapted from [1]).

a parallel computing infrastructures or so-called high performance computing (HPC) to apply domain-decomposition to the solution of large data-sets thereby reducing the overall memory usage and computational runtimes of algorithms. Other techniques based on iterative solvers, such as the FMM and its extension the MLFMA, were also introduced as a means by which the computational burden associated with full-wave MoM formulations, can be reduced. Some challenges that are associated with the aforementioned methods, specifically the uncertainty

associated with the number of iterations, can prove a limitation in these techniques. To address this problem, a class of techniques based on direct solution methods, such as LU-decomposition, was investigated. Detailed attention was devoted to one such a method, namely the CBFM. The CBFM approach entails that the problem as a whole be sub-divided into a number of smaller sub-domains. Each of these sub-domains then support a set of characteristic basis functions (CBFs) that is generated by taking the physics of the problem into consideration by separating "primary" and "secondary" field-coupling.

The previously mentioned techniques were compared in terms of runtime and memory usage on a qualitative manner, based on results obtained from example figures derived throughout the Chapter as well as those available in the literature. From the results it was observed that the CBFM and MLFMA perform significantly better than a direct solver for large problems, both in terms of memory usage and computational runtime.

The CBFM is not subject to convergence rates, as is the MLFMA, and will therefore form a crucial part of the author's research in Chapter 5. In the following Chapter, the results of using a commercial MoM-based CEM software product, FEKO, in a high-performance computing (HPC) environment will be presented.

Chapter 3

Using CEM Tools in a Parallel Computing Environment

In the previous Chapter, Section 2.2 on page 4, the concept of applying HPC resources to improve the computational runtime and memory usage associated with the CEM solution of large electromagnetic structures was introduced. The purpose of this Chapter is to present the results of using the MoM based electromagnetic simulation software package FEKO¹ in a parallel or HPC environment to analyse electrically large FPAs. The FPA structure simulated is that of various sized linearly polarized Vivaldi antenna arrays. The parallel computing infrastructure is called the *iQudu* cluster and is hosted by the Centre for High Performance Computing (CHPC) in Cape Town.

It is to be noted that this Chapter is not focussed on the detailed electromagnetic properties of the specific Vivaldi FPA-design, but rather on *the scalability associated with the full-wave parallel MoM formulation in FEKO, when used to simulate electrically large FPAs, such as the Vivaldi structure*. This will be referred to as *benchmarking* the scalability of FEKO on the HPC cluster used, namely the *iQudu*.

As stated in the previous Chapters, the MoM forms a crucial theme on which most of the work in this thesis is based. Thus far, the results of the MoM formulation was only briefly reviewed in Section 2.3 on page 8 where the so called MoM matrix equation was presented (Eq. (2.2)). A *sequential* overview of the MoM algorithm is presented in the following section, after which the focus shifts to the parallel implementation thereof in the FEKO kernel.s

3.1 Overview of the MoM algorithm

The Method of Moments (MoM), is one of the first and widely accepted numerical methods for the analysis of electromagnetic structures such as antennas and scatterers [1]. As stated in Section 2.1, the MoM is a full-wave numerical solution of integral equations (known as Maxwell's equations) in the frequency domain. A particular advantage of this method, is that it is a "source method", meaning that only the structure in question is discretised [27], as illustrated in Figure 3.1 on page 20.

In [28] the implementation of the sequential and parallel MoM solver in FEKO² is explained. Only the key concepts of that study is repeated in this Section. A graphical illustration of the sequential solution process of the MoM, is illustrated in Figure 3.2.

¹FEKO version 5.4 was used for the simulations.

²It is to be noted that the *general* MoM algorithm as explained here is *generic*, and is not associated with the FEKO implementation explicitly.

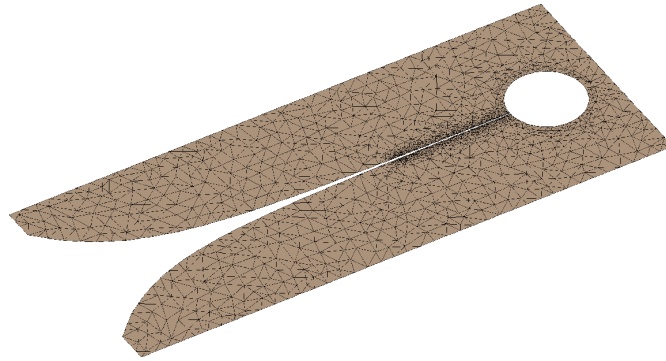


Figure 3.1: Example of an electromagnetic scatterer, with a discretized surface. The figure was obtained from POSTFEKO, part of the FEKO package.

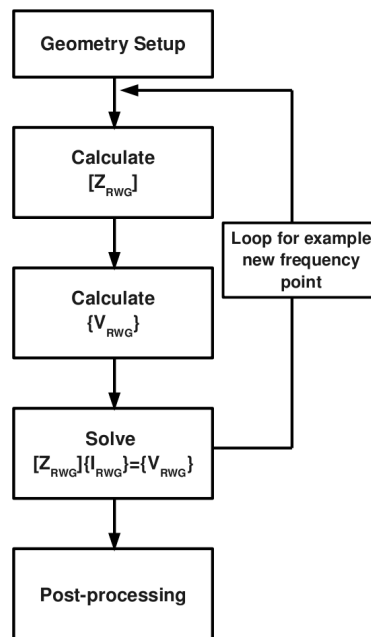


Figure 3.2: Schematic overview of the sequential MoM algorithm

The first phase depicted in the simplified flow-chart of Figure 3.2 is the *Geometry Setup*. This step entails that the input data be read from a file which is then used to discretize the structure into triangular elements (also referred to as "meshing"), after which the common edges between the triangles are located. This information is required to construct the *basis functions*³. As stated in Section 2.1 on page 3 these functions are expressed as a linear superposition that is weighted with unknown complex coefficients in order to approximate the unknown surface current distribution on the electromagnetic structure. With the surface currents known, other electromagnetic properties of the structure can easily be determined, such as electric and magnetic near and far-fields, far-field radiation patterns and frequency characteristics such as input impedance.

³From [1] it follows that the choice of the basis function is one of the most crucial parts of the MoM. A large variety of basis functions exists, such as pulse, polynomial, piecewise sinusoidal, etc. FEKO uses RWG-type, i.e. triangular basis functions for metallic radiators such as that illustrated in Figure 3.1.

The *interaction* between these basis functions are then formulated in a mathematical form to calculate the $N_{RWG} \times N_{RWG}$ *impedance* matrix, $[Z_{RWG}]$, as stated in the previous Chapter, Section 2.3 on page 8. The application of certain boundary conditions for the electromagnetic fields on the discretised surface then leads to a coupled set of integral equations, which can then be transformed into a set of linear equations illustrated in Section 2.3 and repeated here for convenience,

$$[Z_{RWG}] \{I_{RWG}\} = \{V_{RWG}\} \quad (3.1)$$

the derivation of which will be presented in detail in Chapter 4.

The solution of Eq. (3.1) can be obtained by various techniques as was illustrated in Chapter 2, such as LU-decomposition that presents a computational complexity of $\mathcal{O}(N_{RWG}^3)$, the MLFMA of $\mathcal{O}(N_{iter} \times N_{RWG} \log N_{RWG})$ and the CBFM that is of $\mathcal{O}((N_{RWG})^{2\frac{1}{2}})$, where M is the total number of CBFM sub-domains each containing in the order of $N_{RWG}^{(i)}$ RWG basis-functions.

Once the solution vector $\{I_{RWG}\}$ is determined, i.e. the unknown (discretized) surface current distribution on the scatterer, other properties such as the far-field radiation pattern of the scatterer (in the case of an antenna) can then be calculated.

In the following Section, the parallelisation of the various phases of the full-wave MoM solution process as implemented in FEKO is explained. The solution-technique associated with obtaining the unknown current distribution, is that of *parallel* LU-decomposition.

3.2 Parallelisation of the MoM algorithm in FEKO

The parallelisation of the MoM in FEKO is based on the message passing standard, *MPI*. As explained in Section 2.2 on page 4, *MPI* is a programming model that is frequently used for technical applications on cluster systems with distributed memory. The standard describes the programming interface to a communication library which realises the distribution of data over the parallel processes as well as interprocess communication. The remainder of this section discusses the parallelisation of the various MoM solution phases within FEKO.

The first phase in Figure 3.2, namely that of the *Geometry Setup* is, at the time of writing, not yet fully parallelised in FEKO. Preprocessing, such as the triangulation of the surface, the search for common edges between elements, etc., is performed sequentially on a workstation. This data is then written to a file that can be read by the parallel process. According to [28], accelerated techniques, such as spatial decomposition, have also been implemented to speed up this phase.

The second phase, i.e. filling the $N_{RWG} \times N_{RWG}$ matrix, $[Z_{RWG}]$, has been parallelised and requires a distributed storage scheme. Each of the matrix elements, $Z_{RWG}(m, n)$, is a complex entry and therefore occupies the space of two floating point numbers⁴ (for the real and imaginary components respectively). As mentioned in the previous section, the memory requirements for large problems can grow quite substantially as the number of unknowns increase. Problems with unknowns ranging between 30,000 and 60,000, require in total between 7 GByte and 27 GByte of memory.

Computing the matrix elements $Z_{RWG}(m, n)$ in parallel by a number of processes for the impedance matrix is not a trivial task for several reasons, as follows. Firstly, some of the matrix elements need to be stored on the same node. The reason for this being that these matrix elements are computed from certain common integrals. Secondly, in some cases symmetry properties of the model can be exploited. In order to make use of symmetry relations between

⁴When the numbers are stored using single point precision, the matrix element occupies 8 Bytes of memory. This figure increases to 16 Bytes when double point precision is used.

certain matrix elements, they should also preferably be located on the same node to reduce unnecessary interprocess-communication. Finally, the matrix $[Z_{RWG}]$, is in fact composed of different submatrices $[Z_{RWG}^{(i,j)}]$ that correspond to different sections of the geometry, such as metallic wires, PEC surfaces, etc. The CPU-time required to compute each of these submatrices may differ quite substantially (when comparing for instance a 4-dimensional integration for metallic surfaces, with a 2-dimensional integration for metallic wires).

Owing to the above constraints, a *one-dimensional block cyclic row distribution scheme* with a block size, N_B in the range $1 \leq N_B \leq \lfloor \frac{N}{p} \rfloor$ with p equal to the number of nodes, is selected. Briefly, this storage scheme entails that the rows of the matrix, $[Z_{RWG}]$, are divided into groups of size N_B . These groups are then distributed in a cyclic manner amongst the various processors present in the solution. This distributed storage scheme ensures that all the matrix elements of one row are located on the same node, which is important for the reasons discussed in the previous paragraph. Larger values for N_B are preferred, as it was observed that this increases both the performance of efficient matrix-fill techniques, and also that of the matrix solve phase. Load balancing is then achieved by means of a special mapping function, which is introduced in order to exchange rows of the matrix $[Z_{RWG}]$, and elements of the vector $\{V_{RWG}\}$, according to the estimated time required to compute the different rows of the impedance matrix.

For very large problem sizes, i.e. a discretized surfaces with a large number of unknowns, the final phase of solving the dense system of linear equations of Eq. (3.1) dominates the solution time. FEKO uses the ScaLAPACK (or Scalable LAPACK) library that includes a subset of LAPACK [29] routines redesigned for distributed memory MIMD parallel computers [30]. The LAPACK (or Linear Algebra PACKage), provides routines for solving systems of simultaneous linear equations such as LU-decomposition for dense matrices.

In the following section, the computing infrastructure on which the parallelised full-wave MoM solution was analysed, is discussed.

3.3 The *iQudu* - a High Performance Computing (HPC) infrastructure

The computing infrastructure used in the author's research was provided by the Centre for High Performance Computing (CHPC)⁵, housed in Cape Town at the CSIR Rosebank Campus. Initiated in May 2007, the CHPC plays an important role with regards to computational research support and resource supply in Africa. The CHPC is an initiative of the Department of Science and Technology and is managed by the Meraka Institute of the CSIR. The centre operates within a multiple stakeholder and clientele environment and provides a unique HPC resource to researchers in various scientific and engineering disciplines. The computational infrastructure provided by the CHPC is, at the time of writing, primarily that of various IBM cluster implementations, of which the *iQudu* cluster forms part.

The *iQudu* cluster, Figure 3.3 on page 23, consists of a high-speed IBM e1350 Linux cluster with 160 computing nodes. Each of the nodes is equipped with two dual-core AMD Opteron 2.6 GHz processors and 16 GByte of RAM. This aggregate of 640 processors provides the user with a processing power of roughly 2.5 teraflops. In addition to local storage, all the nodes have access to a shared storage system with a capacity of 54 TByte, using a global parallel file system (GPFS). The nodes are connected on a network via one of two interconnects, namely a 1 GBit Ethernet connection and a faster 10 GBit Infiniband connection. Eight of the compute

⁵The information in this section is primarily based on that available from [31]

nodes in the cluster are equipped with Clearspeed Advance e620 Accelerator boards. Each of these boards contains 2 CSX600 processors with 96 processing elements, running at 210 MHz.



Figure 3.3: The iQudu Cluster with 160 computing nodes, housed by the CHPC in Cape Town.

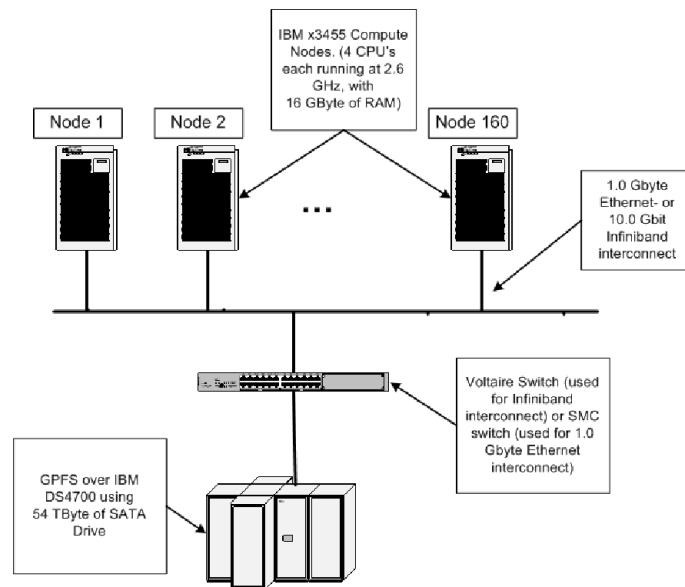


Figure 3.4: Graphical illustration of the iQudu infrastructure.

The iQudu cluster can further be classified according to Flynn's taxonomy [12], which was introduced in Section 2.2, as a MIMD system. This implies that each of the processing elements can operate independently with potentially different instructions on different data. A simplified schematic overview of the CHPC infrastructure is illustrated in Figure 3.4.

With the basics of the parallel MoM implementation in FEKO and the specifics of the CHPC infrastructure presented, the *run-time* of the numerical analysis of general scatterers can now be estimated as explained in the following Section.

3.4 Runtime predictions

In Section 2.2 it was stated that the operation count for the LU-decomposition of a matrix of dimensions $N_{RWG} \times N_{RWG}$ with complex valued entries, is approximately equal to $8/3N_{RWG}^3$ floating point operations. On a system such as the iQudu, which is capable of sustaining 2.5

teraflops with 640 processors, the run-time associated with the LU-decomposition can easily be calculated for various problem sizes, the results of which is depicted in Figure 3.5⁶. The results also include that of one of the nodes of the iQudu cluster, which is capable of sustaining approximately 15.5 gigaflops⁷.

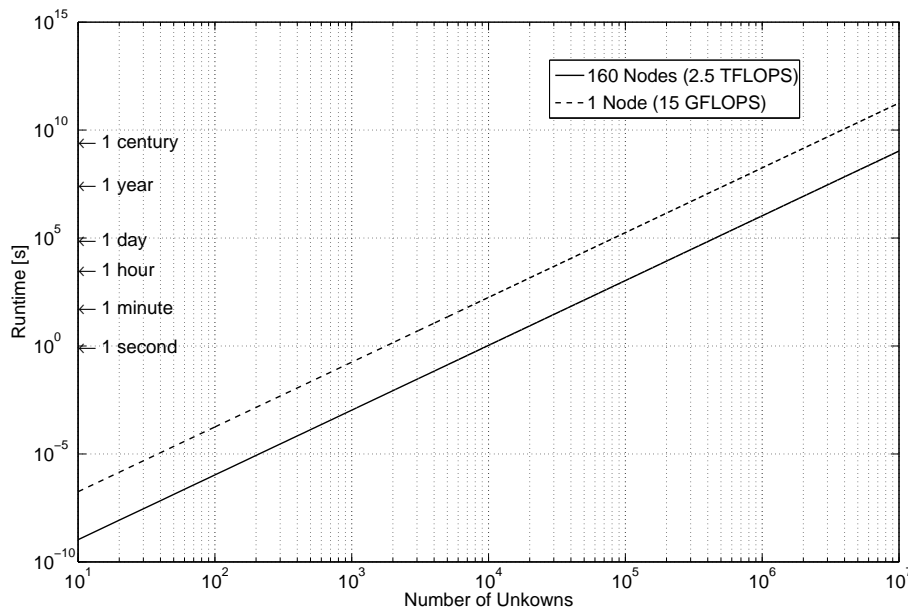


Figure 3.5: Approximate runtimes for LU decomposition on the iQudu cluster with 160 nodes and 1 node respectively.

As noted in the previous Section, one of the iQudu nodes contains two dual-core AMD opteron processors with roughly 16 GByte of RAM, i.e. specifications comparable to that of modern-day desktop workstations at the time of writing. If we compare the approximate runtime of a problem consisting of 10,000 unknowns simulated on 1 node to that simulated on all 160 of the iQudu nodes, we see that the simulation time drops from about half-an-hour to a few seconds, again illustrating the significant potential increase one can expect from using HPC to solve large electromagnetic problems.

From Figure 3.5 the problem sizes for effective benchmarking can also be approximated. If the runtime for the LU-decomposition is to be limited to a maximum of one hour⁸, the number of unknowns is limited to be between 20,000 on one node and 100,000 on 160 nodes. It is however to be noted that this estimate refers to *CPU-time*, and not actual *wall-clock runtime*. The difference between the two times, is that runtime includes overheads such as the CPU-time of other processes when they gained access to the CPU as well as network latency. CPU-time only accounts for the actual CPU-time that a single process was granted. It is therefore reasonable to state that runtime will always be equal to, or larger than the CPU-time. The estimate problem sizes obtained with the above reasoning therefore serves as an upper bound to the benchmarking study.

⁶Note that inherent in this calculation is the assumption that the iQudu operates at 100% efficiency.

⁷The floating point performance of some of the compute nodes on the iQudu cluster was determined with FEKO using the FEKO-MPI-STATISTICS flag. The results indicate that the nodes obtain a performance of roughly 15.3 gigaflop, which corresponds to that reported by the CHPC in [31]. The results obtained by FEKO is for double point precision floating point numbers.

⁸In this context, the simulation is undertaken at a single frequency point of interest.

Before the results of the benchmarking are presented in the remainder of this Chapter, a brief overview is presented in the following Section of the antenna elements that are used for the various simulations.

3.5 The simulated antenna-model

The element selected for the benchmarking simulations is called the *Vivaldi* antenna. By connecting the elements in an array⁹, the size of problems simulated can gradually be increased, as illustrated in Figure 3.6(b). The reason for selecting the Vivaldi antenna array as the model for the benchmarking simulations, is that it forms a crucial part of SKA research at institutions such as the Netherlands institute for radio astronomy (ASTRON) [24] and is therefore a practical example of a FPA configuration. In addition to this, results from a previous study performed by the author in 2007 [6] could also be used to validate that obtained from the iQudu cluster.

The number of unknowns associated with each of the simulated array models, is summarised in Table (3.1). In the following section, the benchmarking criteria is defined.

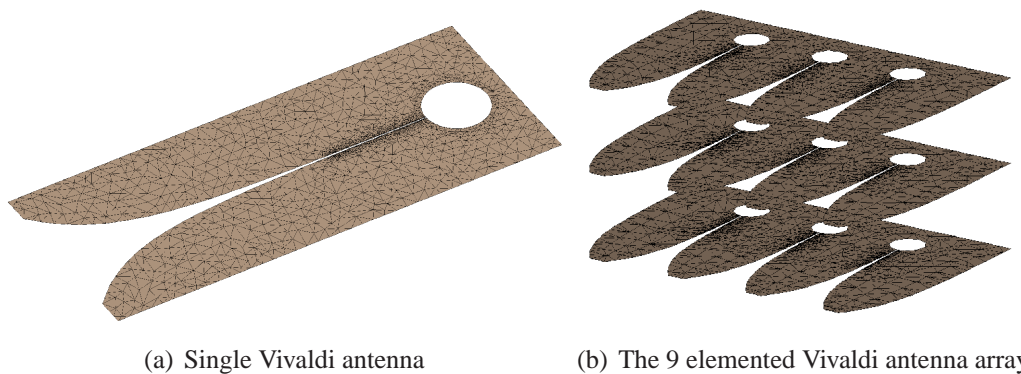


Figure 3.6: Example of a single Vivaldi element and a 9 element Vivaldi array modelled with FEKO.

Table 3.1: Number of unknowns associated with each of the simulated Vivaldi antenna arrays.

Number of Vivaldi Antennas in Array:	1	8	16	32	64
Number of Unknowns	550	4,368	8,736	17,472	33,624

3.6 Benchmarking criteria

Two parameters are used as the primary benchmarking criteria in this study, namely that of *speed-up* and *efficiency*. In [32] speedup is described as an indication of how much faster an algorithm will run on N processors, compared to one; something that is of fundamental importance to the user. In [32] speedup was defined as "*The ratio of time taken by an equivalent serial algorithm running on one processor, T_s , to the time taken by the parallel algorithm using N processors, T_p* ",

⁹The antenna models forming the basis of the design was created with *CADFEKO*, a graphical package forming part of the FEKO suite.

$$S = \frac{T_s}{T_p} \quad (3.2)$$

The speed-up, S , typically has an upper bound of N . Here, speed-up is defined slightly differently to that described by Eq. (3.2), the reason for which is as follow: When one considers that with the arrival of multi-core processors, modern day computer architecture seldomly consists of only one processor anymore. Typical cluster configurations also consist of SMP nodes, connected by high-speed interconnects. It was therefore decided that instead of using the equivalent time taken by a serial algorithm allocated a single processor, it makes more sense to use the time taken by the algorithm on a single *node*, i.e. with four processors, T_{N_1} (in the case of the iQudu cluster).

In this study, speed-up is therefore defined as *the ratio of time taken by the parallel algorithm running on one node (with four processors), T_{N_1} , to the time taken by the same algorithm using X nodes, T_{N_x} ,*

$$S = \frac{T_{N_1}}{T_{N_x}} \quad (3.3)$$

The other primary benchmarking parameter in this study, namely efficiency, is an indication of how efficiently the X processors are being used by the parallel algorithm. It is defined as the speed-up normalised by the number of nodes, X ,

$$\epsilon = \frac{S}{X} \quad (3.4)$$

Combined, the above criteria reflects the *scalability* of the algorithm, i.e. the degree to which it can be parallelised. Before presenting the results in terms of the benchmarking criteria defined in the preceding paragraphs, it necessary to first discuss the impact of the various iQudu interconnects on the absolute simulation runtimes.

3.7 The effect of interconnects on simulation runtimes

As stated in Section 3.3 on page 22, the iQudu infrastructure supports two different interconnects, i.e. a 1 GBit Ethernet and a 10 GBit Infiniband network. In Figure 3.7 on page 27, the absolute runtimes of a 32 Vivaldi array simulated on various number of nodes on both the Ethernet and Infiniband interconnect, are presented.

When comparing the results obtained by using the different networks, it is evident that network latency severely influences the performance of the simulation in terms of absolute runtime. Furthermore, it can be seen that the runtime improves by almost a factor of 2 when using the faster Infiniband interconnect. The improvement in simulation time also increases as more nodes are included in the simulation. The reason for this is that as the processing pool becomes larger, the degree of inter-process communication increases, and plays a more prominent role in the simulation run-time associated with the Ethernet interconnect as compared to that of the faster Infiniband network.

The primary difference between the two interconnects, is that Infiniband does not use the same communication stack as conventional Ethernet. Infiniband network traffic occurs via specialised Infiniband adapters, characterised by a much lower communication overhead.

In the following section, the benchmarking results, as pertaining to that defined in Section 3.6, are presented. In view of the merits of using a faster interconnect as discussed in this Section, the Infiniband network was used to obtain the benchmarking results.

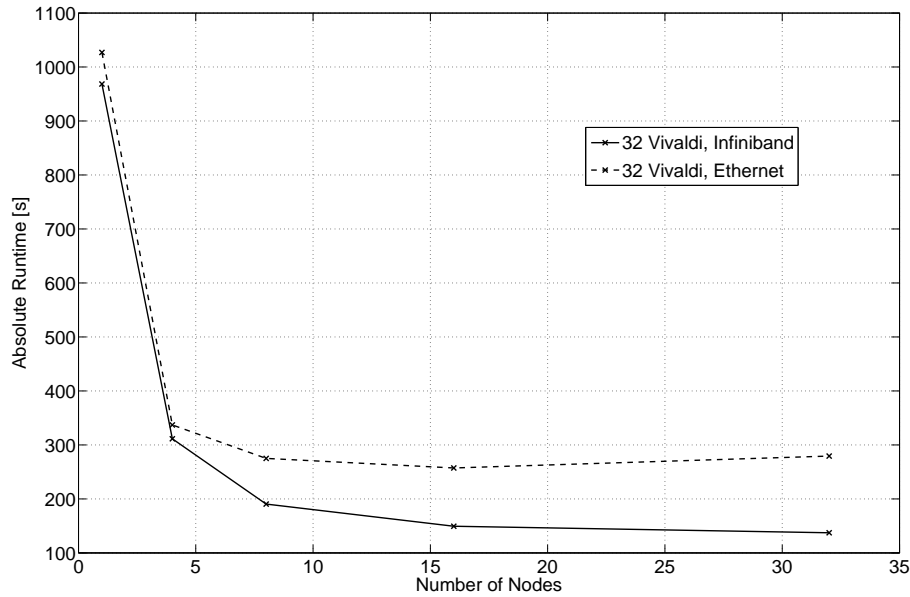


Figure 3.7: Absolute run-times of a 32 element Vivaldi Array (17,472 unknowns) simulated on various number of nodes by using a 1 Gbit Ethernet and a 10 GBit Infiniband interconnect.

3.8 Benchmarking results using the Infiniband interconnect

In the following Subsections, the results of the speed-ups and efficiencies are plotted against the number of nodes as well as *grain-size*¹⁰.

3.8.1 Speed-up versus the number of nodes

The runtime speedup versus the number of nodes is illustrated in Figure 3.8. For comparison, the ideal speed-up, i.e. $S = X$ where X is the number of nodes used in the simulation, is included in the results. From the results illustrated in Figure 3.8 it can be observed that the speed-up follows the ideal case for a limited number of nodes, until $X \approx 5$. When the number of nodes is increased beyond this point, the measured data tends to deviate from the ideal-case quite significantly and tends toward a steady-state value which ranges between $S = 5$ and $S = 12$ for problem sizes between 6,322 and 24,384 unknowns. The final-values also increase as the problem size is increased. The aforementioned observations, can be explained as follows:

The behaviour of the measured speed-up results is consistent with *Amdahl's law*, as explained in [32]. Amdahl's Law states that if an algorithm contains both a serial and a parallel part, the relative time taken by the serial part increases as parallelisation reduces that of the parallel part. Mathematically, this can be expressed by saying that ζ percent of the algorithm cannot be parallelized, while the remaining $(1 - \zeta)$ percent is perfectly parallelizable. Neglecting the effect of any inter-process communication overhead, Amdahl's speedup model can be expressed as follows,

$$S = \frac{T_1}{[\zeta + (1 - \zeta)/P] T_1} = \frac{1}{\zeta + \frac{1-\zeta}{P}} \quad (3.5)$$

¹⁰According to [1], the efficiency of many parallel algorithms is a function of *grain-size* - the number of unknowns per processor, or in this case, node. For many parallel algorithms, the efficiency remains fairly constant once a particular grain-size has been reached, as will be evident from the results

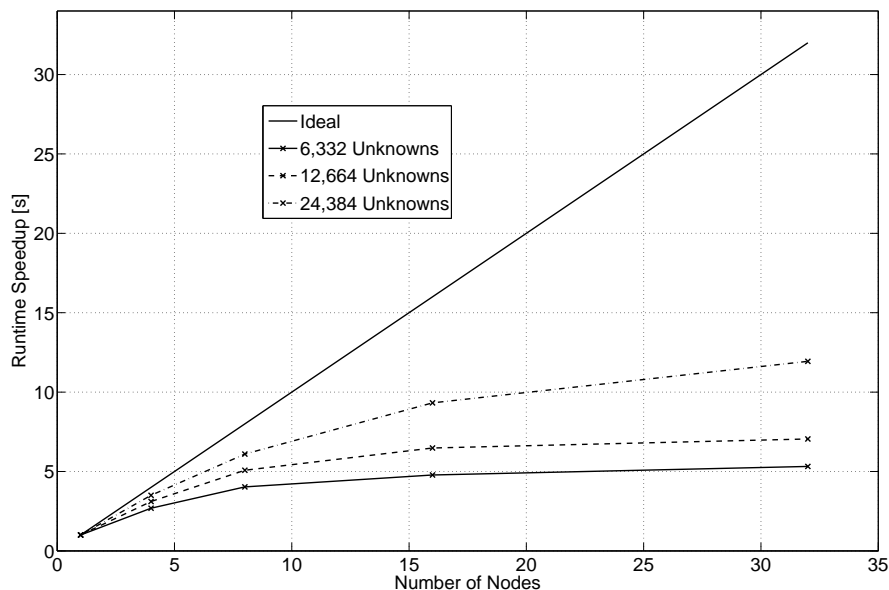


Figure 3.8: Run-time speedup vs. the number of nodes, as measured on the iQudu cluster when using the Infiniband interconnect

where T_1 is the time taken by the algorithm on one process/node. The term, $[\zeta + (1 - \zeta)/P] T_1$, is therefore the fraction of time taken by the algorithm on P nodes. Practically, this means that if 90% of an algorithm can be parallelized, the run-time will always be limited by the remaining 10% that is required to be executed sequentially. If one however increase the degree of parallelisation by considering larger problem size, it has been observed that ζ decreases, leading to a better speed-up performance, as is can be observed in the results [8].

In addition to the above, the measured results also include the effect of interprocess communication. Communication time is typically characterised by two parameters, namely that of latency and bandwidth. Bandwidth can be defined as the rate at which data is transmitted between two parallel processes on a given communication path in unit time. The communication path can be an interconnect between distributed compute nodes in a cluster environment for instance or the connections between the cores of a SMP. Latency can be viewed as the time it takes for an MPI message of effectively zero length to leave one node, traverse the communication stack and the network link, and arrive on the receiving node [33]. Latency can therefore be influenced by the initializations performed in the particular MPI routines. Bandwidth is primarily determined by the interconnect being used, as was illustrated in Section 3.7 on page 26, as well as the algorithm used in the MPI communication routine.

To illustrate the effect of latency and bandwidth for MPI implementations pertaining to (a) a compiled language (such as that used in FEKO) and also (b) interpreted programming languages such as Octave and Python, the performance of a typical point-to-point MPI communication routine, MPI-Send, is measured using a round-robin (ping-pong) approach [34]. In a ping-pong program the total time it takes to send n MPI messages back and forth between two processes is measured. This time is then divided by $2n$ to get the average time associated with a single message travelling from one process to another [33].

In Figure 3.9 the average communication times of a ping-pong program implemented with an MPI binding for Python (mpi4py) and one for Matlab/Octave (MPITB) respectively are illustrated. The communication time associated with a pure C binding, namely Open-MPI, is added for comparison. Arrays of sizes ranging between 10 MByte and 100 MByte were used as

the communication messages. The C, Python and Octave ping-pong implementations are listed in Appendix B on page 82. In terms of the MoM, this corresponds to problems sizes ranging from roughly 800 unknowns to about 2,500 unknowns.

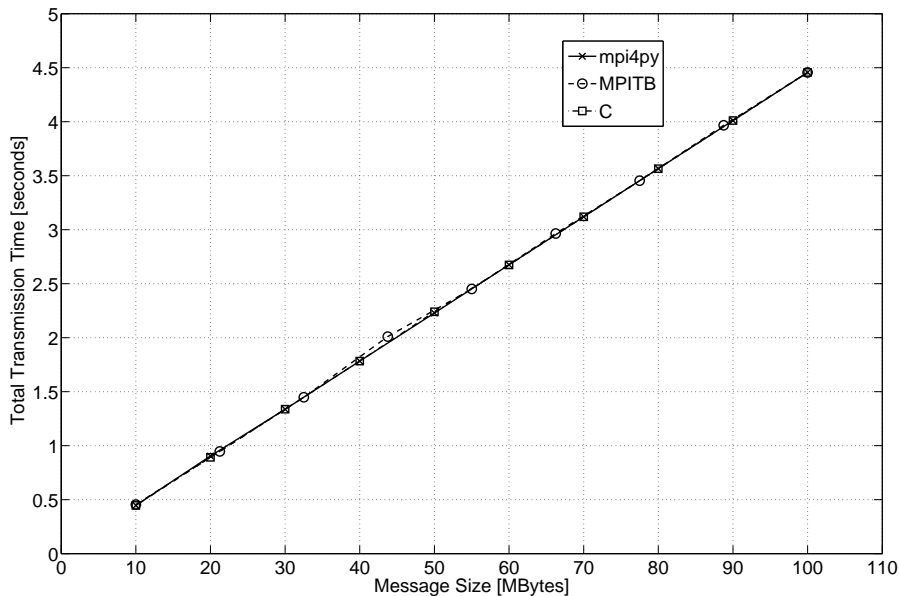


Figure 3.9: The average transmission time in seconds, associated with messages of arbitrary size during a Ping-pong test implemented with various MPI bindings using a GBit Ethernet interconnect.

From the results it is evident that in this region a linear relationship exists between the average communication time and the message length, i.e.

$$T = \frac{1}{b}t_s + t_l \quad (3.6)$$

where b is the bandwidth and t_l the latency [34]. A linear least squares fit was applied to the data to obtain the unknown parameters in equation (3.6) the results of which are illustrated in Table 3.2.

MPI binding	Bandwidth (b) [MByte/s]	Latency (t_l) [msec]
C	22.45	2.3
mpi4py	22.47	4.7
MPITB	22.47	14

Table 3.2: Measured bandwidth and latency for various MPI implementations.

A bandwidth of roughly 22.4 MByte per second is obtained for each of the MPI implementations. The latencies associated with mpi4py and MPITB are higher than that of the C binding and can be ascribed to the interpreted environment in which these MPI-implementations are used. The results emphasise that interprocess communication is influenced by the underlying

MPI-binding being used¹¹ and that for larger message sizes, the total transmission time can significantly impact the total runtime results.

In the following section, the parallel capabilities of FEKO are analysed in terms of *efficiency*.

3.8.2 Efficiency versus the number of nodes

The runtime efficiency versus the number of nodes is illustrated in Figure 3.10. As stated in Section 3.6 on page 25, *efficiency* is an indication of how efficiently the nodes included in the parallel simulation are being used by the parallel algorithm. Efficiency thereby reflects the so-called load-balance, or contribution of process to the parallel execution time. Mathematically this can be expressed by combining Eq. (3.3) and Eq. (3.4) in Section 3.6 as follow,

$$\epsilon = \frac{S}{X} = \frac{T_{N_1}}{XT_{N_x}} \quad (3.7)$$

Ideally, the efficiency should be equal to 100% when each of the processes included in the parallel pool contributes more or less equally to the execution of the algorithm. Like speedup, efficiency is influenced by the serial section of the algorithm typically performed sequentially on a single process and also by the time spent for interprocess communication.

When investigating the results in Figure 3.10 it is evident that for a given problem size, the efficiency deviates from the ideal case, i.e. $\epsilon = 100\%$, when the number of nodes is increased. This is attributed to the non-linear relationship between the speedup and the number of nodes, as explained in the previous subsection and illustrated in Figure 3.8. From Figure 3.10 it is also evident that an increase in the problem size again improves the efficiency obtained by FEKO on a given number of nodes.

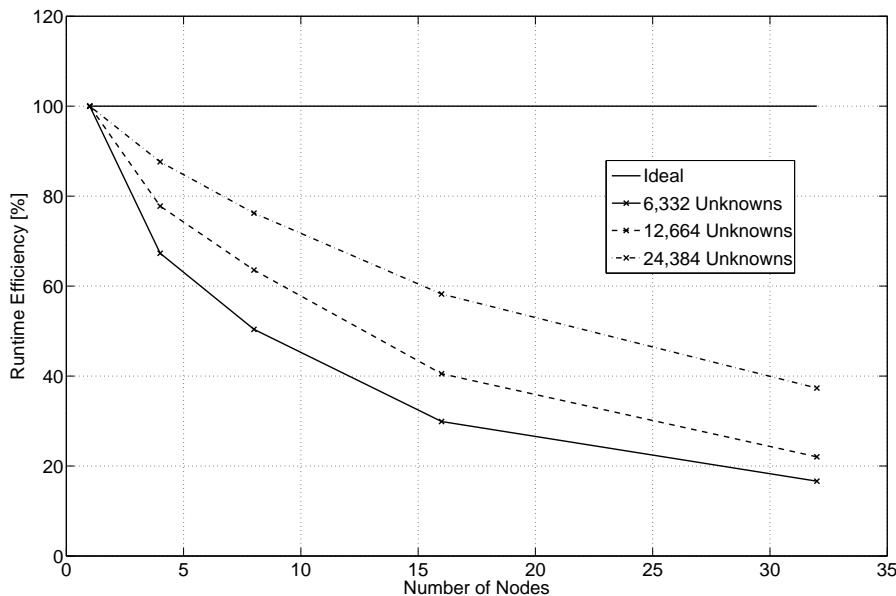


Figure 3.10: Run-time efficiency vs. the number of nodes, as measured on the iQudu cluster when using the Infiniband interconnect

¹¹In Chapter 5, Section 5.3 on page 72 the MPI-bindings for mpi4py and Python will be used in a parallel implementation of the CBFM formulation.

In the following Subsection, efficiency is expressed as a function of a parameter called *grain-size*.

3.8.3 Efficiency versus *Grain-size*

The runtime efficiency versus the number of unknowns divided by the number of nodes, i.e. *grain-size*, is illustrated in Figure 3.11. From the results illustrated it is clear that the efficiency remains approximately constant once a certain grain-size is reached. This grain-size value corresponds to $\frac{N}{P_N}|_{CPU} \approx 6,332$, at which the algorithm obtains an efficiency of $\epsilon \approx 94\%$. These values are of interest, as they can be used as an effective method of determining the "behaviour" of the speed-up for a given problem size *before* the simulation is started. When operating at a grain-size *above* the value of $\frac{N}{P_N}|_{CPU} \approx 6,332$, one is ensured that the speed-up will increase *linearly* with the number of nodes. Alternatively, by operating *below* the previously mentioned grain-size, the speed-up will enter it's non-linear region and tend towards "saturation" as discussed and illustrated in the previous sections. The implication of this, is that although one might still observe a slight decrease in the *absolute simulation runtime*, parallel resources are in-fact not being optimally used.

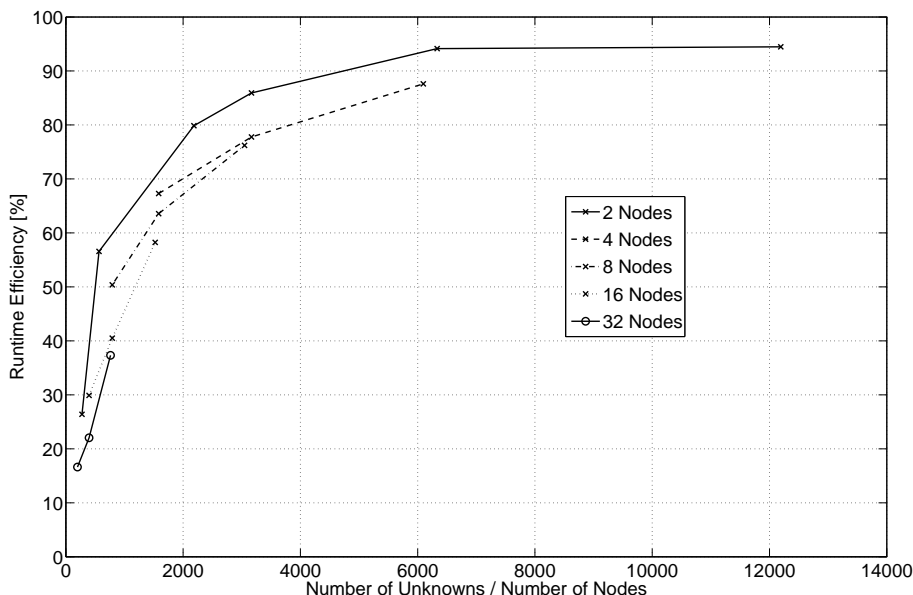


Figure 3.11: Run-time efficiency vs. the number of unknowns divided by the number of nodes (*grain-size*), as measured on the iQudu cluster when using the Infiniband interconnect

3.8.4 Memory usage

In Figure 3.12 the memory requirements of the full-wave MoM algorithm are illustrated. The results include calculated data (based on whether the complex impedance matrix, $[Z_{RWG}]$, are stored in single- or double point precision) as well as data from various Vivaldi array simulations. In all the cases, a quadratic dependency is observed as expected due to the $O(N_{RWG})^2$ storage requirement for the impedance matrix. The impact of the memory requirements of the full-wave MoM solver as implemented in FEKO, plays a substantial role regarding the allocation of parallel resources to solve a given problem, as illustrated by the following example.

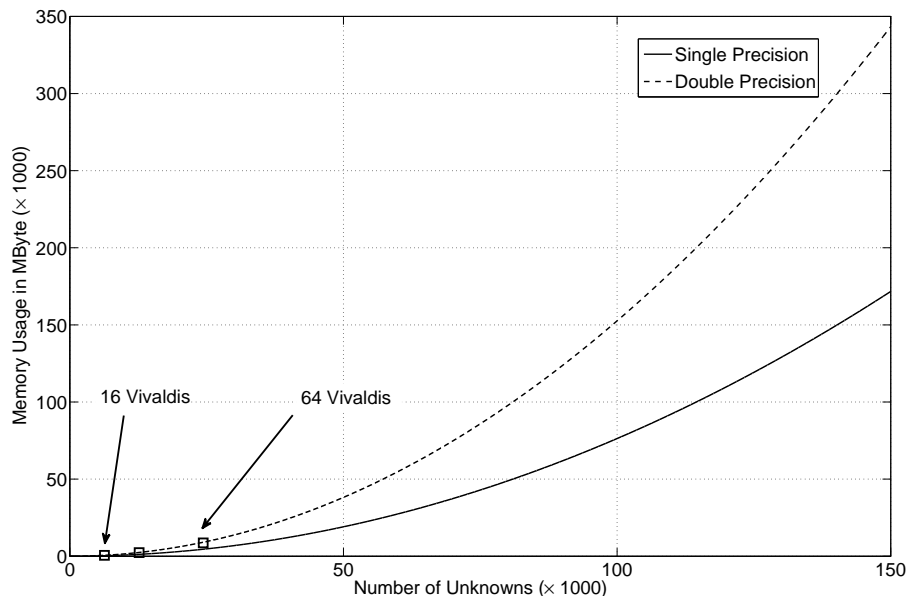


Figure 3.12: Calculated- and recorded memory usage for various number of unknowns associated with the MoM implementation in FEKO.

Consider for instance that one wishes to simulate a problem with N_{RWG} equal to 150,000. We can calculate that the memory requirements of such a problem corresponds to approximately 167 GByte (when using single-point precision storage for the impedance matrix entries). Using for example 8 computing nodes, and assuming that the data is divided more or less equally between them, it can be determined that each node will have to store roughly 21 GByte of data. This is more than the amount of RAM on each of the iQudu nodes. The result of this is that the nodes will have to access the GPFS as the algorithm resorts to the much slower *out-of-core* solution [27]. In this case it would be better to limit the number of nodes to be such that each acquires enough data to utilise the on-board RAM of the node efficiently. By limiting the maximum amount of on-board memory available to 15 GByte in the calculation¹², one can determine a reasonable (minimum) value for the number of nodes to use. In the case of the above example where $N = 150,000$ that requires 167 GByte of RAM, using 12 Nodes instead of 8, will resort in a high efficiency with a faster *in-core* solution. With the number of nodes and the problem-size known, one can then use the results in Figure 3.10 to determine the efficiency as well as the behaviour of the speed-up, to ensure that the parallel resources are utilised effectively.

3.9 Conclusions

In this Chapter, the parallel MoM implementation in FEKO was investigated. The emphasis was placed on simulating "large" electromagnetic problems, such as various sized Vivaldi FPAs illustrated in Section 3.5 on page 25, on a various number of nodes of the iQudu HPC cluster.

In summary, the reader was introduced to the MoM and then provided with a brief overview of the parallelisation of this CEM technique in the software package FEKO. The reader was then

¹²It is to be noted that each node still have to use a certain amount of RAM for the operating system processes.

familiarised with the iQudu cluster housed at the CHPC. In Section 3.4, runtime predictions were made by considering that the *LU-decomposition* phase in the solution (in the case of using a direct solver) dominates the solution time. These runtime-predictions also served to emphasise the significant (theoretical) improvement in simulation time that can be obtained by using a parallel cluster.

With the Benchmarking criteria defined in Section 3.6, the effect of various levels of parallelisation and problem sizes on *speedup*, *efficiency* and *memory usage* were investigated. These parameters also proved useful in determining the optimal number of nodes to use when attempting to simulate an electrically large MoM problem.

From the results illustrated in this Chapter, it is evident that High Performance Computing plays a significant role in the simulation of electrically large FPA structures that may be used as feed-structures for parabolic reflector antennas in the SKA and the MeerKAT. CEM software packages such as FEKO also simplify the design process by providing the antenna designer with an easy to use interface and an efficient parallelisation of the full-wave MoM solver.

Unfortunately the control parameters available when using HPC resources with a package such as FEKO, are limited to selecting the number of nodes, type of interconnect and by adjusting the storage method (i.e. single or double-point precision). If one considers ways to further improve the simulation of electrically large FPA structures, based on the MoM, it is however necessary to investigate better solving techniques for the matrix equation (Eq. (2.2)) introduced in Section 2.3 on page 8. One such a technique is the *Characteristic Basis Function Method* (CBFM), which will be introduced formally in Chapter 5 of this thesis. To incorporate this technique into the MoM formulation of FPA structures it is however necessary to gain more insight into the underlying implementation of the MoM algorithm, which is the focus of the following Chapter.

Chapter 4

A Method of Moments Formulation in 3D

This section will be a formal overview of the Method of Moments (MoM) as well as a discussion of the author's MoM based package, *GMoM*. The focus is directed to three dimensional perfectly conducting radiators and scatterers and the underlying theory is based on that presented by Rao, Wilton and Glisson in [2]. Results generated by *GMoM* will be compared to that obtained by FEKO and results that can be found in the literature.

This Chapter is structured as follows: In the following Section, the electric field integral equation (EFIE) is presented, that relates the induced surface current distribution on a conducting body to the scattered electric field by applying certain boundary conditions. To solve the integral equation by the application of the MoM, a set of expansion coefficients, namely the RWG basis functions are introduced, that together with an appropriate testing-procedure are used to derive the elements of the MoM matrix equation presented in Section 2.3, Eq. (2.2). The second half of the Chapter will be concerned with extending this MoM formulation to incorporate radiating elements such as dipole antennas and the Vivaldi element as a step towards formulating an efficient solution technique for FPA configurations for the SKA and the MeerKAT. Numerical results will be presented for the various structures, and will be compared to that obtained by FEKO and also that available in [2].

4.1 The Rao-Wilton-Glisson (RWG) MoM formulation for arbitrary 3D scatterers and radiators

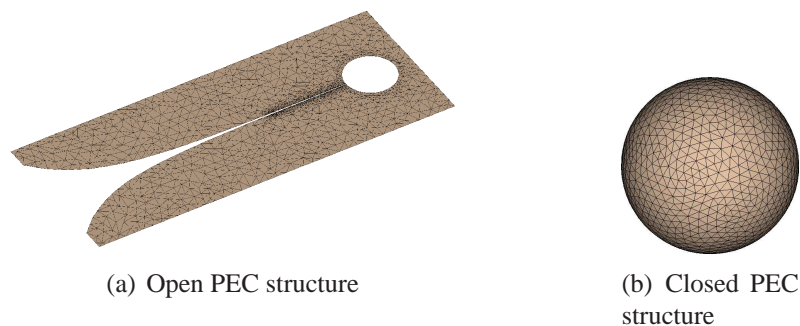


Figure 4.1: Example of an (a) open and (b) closed PEC structure, modelled with triangular patches

As stated in Section 2.1 on page 3, the underlying goal of the frequency based MoM approach is to obtain a mathematical expression for the discretized current distribution on an ar-

bitrary electromagnetic scatterer or radiator. In Figures 4.1(a) and 4.1(b) examples of so called *open* and *closed* PEC structure are presented. The structures are sub-divided into triangular patch elements, which are capable of accurately conforming to nearly any geometrical surface as stated in [2]. Integral equations can then be formulated for the currents flowing on the surface of the structure. The type of equation used is dependant on whether the structure is closed or open [1]. For closed structures, the magnetic field integral equation (MFIE) can be used which in general leads to more well-posed equations. In this context, a well-posed problem is one whose solution is not strongly dependant on the physics and geometry of the problem, as stated in [1]. In arbitrary surface modelling however, the EFIE has the advantage of being applicable to *both* open and closed bodies, and is therefore selected as the underlying formulation in many practical algorithms¹, including the author's formulation GMoM. In the following subsection, the EFIE formulation will be presented based on that discussed in [2].

4.1.1 The electric field integral equation (EFIE)

Consider an electric field, \mathbf{E}^i , incident on the PEC structures depicted in Figures 4.1(a) and 4.1(b). This field is known as the *incident* electric field, and is defined as that due to an impressed source in the absence of the scatterer. This incident field induces a surface current, \mathbf{J}_{RWG} , on the PEC structure. If we consider an open structure, such as that depicted by Figure 4.1(a), \mathbf{J}_{RWG} represents the vector sum of surface currents on opposite sides of the surface, S . In this case, the normal components of the current, \mathbf{J}_{RWG} , must vanish on the boundaries of the problem domain. The scattered electric field, \mathbf{E}^s , can then be computed from the induced surface current as follows,

$$\mathbf{E}^s = -j\omega\mathbf{A} - \nabla\Phi \quad (4.1)$$

where ∇ is the gradient operator and is given by,

$$\nabla = \frac{\partial}{\partial x}\hat{\mathbf{x}} + \frac{\partial}{\partial y}\hat{\mathbf{y}} + \frac{\partial}{\partial z}\hat{\mathbf{z}} \quad (4.2)$$

with $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ representing the unit vectors in the x, y and z direction respectively. The vector quantity, \mathbf{A} , is the so called *magnetic vector potential*, and is defined as,

$$\mathbf{A} = \frac{\mu}{4\pi} \int_S \mathbf{J}_{\text{RWG}} \frac{e^{-jkR}}{R} dS' \quad (4.3)$$

The scalar quantity, Φ , in Eq. (4.1), is the *scalar potential* and is defined as,

$$\Phi = \frac{1}{4\pi\epsilon} \int_S \sigma \frac{e^{-jkR}}{4\pi R} dS' \quad (4.4)$$

where $k = \omega\sqrt{\mu\epsilon} = 2\pi/\lambda$ is known as the wavenumber, with λ being the wavelength. The quantities ϵ and μ are the permittivity and permeability of the surrounding medium respectively. The free-space scalar Green function is present in the above terms and may be expressed as follows,

$$G(\vec{r}, \vec{r}') = \frac{e^{-jkR}}{R} \quad (4.5)$$

¹It is however to be noted that linear combinations of the MFIE and EFIE known as the combined field integral equation (CFIE) are also available, which are available in packages such as FEKO.

where the term $R = |\vec{r} - \vec{r}'|$ is the distance between an arbitrarily located source point \vec{r}' and observation point \vec{r} on the structure. It is important to note that both \vec{r}' and \vec{r} are defined with respect to a *global* coordinate origin. The scalar quantity, σ , is the surface charge density and is related to the surface divergence of the current density, \mathbf{J}_{RWG} , through the continuity equation,

$$\nabla_S \cdot \mathbf{J}_{\text{RWG}} = -j\omega\sigma \quad (4.6)$$

If one considers the boundary condition for the tangential component of the electric field on the PEC surface of S ,

$$\hat{\mathbf{n}} \times \mathbf{E}^{\text{tot}} = \hat{\mathbf{n}} \times (\mathbf{E}^i + \mathbf{E}^s) = 0 \quad (4.7)$$

where $\hat{\mathbf{n}}$ is the unit normal for S , the following integro-differential equation² for \mathbf{J}_{RWG} can be enforced,

$$-\mathbf{E}_{\text{tan}}^i = (-j\omega\mathbf{A} - \nabla\Phi)_{\text{tan}}, \text{ with } r \text{ on } S \quad (4.8)$$

Equation (4.8), with (4.3) to (4.6), forms the electric field integral equation (EFIE) that relates the *unknown* induced surface current, \mathbf{J}_{RWG} , to the *specified* incident field, \mathbf{E}^i . The first step in obtaining a solution for Eq. (4.8), is to represent the unknown surface current distribution with a set of basis-functions, as discussed in the following subsection.

4.1.2 The RWG basis-functions

The set of basis-functions used to model the surface current distribution, \mathbf{J}_{RWG} , needs to be suited to both the triangular patch elements used for the geometry representation, as well as the EFIE used for the underlying formulation. The basis-functions selected for this, is known as the Rao-Wilton-Glisson (RWG) basis-functions, and has been mentioned throughout this thesis. In this subsection, these functions will be investigated in more detail.

Before continuing, it might be useful to note that the primary role or outcome of the RWG basis-functions is to obtain a suitable approximation for the current distribution on S , i.e.,

$$\mathbf{J}_{\text{RWG}} \cong \sum_{n=1}^{N_{\text{RWG}}} I_n \vec{f}_n(\vec{r}) \quad (4.9)$$

with I_n the entries for the vector quantity, $\{I_{\text{RWG}}\}$, present in Eq. (2.2) on page 8, and $\vec{f}_n(\vec{r})$ the RWG basis function respectively. The meaning of each quantity in Eq. (4.9) will be discussed in the remainder of this Section.

The starting point of the development, is to note that each basis function, $\vec{f}_n(\vec{r})$, is associated explicitly with an *interior* edge (i.e. a non-boundary edge) of a triangular patch as is illustrated in Figure 4.2. The basis-function pertaining to this edge, n , is zero on every other triangular patch, except for the two triangles attached to edge n , i.e. T_n^+ and T_n^- . The points in any of the triangles may be designated either by the position vector, \vec{r} , with respect to a global coordinate origin, or by the local position vector, ρ^{\pm} , which is defined with respect to the free-vertex of T_n^{\pm} respectively as illustrated in Fig. (4.2). The plus and minus sign associated with each of the triangles, is used to define the reference direction for the RWG element associated with edge n , which is assumed to be from T_n^+ to T_n^- . With the previous definitions, the vector basis function associated with the n th edge may then be defined as follows,

²An integro-differential equation is an equation that contains the derivative of the unknown quantity, in this case the current distribution, inside the integral. In [18] it is however mentioned that these equations are mostly referred to as integral equations.

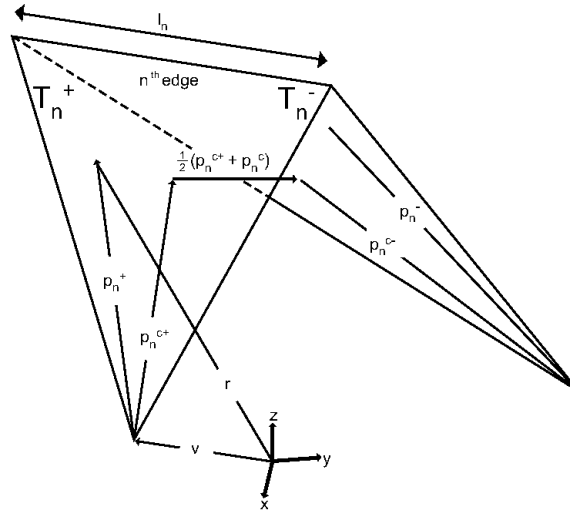


Figure 4.2: The Rao-Wilton-Glisson (RWG) basis function [2]. The diagram illustrates a triangle pair forming a surface that shares an internal (i.e. non-boundary) edge.

$$\vec{f}(\vec{r}) = \begin{cases} \frac{l_n}{2A_n^+} \vec{\rho}_n^+ & \vec{r} \text{ in } T_n^+ \\ \frac{l_n}{2A_n^-} \vec{\rho}_n^- & \vec{r} \text{ in } T_n^- \\ 0 & \text{otherwise} \end{cases} \quad (4.10)$$

where l_n is the length of the n th edge, A_n^\pm is the area of triangle T_n^\pm and $\vec{\rho}^\pm$ as defined as in Fig. (4.2). Representing the current distribution associated with the n th edge according to the RWG formulation presented in Eq. (4.10), is advantageous for several reasons [2] as summarised below:

1. The current component normal to the boundary formed by triangles T_n^+ and T_n^- is zero, which means that no fictitious line-charge exists along this boundary.
2. The current component normal to edge n is constant and continuous across this edge. This is ensured by the normal component of the vectors $\vec{\rho}^\pm$ being equal to the inverse of the coefficients of $\vec{f}(\vec{r})$, i.e. $2A_n^\pm/l_n$ respectively. This ensures that the n th edge is also free of fictitious line-charge densities and hosts a normal current density component of unity.
3. The surface charge density that is proportional to the surface divergence of the vector function, $\vec{f}(\vec{r})$, can be calculated as,

$$\nabla \cdot \vec{f}(\vec{r}) = \begin{cases} \frac{l_n}{2A_n^+} & \vec{r} \text{ in } T_n^+ \\ \frac{l_n}{2A_n^-} & \vec{r} \text{ in } T_n^- \\ 0 & \text{otherwise} \end{cases} \quad (4.11)$$

The charge density is therefore constant in each triangle with the *total* charge associated with triangles T_n^\pm being zero.

Reviewing Eq. (4.9), i.e. the expression for the surface current distribution, it is now evident that a vector basis function is associated with each of the N_{RWG} non-boundary edges. Furthermore, the coefficients I_n may then be interpreted as the normal component of current

density flowing over the n th edge. The reason for this is that for any given non-boundary edge n , only the basis function $\vec{f}_n(\vec{r})$ has a normal component of current flowing over that edge and, as specified in 2), that component is unity.

A final important observation that can be made with regards to the RWG vector element, is that these basis functions are essentially independent of each other in each of the triangles, as the quantity I_n in Eq. (4.9) is an independent quantity for each of the non-boundary edges. This is an important observation when considering how each of the I_n components will be calculated, which forms the core focus of the following Section.

4.1.3 The Galerkin testing procedure

At this stage, it is useful to summarise the MoM process followed thus far, by noting that by substituting the expression for the current-distribution modelled according to Eq. (4.9) into the EFIE equations presented by Eq. (4.8), with (4.3) to (4.6), one is left with an equation that contains N_{RWG} unknowns. Solving these equations according to the MoM, entails that one incorporates the use of so-called testing functions, the role of which is to generate N_{RWG} independent equations that can be used to obtain the N_{RWG} unknowns. This is also known as a so-called testing procedure. By choosing as testing functions the expansion functions \vec{f}_n developed in Section 4.1.2, one is applying the well-known *Galerkin's method* as discussed in [18]. By defining the inner-product between vectors and/or matrices \mathbf{f} and \mathbf{g} as

$$\langle \mathbf{f}, \mathbf{g} \rangle = \int_S \mathbf{f} \cdot \mathbf{g} \quad (4.12)$$

the expression for testing the EFIE equation, Eq. (4.8), with \vec{f}_m where $m = 1, \dots, N_{RWG}$ can be written as follows,

$$\langle \mathbf{E}^i, \vec{f}_m \rangle = j\omega \langle \mathbf{A}, \vec{f}_m \rangle + \langle \nabla \Phi, \vec{f}_m \rangle \quad (4.13)$$

According to [2], the last term in Eq. (4.13) can be expressed as,

$$\langle \nabla \Phi, \vec{f}_m \rangle = - \int_S \Phi \nabla_S \cdot \vec{f}_m dS \quad (4.14)$$

By further utilising the expression for the surface divergence of \vec{f}_m , i.e. Eq. (4.11), the integral in Eq. (4.14) may be rewritten as follows,

$$\int_S \Phi \nabla_S \cdot \vec{f}_m dS = l_m \left(\frac{1}{A_m^+} \int_{T_m^+} \Phi dS - \frac{1}{A_m^-} \int_{T_m^-} \Phi dS \right) \quad (4.15)$$

By approximating the average of the scalar potential over each triangle, T_n^\pm , as the value of the Φ at the centre of the triangles, Eq. (4.15), may be simplified as follows,

$$l_m \left(\frac{1}{A_m^+} \int_{T_m^+} \Phi dS - \frac{1}{A_m^-} \int_{T_m^-} \Phi dS \right) \cong l_m \left[\Phi(r_m^{c+}) - \Phi(r_m^{c-}) \right] \quad (4.16)$$

with \vec{r}_m^{\pm} as illustrated in Figure 4.2.

Similarly, the first term in Eq. (4.13) can be expressed as,

$$\langle \nabla \mathbf{E}^i, \vec{f}_m \rangle \cong \frac{l_m}{2} \left[\mathbf{E}^i(r_m^{c+}) \cdot \vec{\rho}_m^{c+} - \mathbf{E}^i(r_m^{c-}) \cdot \vec{\rho}_m^{c-} \right] \quad (4.17)$$

and the second term, i.e. the testing of the magnetic vector potential, as,

$$\langle \nabla \mathbf{A}, \vec{f}_m \rangle \cong \frac{l_m}{2} \left[\mathbf{A}(r_m^{\vec{c}^+}) \cdot \vec{\rho}_m^{c^+} - \mathbf{A}(r_m^{\vec{c}^-}) \cdot \vec{\rho}_m^{c^-} \right] \quad (4.18)$$

The approximation made in Equations (4.16) to (4.18) eliminates the integration over the testing domains defined by triangles T_m^\pm , allowing a double integration to be approximated by a single integration in the source domain, defined by triangles T_m^\pm respectively. As explained in [2], these approximations can be justified by noting that the potentials are locally smooth within each of the sub-domains (defined by the triangle pair T_n^+ and T_n^-) which follows from their integral definitions and the locally smooth nature of the source representation in terms of the RWG basis function introduced in the previous Section. Unfortunately however, this approximation causes the resulting impedance matrix, Z_{RWG} , which will be discussed in the following section, to lose its symmetrical properties, i.e. $Z_{RWG}(m, n) \neq Z_{RWG}(n, m)$ as would be the case if the double integration over the testing/observation and source region was carried out³.

4.1.4 Derivation of the MoM matrix equation

By using the discretized representation for the current density, expressed by Eq. (4.9) with the weighted EFIE equation⁴ representation, i.e. Eq (4.13) with the approximation made in Equations (4.16) to (4.18), one obtains the $N_{RWG} \times N_{RWG}$ MoM matrix equation first presented in Section 2.2 on page 8, i.e.

$$[Z_{RWG}] \{I_{RWG}\} = \{V_{RWG}\} \quad (4.19)$$

where the elements of the impedance matrix, $[Z_{RWG}]$ can be calculated from the results of the previous Sections as follows,

$$Z_{RWG}(m, n) = \left[j\omega \left(\mathbf{A}_{mn}^+ \cdot \frac{\vec{\rho}_m^{c^+}}{2} + \mathbf{A}_{mn}^- \cdot \frac{\vec{\rho}_m^{c^-}}{2} \right) + \Phi_{mn}^- - \Phi_{mn}^+ \right] \quad (4.20)$$

and

$$V_{RWG}(m) = l_m \left(\mathbf{E}_m^+ \cdot \frac{\vec{\rho}_m^{c^+}}{2} + \mathbf{E}_m^- \cdot \frac{\vec{\rho}_m^{c^-}}{2} \right) \quad (4.21)$$

where

$$\mathbf{A}_{mn}^\pm = \frac{\mu}{4\pi} \int_S \vec{f}_n(\vec{r}') \frac{e^{-jkR_m^\pm}}{R_m^\pm} dS' \quad (4.22)$$

and

$$\Phi_{mn}^\pm = -\frac{1}{4\pi j\omega\epsilon} \int_S \nabla' \cdot \vec{f}_n(\vec{r}') \frac{e^{-jkR_m^\pm}}{R_m^\pm} dS' \quad (4.23)$$

with

$$R_m^\pm = |r_m^{\vec{c}^\pm} - r'_m| \quad (4.24)$$

and

³This symmetrical matrix is a result of applying Galerkin's testing procedure to the corresponding integral equation.

⁴In this context, the term "weighted" refers to the equation on which the testing functions have been applied.

$$\mathbf{E}_m^\pm = \mathbf{E}^i(r_m^{\vec{c}^\pm}) \quad (4.25)$$

i.e. the incident field calculated at the centre of the observation triangles, T_m^\pm , respectively.

4.1.5 Numerical evaluation of the MoM matrix elements

Calculating all the impedance matrix entries, $Z_{RWG}(m, n)$, is by far the more costly task in the setup of the MoM matrix equation, as it is of $\mathcal{O}(N_{RWG}^2)$ if one considers an edge-pair approach. This cost can however be reduced significantly if one considers that certain common integrals arise when calculating the impedance matrix entries. By taking this observation into account, it will therefore be much more time-saving to consider face-pair combinations as opposed to edge-pair combinations.

Regardless of whether edge-pair or face-pair iterations are concerned when filling the impedance matrix, there will be a number of numerical integrations to be carried out over the triangular sub-domains. Numerical evaluation of these integrals can be accomplished by using numerical quadrature techniques specially developed for triangular sub-domains as will be discussed in Section (4.2.3). It is also to be noted that for the self-terms, i.e. when $p = q$, the integrands in the integrals are singular (this happens when the source and observation domains coincide causing $r_m^{\vec{c}^\pm} = r_m^{\vec{r}}$) and need to be handled explicitly in many cases. Typically this entails that the singular portion of the integrand needs to be extracted and integrated analytically [2].

In the following Section, a practical implementation for the underlying mathematical formulations discussed in this and the previous Sections will be presented. The implementation has been developed by the author and is entitled, *GMoM*.

4.2 A practical implementation of the RWG MoM formulation, viz. *GMoM*

The discussion of the *GMoM*-implementation will parallel the general MoM overview presented in Section 3.1 on page 20, i.e. Figure (3.2), which is somewhat altered to include the post-processing phase, as illustrated in the general *GMoM* approach presented in Figure (4.3).

Before considering the numerical techniques associated with each of the *GMoM* phases depicted in Figure (4.3), it is necessary to first introduce the programming approach followed for the underlying implementation.

4.2.1 Programming considerations

In [1], pages 267 to 268, a number of useful considerations are listed regarding the implementation of a CEM code. These factors are summarised below:

1. Start with *simple models* first and gradually extend the algorithm to incorporate a more general approach.
2. Use *existing packages* where possible. Frequently used solution techniques, such as LU-decomposition, are freely available in routines such as that offered by the LAPACK suite.

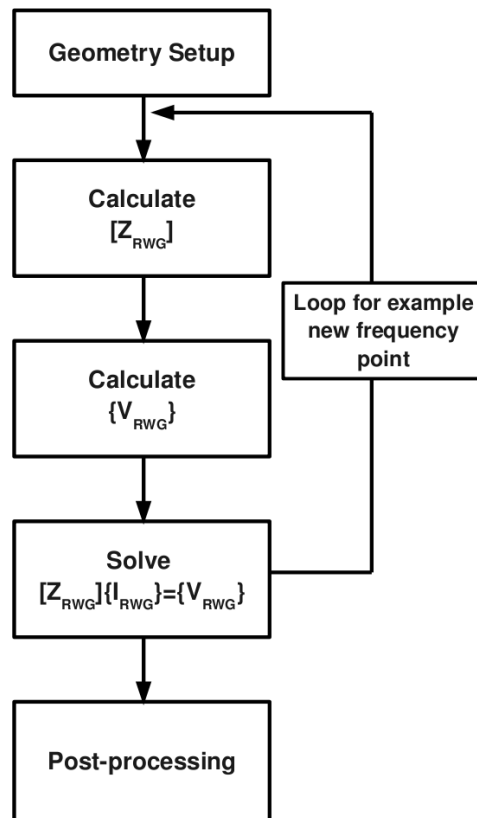


Figure 4.3: Schematic overview of the general GMoM algorithm, a practical implementation of the MoM that incorporates the use of RWG basis functions.

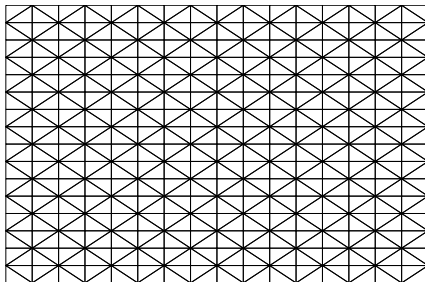
3. Use a *proper scientific programming environment*. It is important to consider programming environments that provide easy support for all the necessary tools required by the particular implementation. Typically, this includes support for complex numbers, parallel programming extensions and the post-processing of results. When considering the parallelisation of the implementation, it is also important to note that various commercially available software environments that require licenses can be costly when applied in a distributed computing environment, which may require additional or special versions of the license. In this case, it is the present author's viewpoint that one should rather consider open source codes such as Octave and Python.
4. Program with the goal of ensuring *modularity*. Various well-known programming languages, such as Python, Matlab, C++, etc. provide the user with an object-orientated design approach, that strongly supports the concept of modularity. If a sequential implementation is however considered, then modular-design entails that one should develop and test sections of code independently and not as a single complex aggregation of statements.
5. *Debug intelligently*. It is important to consider both programming errors and also errors in the underlying numerical implementations of the physics of the problem.
6. Code *validation* is very important. This can be accomplished by comparing the formulation with results that is obtained by other CEM software packages, and also (if possible) to measured data.

In addition to the above, an article that is specifically concerned with the practical implementation of the MoM, is that of Makarov [35]. This article is focussed on using Matlab for practical antenna simulations enabled by a package entitled the *partial differential equation* (PDE) toolbox. This package enables the user to simulate the radiation and scattering of simple metal antennas. Various practical implementation concepts, such as the modelling of an antenna feed with a thin strip dipole model and post-processing techniques for the visualisation of the current density in three dimensions, are discussed in this paper.

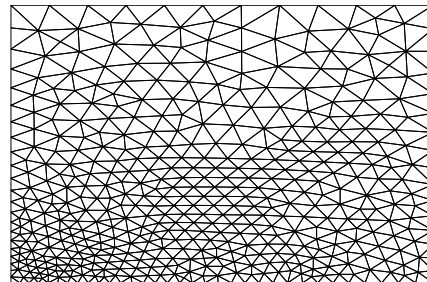
The author's approach incorporates the considerations listed above as expressed in [1] and also makes use of many concepts explained in [35] as will be illustrated in the following Subsections. In addition to the aforementioned, an important goal of the author's implementation was to incorporate a distributed programming model. By considering the speed-up results obtained by the MPI-bindings, mpi4py and MPITB, illustrated in Section 3.8.1, Figure 3.9 on page 29, for the freely distributed software packages, Octave and Python, a GMoM implementation was developed in both these programming environments. At the time of writing however, the graphical toolboxes for these packages were quite limited compared to a commercial package such as Matlab. Since the post-processing phase is mainly concerned with data-sets of $O(N_{RWG})$ a single license for the Matlab package was therefore obtained, to make use of the graphical processing capabilities offered by this package.

The key considerations undertaken in the development of the various phases of GMoM are discussed in the following Subsections.

4.2.2 Geometry setup



(a) Example of a rectangular plate discretized with the Delaunay triangulation obtained with Matlab



(b) Example of a rectangular plate discretized with the triangulation scheme presented by Gmsh

Figure 4.4: Example of triangulation schemes obtained by (a) Matlab and (b) Gmsh respectively

The first step in the RWG MoM formulation is the so called *Geometry Setup* and entails that the structure be discretized into a number of triangular sub-domains. It is common practice to choose triangular patches, as they offer flexibility to fit nearly any arbitrary geometry [2]. Standard Matlab supports the Delaunay triangulation⁵ of a grid of arbitrary points and can be used to obtain the triangular mesh. To illustrate the use of this triangulation scheme, the following built-in Matlab commands can be used to construct a triangular mesh on a rectangular plate of dimensions $plateWidth \times plateLength$ as illustrated in Figure 4.4(a),

⁵If a set of data-points is provided, then the Delaunay triangulation is a set of lines that connects each point to its natural neighbour.

```

1  [x,y] = meshgrid(1:delta:plateWidth , ...
2      1:delta:plateLength);
3  nodes = [x(:) y(:)];
4  triangles = delaunay(x,y);

```

Listing 4.1: Code fragment illustrating how a triangular mesh can be generated with Matlab

In Listing 4.1, the function `meshgrid` creates an array of data-points discretized in steps of δ , which ideally ranges between $\lambda/10$ and $\lambda/20$. This grid also defines the `nodes`-datastructure containing the coordinates of all the triangle vertices. The grid-data then serves as input to the `delaunay` function, which produces a $T \times 3$ matrix `triangles` where each row holds the vertex-information of each of the T -triangles.

Frequently however, it is necessary to vary the mesh size over different sections of the geometry. A finer mesh is required by sections of the geometry where a more accurate sampling of the current distribution is required, e.g. at the port of an antenna. On parts of the geometry where the current distribution is less significant, a courser mesh can be used. By varying the mesh in this manner, a significant saving in computational resources can be accomplished.

Varying the mesh size can be challenging if using the built-in Matlab functions, `meshgrid` and `delaunay`. In [35], the *PDE* toolbox is used to accomplish this. It is however to be noted that an additional cost is associated with this toolbox. The author's approach was to create a simple interface in Matlab for the use of another freely distributed software package called Gmsh [36]. Varying mesh sizes, can easily be accomplished as is illustrated in the following Gmsh code fragment for the discretization of a rectangular plate, the result of which is illustrated in Figure 4.4(b),

```

1 Point(1)={0,0,z,lc/2};
2 Point(2)={length,0,z,lc/2};
3 Point(3)={length,width,z,lc};
4 Point(4)={0,width,z,lc};
5 Line(1) = {1,2};
6 Line(2) = {2,3};
7 Line(3) = {3,4};
8 Line(4) = {4,1};
9 Line Loop(1)={1,2,3,4};
10 Plane Surface(1)={1};

```

Listing 4.2: Code fragment illustrating how a triangular mesh can be generated with Gmsh

In Listing 4.2, lc is the maximum edge-length that is associated with a given node, specified as a *Point* in the code fragment. The boundary of the plate is constructed by connecting a set of *Lines*, which is then used to specify the *Plane Surface*.

The Gmsh interface implemented by the author creates the above geometry specification which is then used as input for Gmsh. Gmsh is then executed from Matlab by means of the `unix` or `dos` commands, depending which operating system is used. The triangulation-data is then written to a mesh file that is parsed with Matlab from which the `nodes` and `triangles` datastructures are filled.

After the triangular mesh is obtained, the data can be used to locate the $N_{RWG}^{(i)}$ non-boundary edges of this domain. An RWG basis function is then allocated to each of these non-boundary edges. It is useful to store this information in a new data-structure, e.g. `sharedEdgesList`, where each entry contains information such as the positive and negative triangle-indices used to

define the polarity of the RWG basis function. It is to be noted that for the Python implementation, the above data-structures are declared as *classes* and for Matlab as `struct` datatypes.

4.2.3 Matrix equation setup

The mathematical expressions for the MoM matrix equation elements are listed in Section 4.1.4 on page 39, Equations (4.20) to (4.25). Numerically, the most challenging task is related to the integration of the free-space Green function (Eq. (4.5)) over the triangular subdomains. This is required when computing both the magnetic vector potential (Equations (4.20)) and the scalar potential (Equations (4.20)).

The approach followed by the author to compute these integrals, is based on finite point integration techniques using Gaussian quadrature, i.e. where the integral is calculated by evaluating the integrand at a number of points on the integration domain. A difficulty however arise when the source and observation points coincide, leading to the free-space Green function becoming singular, as stated in Section 4.1.5 on page 40. A typical approach to address this problem is to extract the singular integrand and to evaluate it by means of analytical techniques [2]. Although leading to a better accuracy for source and observation points in close proximity to each other, extensive preliminary mathematical work is required to implement this accurately, as stated in [35]. With the primary focus of the author's work directed towards reducing the time associated with solving the MoM matrix equation, this was therefore not pursued further. Instead, the integration points were selected to not coincide with the triangular midpoints, i.e. where the observation points are located. No separate formulas were therefore required to evaluate the integrals pertaining to self-terms of the impedance matrix.

The approach followed by the author for the numerical integration, namely Gaussian quadrature involves approximating an integral by a summation of integrand samples. Each of the sample points has a weighting coefficient associated with it, which when considering a triangular sub-domain may be expressed in the following general form [37],

$$\iint f(\lambda_1, \lambda_2, \lambda_3) dA = A \sum_{i=1}^N w_i f(\lambda_1, \lambda_2, \lambda_3) \quad (4.26)$$

where the function is written in terms of simplex coordinates, $(\lambda_1, \lambda_2, \lambda_3)$. The area of the triangle is expressed as A and N is the total number of sampling points on the integration domain. The weighting coefficient associated with the i th sampling point is denoted as w_i .

The simplex coordinates introduced into the above expressions are merely a transformation of the global coordinate system to a local coordinate system that is defined within the triangle, T_n which can be explained with the aid of Figure 4.5 as done in [2]. Firstly, it is important to note that the vectors $\vec{\rho}_i$ in Figure 4.5 divide the triangle into three sub-triangles of areas A_1 , A_2 and A_3 respectively. The areas of the three sub-triangle are related to the total area, A , as follows,

$$A = A_1 + A_2 + A_3 \quad (4.27)$$

The simplex coordinates⁶ can then be defined as,

$$\lambda_1 = \frac{A_1}{A}, \lambda_2 = \frac{A_2}{A}, \lambda_3 = \frac{A_3}{A} \quad (4.28)$$

which are inter-related due to the area constraint as follows,

⁶In [2], the term *normalised area coordinates* are used for this coordinate system.

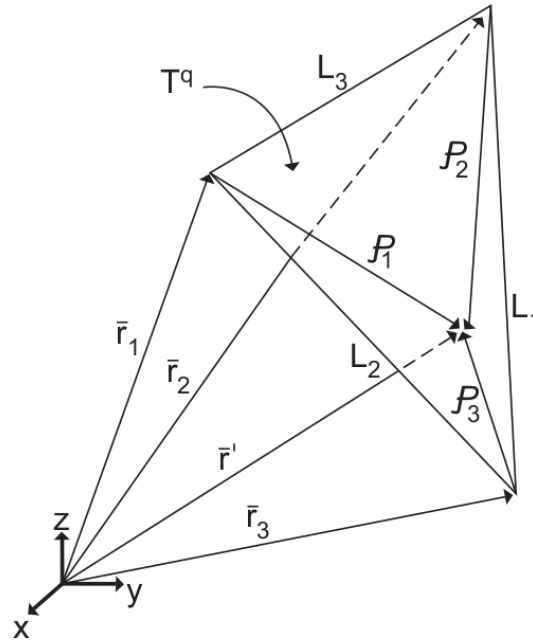


Figure 4.5: Simplex coordinates and edges illustrated on a triangular subdomain

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \quad (4.29)$$

All three coordinates vary between unity and zero according to the area of each sub-triangle, and $(\lambda_1, \lambda_2, \lambda_3)$ takes on the values $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$ at the triangle vertices, \vec{r}_1 , \vec{r}_2 and \vec{r}_3 respectively. An arbitrary point, \vec{r}' , in terms of global coordinates can then be expressed in terms of the simplex coordinates as follows,

$$\vec{r}' = \lambda_1 \vec{r}_1 + \lambda_2 \vec{r}_2 + \lambda_3 \vec{r}_3 \quad (4.30)$$

where the three triangle vertices are expressed in vector form as \vec{r}_1 , \vec{r}_2 and \vec{r}_3 respectively.

Many of the quadrature rules available, are formulated to exactly integrate polynomial functions of order $n \leq p$, which are said to be accurate to degree p . Therefore, as the number of sampling points are increased, the degree of accuracy usually increases. It is however important to remember that a number of integrations are required for each impedance matrix entry which directly influences the computational runtime of the matrix-filling phase. It is therefore important to consider the number of sampling points that is required to give a certain degree of accuracy [37].

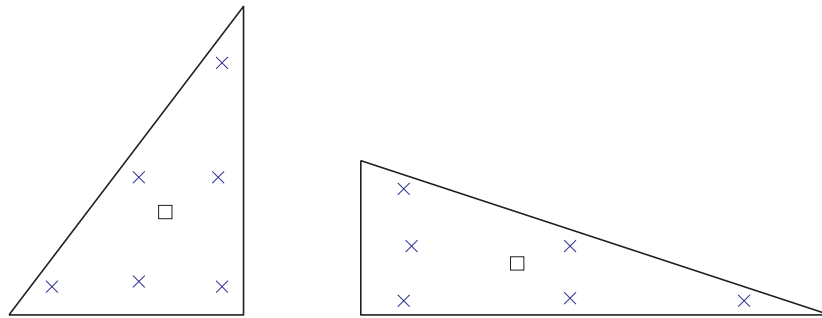
In general, deriving a quadrature rule of a given order, involves determining the number of sampling points, the location of each point and their corresponding weights. Additionally, symmetry constraints can be incorporated on the sampling-locations to eliminate variations that are caused by element vertex reordering. In [38], efficient symmetrical Gaussian quadrature rules for polynomials of order $1 \leq p \leq 20$ are provided. The simplex coordinates and corresponding weights calculated in [38], derived for three and six sampling points respectively, are summarised in Table 4.1.

The MoM formulation developed by the author, namely GMoM, incorporates the use of the six sampling points presented in Table 4.1, as this was obtained to result in a sufficient degree of accuracy while limiting the computational overhead related to the integration of the magnetic and scalar potential integrals to within a reasonable degree. Graphically, these integration points

Table 4.1: The number of sampling points, N , for a symmetric Gaussian quadrature rule of degree P

P	N	weight	λ_1	λ_2	λ_3
2	3	1/3	2/3	1/6	1/6
4	6	0.2233	0.10810	0.4459	0.4459
		0.1099	0.8168	0.0915	0.0915

are illustrated in Figure 4.6 for various rotations of a triangle, to illustrate the symmetrical properties of this technique. The centre-point of the triangle is included for reference.

**Figure 4.6:** Simplex coordinates illustrated on a triangular subdomain for various rotations. The centre point of the triangle is included for reference (depicted by the square).

The remainder of the terms present in the calculation of the MoM matrix equation primarily involves dot-products. When performing all calculations in the Cartesian coordinates, the dot-product between two arbitrary vectors $\vec{a} = a_1\hat{x} + a_2\hat{y} + a_3\hat{z}$ and $\vec{b} = b_1\hat{x} + b_2\hat{y} + b_3\hat{z}$ can be evaluated as follows,

$$\vec{a} \cdot \vec{b} = a_1b_1 + a_2b_2 + a_3b_3 \quad (4.31)$$

As noted in Section 4.2.1, 2), by making use of numerical packages provided by certain programming language, operations such as the dot-product defined in Eq. (4.31) can be evaluated without implementing the underlying mathematics. In Matlab/Octave, the `dot(a,b)` function provides this functionality. In Python, the equivalent function is implemented in a numerical package called *Numpy*, and can be accessed as `numpy.dot(a,b)`.

An overview of the solution-techniques for calculating the surface current distribution that is incorporated in GMoM will be discussed in the following Section.

4.2.4 Solving the MoM matrix equation

In Chapter 2 various approaches for solving the MoM matrix equation were reviewed. The approach followed by the author is focussed on direct solution techniques based on the CBFM rather than iterative methods such as the MLFMA. The reason for this is related to possible convergence issues in the iterative solution phase as was noted in Section 2.3 and also due to promising results presented by techniques such as the CBFM when specifically applied to FPA simulations [39].

The CBFM solution technique has been introduced in Section 2.4 and will be discussed in detail in the following Chapter and will therefore not be pursued in this Section. Instead, a crucial part of the CBFM (and also for a full-wave MoM solution), namely that of *LU-decomposition* which has been referred to throughout the previous Chapters, will be reviewed in this Section.

When considering a symbolic solution to the MoM matrix equation listed in Eq. (4.19) on page 8, a straight forward approach would be to multiply the left and right hand side by the inverse of the impedance matrix,

$$\{I_{RWG}\} = [Z_{RWG}]^{-1} \{V_{RWG}\} \quad (4.32)$$

However, as explained in [1], this approach is seldom followed due to a high cost in the calculation of the matrix inverse. Instead, the impedance matrix is factored into the product of a lower and upper triangular matrix as follows,

$$\{Z_{RWG}\} = [L][U] \quad (4.33)$$

where $[U]$ is the upper triangular matrix and $[L]$ the lower triangular matrix respectively. The MoM matrix equation can then be expressed in terms of these triangular matrices as,

$$[L][U]\{I_{RWG}\} = \{V_{RWG}\} \quad (4.34)$$

An auxiliary vector is then introduced by grouping the $[U]\{I_{RWG}\}$ matrix-vector product, i.e.

$$\{b\} = [U]\{I_{RWG}\} \quad (4.35)$$

The equation,

$$[L]\{b\} = \{V_{RWG}\} \quad (4.36)$$

is then solved for the vector $\{b\}$ by using *forward substitution*. Finally, the desired solution, i.e. the unknown expansion coefficient for the RWG basis functions, can be obtained from

$$\{b\} = [U]\{I_{RWG}\} \quad (4.37)$$

by means of *backward substitution*. This approach is inherently an extension of *Gaussian elimination* typically used for solving matrix equations.

The above LU-decomposition approach used to obtain a solution for the MoM matrix equation is implemented in both Matlab and Python, the latter of which requires the *linear algebra* (linalg) module of the scientific python (Scipy) package.

The following built-in Matlab commands can be used to solve a matrix equation of the form $ZI=V$:

```

1  [L,U] = lu(Z)
2  b = L\V
3  I = U\b

```

Listing 4.3: LU decomposition implemented in Matlab.

which can be replaced by a single Matlab command, $I=Z\V$

In Python, the LU-decomposition is implemented as follows by using the linalg module of the Scipy package,


```

1  (lu , piv ) = Scipy . linalg . lu_factor (Z)
2  I = Scipy . linalg . lu_solve ((lu , piv ),V)

```

Listing 4.4: LU decomposition implemented in Python with the Linear Algebra (linalg) module provided by the Scientific Python package (Scipy).

4.2.5 Post-processing

With the resulting expansion coefficients calculated as explained in the previous Subsection, it is convenient to visualise the results in three dimensions depicted according to a specified colour scale. In [35], the following Matlab code-fragment is provided for the surface current visualization.

```

1  normCurrent = Current ./max(Current)
2  C = repmat(normCurrent,3,1)
3  h=fill3(x,y,z,C)

```

Listing 4.5: Displaying the normalised surface current distribution in three dimensions according to a colour scale with Matlab.

In Listing 4.5, `Current` is an $1 \times M_{RWG}$ vector of current magnitudes calculated in the *centre* of each of the M triangles. The vectors \mathbf{x} , \mathbf{y} and \mathbf{z} are the triangle vertices in Cartesian coordinates. The function `fill3` then fills the triangular patches according to the colour specified in `C`, which is calculated according to the entries of the normalised current distribution. All the surface current plots presented in the numerical results were generated in this fashion. At the time of writing, the plotting capabilities of Python were limited compared to that offered by Matlab and was therefore not utilised.

With the surface current calculated, another important quantity specific to radiating structures in the context of the SKA and MeerKAT is that of far-field directivity patterns. To incorporate this capability into GMoM, the author followed the elegant *dipole model*, as explained in [40] and referenced in [35]. In this technique, the discretized conducting surface is treated as a set of short dipoles of constant current stretching between the centroids of the triangles adjacent to each of the non-boundary edges. In Figure 4.7, such as dipole is illustrated.

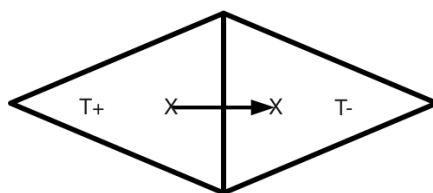


Figure 4.7: Illustration of a short-dipole modelled between the centroids of two triangles, T^+ and T^- , respectively

In the dipole model, each of the N_{RWG} elements therefore behave as an infinitesimal⁷ dipole of constant current. An important quantity that will surface in the remainder of this Subsection, is that of the *dipole moment* associated with the RWG element, which is the product of the

⁷This condition is ensured by conforming to the $\lambda/10$ to $\lambda/20$ meshing guideline, as discussed in Section 2.1.

dipole current and an effective dipole length. This quantity can be obtained by integrating the surface current, corresponding to the m th element, over the triangular subdomains on which it is defined, i.e. T_m^+ and T_m^- .

$$m = \int_{T_m^+ T_m^-} I_m \vec{f}_m(\vec{r}) dS = l_m I_m (\vec{r}_m^{x^-} - \vec{r}_m^{x^+}) \quad (4.38)$$

where the current associated with this dipole is merely the m th expansion coefficient, I_m , multiplied by the length of the m th non-boundary edge, l_m . The radiated magnetic and electric fields of a short dipole can then expressed as follows in terms of the dipole moment,

$$\begin{aligned} \vec{H}(\vec{r}_l) &= \frac{jk}{4\pi} (\vec{m} \times \vec{r}_l) C e^{-jkr_l}, \\ C &= \frac{1}{r_l^2} \left[1 + \frac{1}{jkr_l} \right], \\ \vec{E}(\vec{r}_l) &= \frac{k\eta}{4\pi} \left\{ (\vec{M} - \vec{m}) \left[\frac{jk}{r_l} + C \right] + 2\vec{M}C \right\} e^{-jkr_l}, \\ \vec{M} &= \frac{\vec{r}_l \cdot \vec{m}}{r_l^2} \vec{r}_l \end{aligned} \quad (4.39)$$

where η is the wave impedance of free-space, $\eta \simeq 120\pi$ and \vec{r}_l is calculated relative to the dipole centre. When considering an arbitrary point \vec{r} relative to the global coordinate origin, the vector \vec{r}_l can then be calculated as,

$$\vec{r}_l = \vec{r} - \frac{1}{2}(\vec{r}_m^{x^-} - \vec{r}_m^{x^+}) \quad (4.40)$$

where \vec{r}_m^{\pm} is the distance vector to the centroids of triangles T_m^{\pm} respectively. The above equations hold for both the near and far-fields of a radiator [35].

With an expression for the electric and magnetic fields at an arbitrary point \vec{r} defined in Equations (4.39) and (4.40) the task of calculating the far-field directivity pattern for the radiating structure is rather straight forward. Consider the general expression for the directivity, D , of a radiator in a given direction depicted in terms of spherical coordinates (\vec{r}, θ, ϕ) ,

$$\vec{D}(\vec{r}, \theta, \phi) = \frac{\vec{U}(\vec{r}, \theta, \phi)}{P_{rad}/4\pi} \quad (4.41)$$

where the radiation intensity, $U(\theta, \phi)$, i.e. the power radiated by the structure per unit solid angle can be expressed as follows,

$$U(\theta, \phi) = \frac{r^2}{2\eta} |\vec{E}(\vec{r}, \theta, \phi)|^2 \quad (4.42)$$

It is to be noted that radiation intensity is a *far-field parameter*. The distance vector \vec{r} should therefore be taken sufficiently far from the source where the radial component of the electric field, E_r , is assumed (if present) to be negligible. The most commonly used criterion for the minimum distance of far-field observations is $2K^2/\lambda$ as obtained from [41]. In this case, K is the largest dimension of the radiating structure⁸.

The term $P_{rad}/4\pi$ in Eq. (4.41) is the radiation intensity of the antenna averaged over all directions, i.e. a unit sphere, and may be calculated as follows,

⁸In the literature, the symbol K is typically replaced with D , which is not used here to avoid confusing it with the directivity of the radiator.

$$P_{rad} = \int_0^{2\pi} \int_0^\pi U \sin \theta \, d\theta d\phi \quad (4.43)$$

By considering Eq. (4.42) and (4.43), the directivity of the radiator can therefore be defined as the ratio of the radiation intensity in a given direction, (\vec{r}, θ, ϕ) , from the radiator, to the radiation intensity averaged over all direction [41].

The numerical integration of the radiation intensity in Eq. (4.43) is discussed in Section 2.7 of [41] and can be accomplished by firstly rewriting the expression for this quantity as,

$$U = \frac{r^2}{2\eta} |\vec{E}(\vec{r}, \theta, \phi)|^2 = B_0 F(\theta, \phi) \quad (4.44)$$

where B_0 is a constant.

A general expression for the radiated power, P_{rad} , can then be written in terms of Eq. (4.44) as,

$$P_{rad} = B_0 \int_0^{2\pi} \int_0^\pi F(\theta, \phi) \sin \theta \, d\theta d\phi \quad (4.45)$$

From integral calculus a series approximation for Eq. (4.45) can then be obtained,

$$P_{rad} \simeq B_0 \Delta\theta \Delta\phi \sum_{j=1}^M \sum_{i=1}^N F(\theta_i, \phi_j) \sin \theta_i \quad (4.46)$$

where the values of $\Delta\theta$ and $\Delta\phi$ may be taken as

$$\Delta\theta = \left(\frac{\pi}{N}\right) \text{ and } \Delta\phi = \left(\frac{2\pi}{M}\right) \quad (4.47)$$

when considering N and M uniform divisions over the π and 2π interval respectively.

The angular increments, θ_i and ϕ_j are selected in the centre of each of the divisions as follows,

$$\theta_i = \frac{\pi}{2N} + (i-1)\frac{\pi}{N}, \quad i = 1, 2, \dots, N \quad (4.48)$$

$$\phi_j = \frac{2\pi}{2M} + (j-1)\frac{2\pi}{M}, \quad j = 1, 2, \dots, M$$

and is illustrated graphically in Figure (4.8).

With the numerical scheme presented in this Subsection, the directivity pattern for an arbitrary radiator can be obtained. A typical and usefull method to view the results, is to depict these values on a polar-plot. In Matlab, this can be accomplished with the `polar` function.

4.3 Figure of merit used for evaluating *GMoM*

In the previous Subsections, an overview was presented of the numerical implementation of the RWG MoM formulation followed in GMoM by considering each of the key phases depicted in Figure (4.3) on page 41 respectively. Before presenting numerical results for scattering and radiating PEC structures simulated with GMoM, a figure of merit is defined by which the *accuracy* of this formulation can be compared to reference results.

Various quantities that will be presented in the remainder of the thesis, are of $O(N)$, and are contained in an algebraic vector of the form $\{x\}$. These quantities can range from the expansion

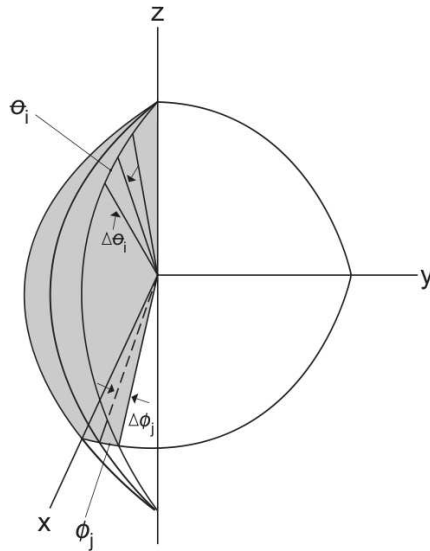


Figure 4.8: Digitization scheme of the pattern in spherical coordinates

coefficients obtained for the RWG basis functions, $\{I_{RWG}\}$, to the input reflection coefficient or s-parameters calculated over a frequency range.

The approach followed by the author, is to use the so-called 2-norm for a vector $\{x\}$ from which a normalised error percentage, $\epsilon_{\%}$ can be obtained as,

$$\epsilon_{\%} = \sqrt{\frac{\sum_{n=1}^N |x_n^{REF} - x_n|^2}{\sum_{n=1}^N |x_n^{REF}|^2}} \times 100\% = \frac{\|x_n^{REF} - x_n\|_2}{\|x_n^{REF}\|_2} \times 100\% \quad (4.49)$$

where $\epsilon_{\%}$ therefore expressing the relative error in $\{x\}$ with respect to the reference quantity $\{x^{REF}\}$ as a percentage.

4.4 Applying GMoM to a PEC scatterer

The current distribution on a $\lambda \times \lambda$ PEC plate illuminated by a normally incident plane wave was calculated with GMoM. The incident plane wave used as excitation is of the following form,

$$\mathbf{E}^i = E_x \hat{\mathbf{x}} \text{ with } E_x = 1\text{V/m} \quad (4.50)$$

The PEC plate is illustrated in Figure 4.9. Included in the illustration are the two principal cuts ($A - A'$) and ($B - B'$) on which the dominant component of current is extracted and compared to the results obtained from [2]. In addition, the results computed with FEKO along the same sections, are included and is illustrated in Figure 4.10. A $\lambda/16$ discretization was used for the triangular sub-domains. In Figure 4.11(a) and Figure 4.11(b) the normalised surface current density obtained by GMoM and FEKO are compared. For the FEKO results illustrated in Figure 4.11(b), the same scale is used as that for the GMoM results (Figure 4.11(a)).

According to the normalised error-percentage defined in Section 4.3, $\epsilon_{\%}$, GMoM compares to within 19.38% to the results obtained in the classic RWG article [2], and to within 21.56% when compared to FEKO for the ($A - A'$) cut. From Figure 4.9, the deviation can be observed at the top and bottom edge where the current distribution varies significantly. This can be attributed to the fact that the singularity term is not handled explicitly in the integration of

the Green function over the triangular sub-domain, as explained in Section 4.2.3. For the cut $(B - B')$, the value of $\epsilon_{\%}$ ranges between 2.14% and 2.17% when comparing GMoM to the results obtained in [2] and FEKO respectively.

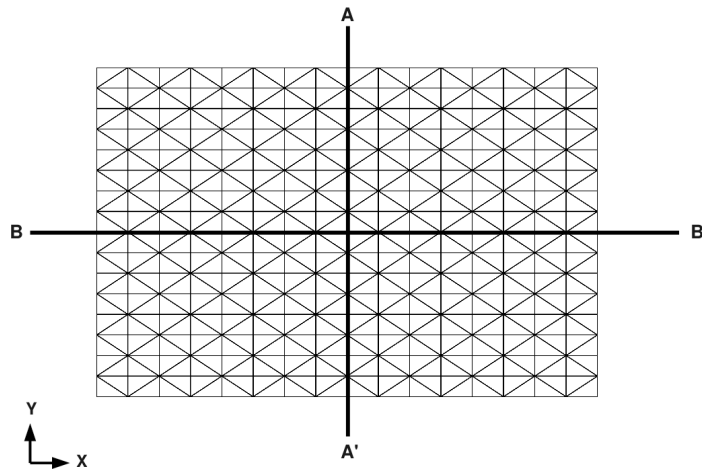


Figure 4.9: A square $\lambda \times \lambda$ PEC plate, discretized with triangular patches conforming to a mesh density of roughly $\lambda/16$. The two principal cuts are illustrated as $(A - A')$ and $(B - B')$ respectively.

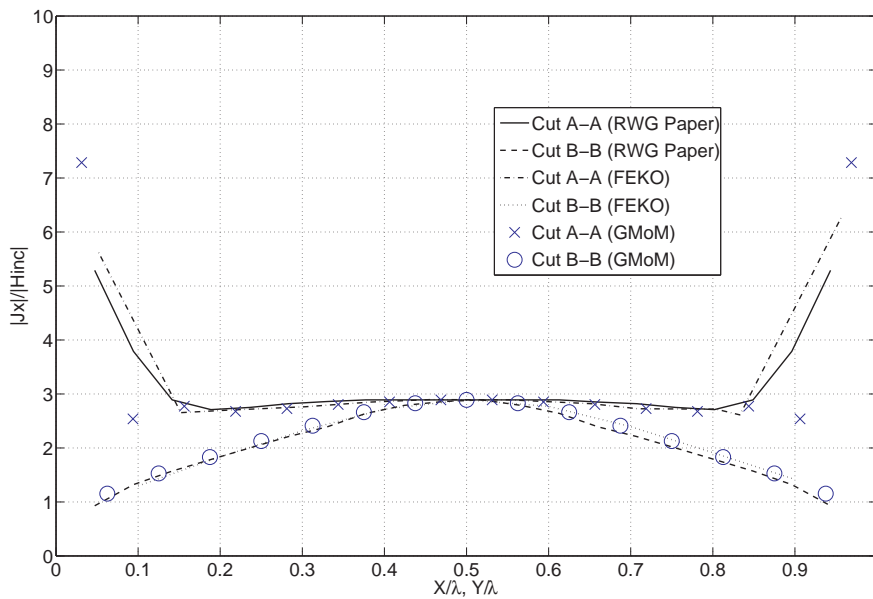


Figure 4.10: The distribution of the dominant current components on a square $\lambda \times \lambda$ PEC plate

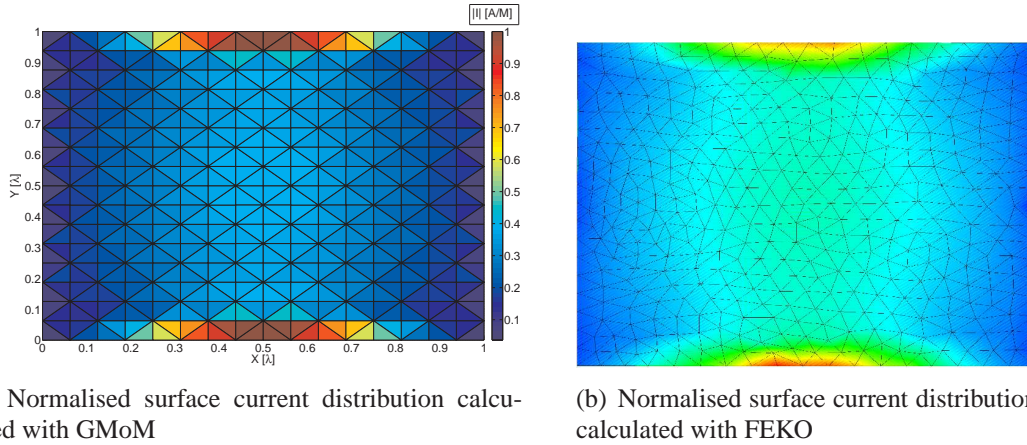


Figure 4.11: Surface current distribution calculated on a $\lambda \times \lambda$ PEC plate with (a) GMoM and (b) FEKO respectively.

4.5 Applying GMoM to a single Vivaldi radiator

The results presented in the previous Section are related to a scattering object. In this Section, the focus shifts towards radiating elements⁹, such as the Vivaldi antenna, which is typically encountered in a FPA arrangement [24] and will form the basis of the CBFM work that is to follow in the next Chapter. Before presenting the numerical results calculated with GMoM, the method followed to model the excitation of the antenna, is explained.

4.5.1 Modelling the antenna feed

To apply a voltage source to a radiator that is discretized with triangular patches, one approach that can be followed that is ideally suited to RWG type elements, is the *feeding-edge* model [35]. The derivation of this model, can be explained with the aid of Figure 4.12.

Consider the triangle pair, T_+ and T_- , that supports an RWG function, \vec{f}_m . The purpose of the feeding edge model, is to apply a voltage V_s across a gap Δ as illustrated in Figure 4.12 (b). When the edges are separated, in a mathematical sense, an $\hat{\mathbf{x}}$ directed electric field, \mathbf{E} , is formed as illustrated. According to *Laplace's equations*, the voltage in the source-free region between the edges may be expressed as,

$$\nabla^2 V(x) = \frac{\partial^2}{\partial x^2} V(x) = 0 \quad (4.51)$$

A general solution to this partial differential equation can then be expressed as,

$$V(x) = K_1 x + K_2 \quad (4.52)$$

The following boundary conditions that arise at the two edges depicted as l_m^+ and l_m^- respectively, can be used to obtain the unknown constant K_1 and K_2 ,

$$\begin{aligned} V(x = \Delta) &= 0 \\ V(x = 0) &= V_s \end{aligned} \quad (4.53)$$

⁹In general, when evaluating an antenna, the characteristics remain the same when considering the transmitting or receiving case.

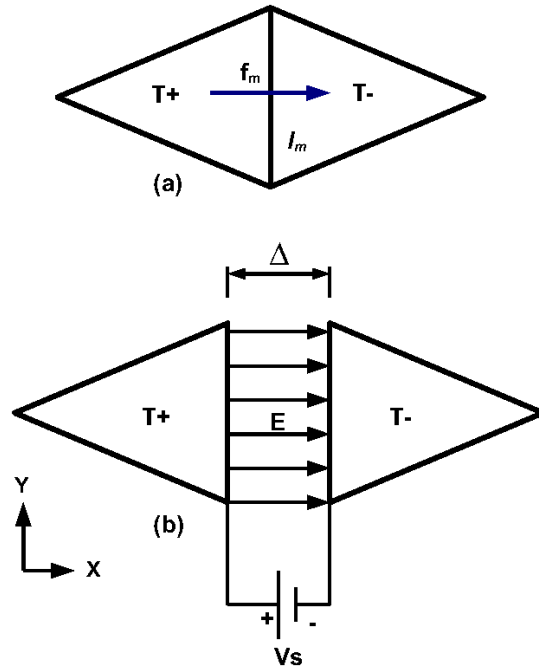


Figure 4.12: The feeding edge model associated with a driving edge, l_m

where it was specified that edge l_m^+ is located at $x = 0$ and edge l_m^- at $x = \Delta$ respectively. The voltage can then be written as,

$$V(x) = \frac{-V_s}{\Delta}x + V_s \quad (4.54)$$

The electric field, \mathbf{E} , can then be calculated as,

$$\mathbf{E} = -\nabla V(x) \hat{\mathbf{x}} = \frac{V_s}{\Delta} \hat{\mathbf{x}} \quad (4.55)$$

When the gap, Δ , tends to zero, the electric field calculated in Eq. (4.55) can then be expressed as,

$$\mathbf{E} = V(x)\delta(x) \hat{\mathbf{x}} \quad (4.56)$$

which is the delta-function approximation as given in Eq. (2) of [35]. The question now arises as to how one incorporates the feeding edge model into the excitation vector, $\{V_{RWG}\}$, of the MoM matrix equation. From [35] it follows that the excitation vector will have all zeros, except at the m th RWG element which takes on the form,

$$V_m = \int_{T^+ + T^-} \mathbf{E} \cdot \vec{f}_m dS = V_s \int_{T^+ + T^-} \delta(x) \cdot \vec{f}_m dS = l_m V_s \quad (4.57)$$

where V_s is the applied voltage depicted in Figure 4.12.

When using RWG basis functions, only the basis-function, \vec{f}_m , will contribute to the impedance calculation as only this function will have a component that is normal to edge m . The total normal current flowing over the m th edge, is equal to $l_m I_m$. The antenna impedance associated with the feeding edge model applied to the m th edge, can then be calculated as,

$$Z_A = \frac{V_s}{l_m I_m} \quad (4.58)$$

In Appendix A on page 79, the feeding edge approach is used to model a dipole antenna with a thin rectangular strip. The results that are obtained, are compared to FEKO and illustrate the efficiency and accuracy related to this technique. The numerical results obtained by GMoM for a Vivaldi radiator will be presented in the following Subsection.

4.5.2 Results

The discretized Vivaldi radiator is presented in Figure 4.13(a), with the details of the feeding edge model illustrated in Figure 4.13(b). The structure is analyzed at frequencies ranging between 1 GHz and 3 GHz, a frequency range of specific interest in the context of the SKA and the MeerKAT project. The principal dimensions, i.e. the aperture height and length of the Vivaldi antenna is 0.6λ and 1.5λ respectively, where λ is calculated at the lowest operating frequency, i.e. $f_l = 1$ GHz. The selected dimensions reflect typical values associated with single Vivaldi radiators [42].

To reduce simulation runtime, the mesh density is varied from $\lambda/20$ in the feed-region¹⁰ and along the tapered edge where the current is concentrated, to $\lambda/10$ at the outer edges where the current density is less significant. In this case, the wavelength, λ , is calculated according to the highest operating frequency, $f_h = 3$ GHz. The aforementioned discretization leads to a problem size of 2,172 RWG unknowns.

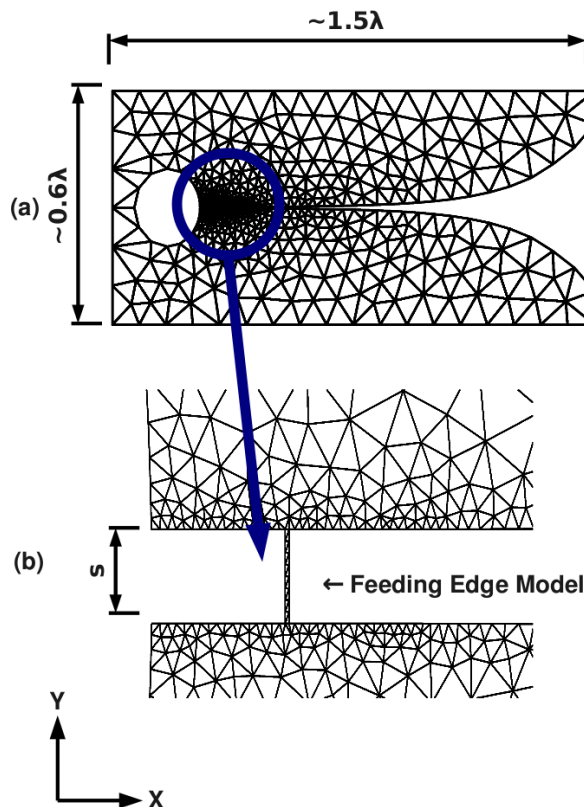
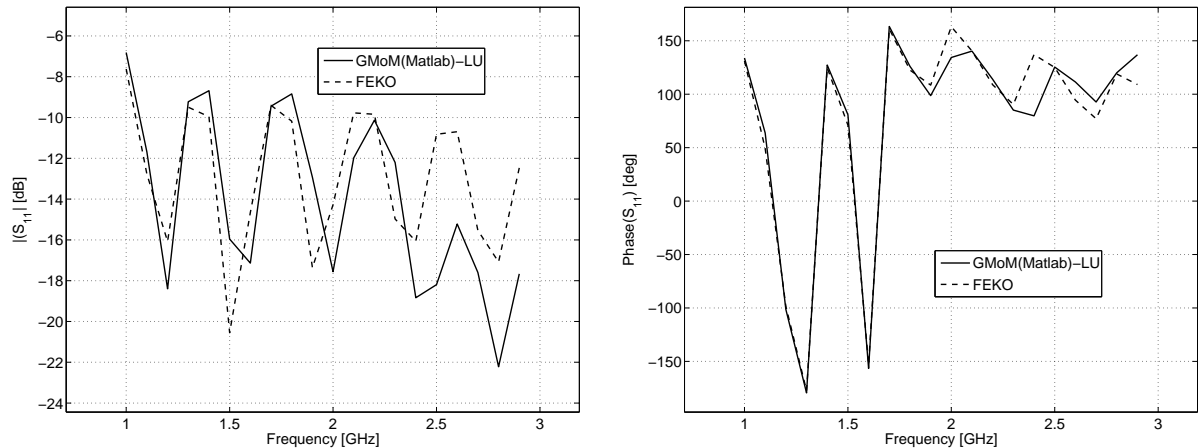


Figure 4.13: (a) A Vivaldi radiator, discretized with triangular patches of mesh-densities varying between $\lambda/20$ in the feed-region, to $\lambda/10$ at the outer edges of the structure. (b) The feeding edge location

In Figure 4.14(a) and 4.14(b), the magnitude and phase of the input reflection coefficient

¹⁰The strip-width is actually discretized according to its width, and is taken as $s/10$.

calculated with GMoM and FEKO respectively, are illustrated. The input reflection coefficient is calculated over a frequency range of 1 GHz to 3 GHz and corresponds to a characteristic impedance of 150Ω ¹¹. The E and H-plane directivity patterns for the Vivaldi radiator is illustrated in Figure 4.15(a) and 4.15(b). With reference to Figure 4.13 (a), the E-plane corresponds to the (x-y) plane and the H-plane to the (z-x) plane respectively. The normalised surface current distributions calculated with GMoM and FEKO are illustrated in Figure 4.16(a) and 4.16(b). The directivity patterns and surface current distributions are calculated at a centre frequency of 2 GHz. In Table 4.2, the parameters illustrated in Figures 4.14(a) to 4.15(b) calculated with GMoM and FEKO are compared in terms of the normalised error percentage, $\epsilon_{\%}$.



(a) The magnitude of the input reflection coefficient $|S_{11}|$ in dB for a Vivaldi antenna, calculated with GMoM and FEKO respectively

(b) The phase of the input reflection coefficient $\angle S_{11}$ in degrees for a Vivaldi antenna, calculated with GMoM and FEKO respectively

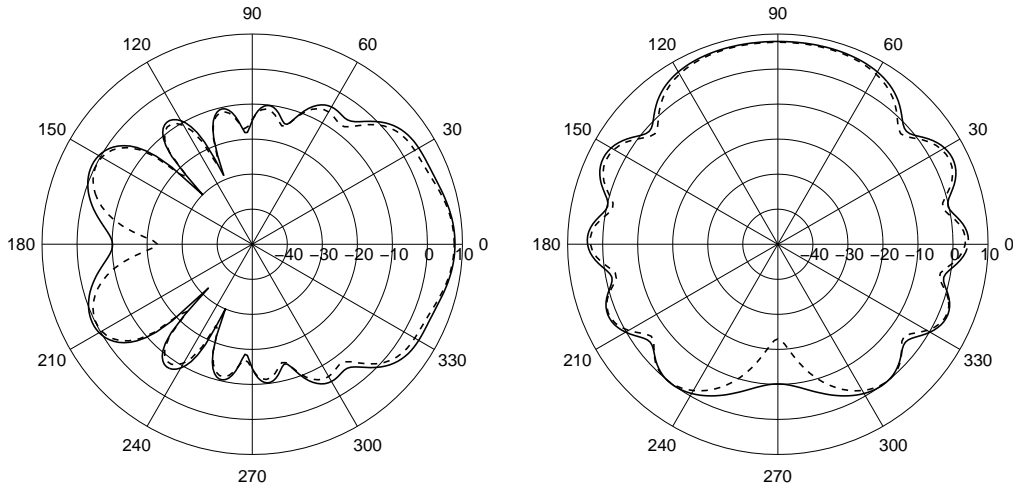
Figure 4.14: Comparing the input reflection for a Vivaldi antenna over a frequency range of 1 GHz to 3 GHz, calculated with GMoM and FEKO respectively

The results obtained by GMoM compare well to those calculated with FEKO. Deviations that can be observed are due to the approximations made in the calculation of the MoM impedance matrix, specifically with regard to the singular behaviour of the Green function when the source and observation points are in close proximity to each other. In addition to this, the excitation used for the FEKO model is that of a thin wire connected at the ports of the antenna, similar to that illustrated in Figure 4.13(b). For GMoM, a thin strip feeding-edge model is used, which again leads to a small sacrifice in the accuracy.

Table 4.2: The normalised error percentage obtained by GMoM when compared to FEKO for various quantities calculated for a Vivaldi antenna

Calculated quantity	$\epsilon_{\%}$
$ S_{11} $	22.68%
$\angle S_{11}$	14.12%
maximum E-plane directivity	0.4%
maximum H-plane directivity	0.4%

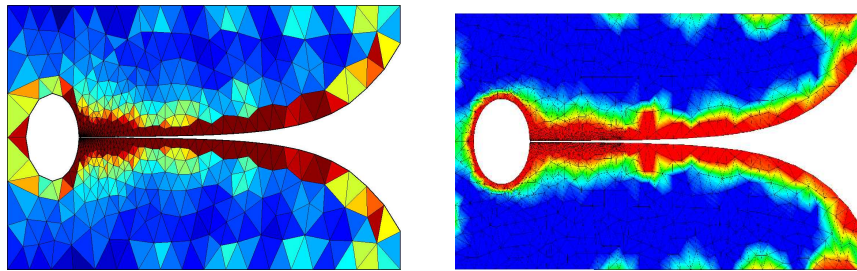
¹¹This value was obtained from research conducted by the author in [6].



(a) The E-plane directivity pattern in dB for a Vivaldi antenna, calculated with GMoM (-) and FEKO (- -) respectively

(b) The H-plane directivity pattern in dB for a Vivaldi antenna, calculated with GMoM (-) and FEKO (- -) respectively

Figure 4.15: Comparing the E and H-plane directivity patterns for a Vivaldi antenna at a frequency of 2 GHz, calculated with GMoM and FEKO respectively



(a) The normalised surface current distribution for a Vivaldi antenna, calculated with GMoM at 2 GHz.

(b) The normalised surface current distribution for a Vivaldi antenna, calculated with FEKO at 2 GHz.

Figure 4.16: Comparing the normalised surface current distribution for a Vivaldi antenna at a frequency of 2 GHz, calculated with (a) GMoM and (b) FEKO respectively.

4.6 Applying GMoM to an interconnected dipole antenna array

In this section, the focus shifts toward applying GMoM to an array of antenna elements. Before presenting numerical results, additional formulations that need to be implemented to characterise such a model are discussed in the following Subsection.

4.6.1 Modelling element interactions

When considering a general N port antenna configuration, important quantities that are required to evaluate the structure, are those of scattering parameters (S-parameters), which presents useful information regarding the interaction between various array elements. For a general N port antenna array, the S-parameters can be calculated as follows [43],

$$\begin{aligned}
 V_1^- &= S_{11}V_1^+ + S_{12}V_2^+ + \dots + S_{1N}V_N^+ \\
 V_2^- &= S_{21}V_1^+ + S_{22}V_2^+ + \dots + S_{2N}V_N^+ \\
 &\vdots \\
 V_N^- &= S_{N1}V_1^+ + S_{N2}V_2^+ + \dots + S_{NN}V_N^+
 \end{aligned}
 \tag{4.59}$$

from which the S-parameter, S_{ij} , can be extracted as,

$$S_{ij} = \frac{V_i^-}{V_j^+} \Big|_{V_k^+ = 0 \text{ for } k \neq j}
 \tag{4.60}$$

In practical terms, Eq. (4.60) states that the S-parameter, S_{ij} , is found by driving port j of the structure with an incident voltage, V_j^+ , and measuring the induced¹² voltage, V_i^- , at port i . The incident voltages on all the ports, except for the j th port are set to zero, implying that the remaining ports be terminated in matched loads to avoid reflections at the source end. When considering the case when $i = j$, Eq. (4.60) can be simplified as follows by using the expression for the input reflection coefficient of a radiator,

$$S_{ii} = \frac{Z_A - Z_0}{Z_A + Z_0} \Big|_{V_k^+ = 0 \text{ for } k \neq i}
 \tag{4.61}$$

where Z_A is the input impedance of the antenna and Z_0 the characteristic impedance used as the reference. As Z_A approaches Z_0 , the value of S_{ii} approaches zero, indicating that all the power that is applied at the input port is radiated by the antenna.

To provide GMoM with the ability to calculate S-parameters for a general N -port network, the feeding edge model presented in Section 4.5.1 on page 53 needs to be extended to include a source impedance at the input port, as illustrated in Figure 4.17.

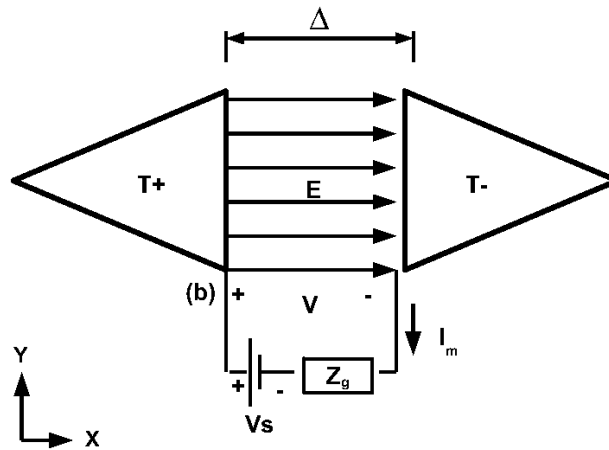


Figure 4.17: An extension of the feeding-edge model between two triangles, that includes a source impedance, Z_g .

By following the rationale underlined in Section 4.5.1, the applied voltage at the edge now changes to,

$$V = V_s - (I_m l_m) Z_g
 \tag{4.62}$$

¹²In some contexts, this value is also referred to as the *reflected* voltage.

where $I_m l_m$ is the total current flowing past the n th edge. By using the applied voltage, V , the excitation vector entry, V_m , changes as

$$V_m = l_m \{V_s - Zg(I_m l_m)\} \quad (4.63)$$

which is similar to the result obtained in Eq. (4.57), where the source impedance is omitted.

The question now arises as to how the source voltage and impedance influence the MoM matrix equation. The approach followed to address this, is an extension of what is done in Section 7.9 of [44] for wire structures and can be explained as follows:

The m th entry of the excitation vector can be expressed as,

$$\begin{aligned} \sum_{n=1}^{N_{RWG}} Z_{mn} I_n &= V_m \\ &= l_m \{V_s - Zg(I_m l_m)\} \end{aligned} \quad (4.64)$$

which can be rewritten as,

$$\sum_{n=1}^{N_{RWG}} Z'_{mn} I_n = V_m \quad (4.65)$$

where the impedance matrix entry, Z'_{mn} is calculated as follows,

$$Z'_{mn} = \begin{cases} Z_{mm} + Z_g l_m^2 & \text{when } m = n \\ Z_{mn} & \text{when } m \neq n \end{cases} \quad (4.66)$$

which illustrates that except for the diagonal entries, the new impedance matrix is the same as the original. The effect of applying a source with a series impedance, can therefore be accounted for by simply adding the source impedance to the corresponding diagonal impedance matrix entry. The excitation vector can then be calculated according to Eq. (4.57).

When following the methodology discussed in the above to calculate the S-parameters of an N port network, the matched loads can be added by zeroing the source, i.e. $V_s = 0$, where needed. Care should however be taken to restore the impedance matrix to its original form, i.e. remove the lumped load, before calculating the next S-parameter.

4.6.2 Results

To validate and verify the simulation of an array of antenna elements with GMoM, a 2×2 Dipole configuration was created in GMoM and FEKO respectively, as illustrated in Figure 4.18(a) and 4.18(b). The element numbering used for the S-parameter calculation is also illustrated in the diagram. The element spacing corresponds to $\lambda/2$, where λ is calculated at a centre frequency of 2 GHz. The operating frequency for this array again ranges between $f_l = 1$ GHz and $f_h = 3$ GHz. A discretization size of $\lambda/15$ was used for the GMoM array, which results in 108 RWG unknowns. For the dipole array created with FEKO, a wire model is used, with voltages applied as illustrated.

In Figure 4.19(a) and 4.19(b), the magnitudes and phases of the S-parameters, S_{1j} with $j = 1, 2, 4$, are illustrated respectively. The reason for evaluating only these three S-parameters and not the full sixteen element matrix, is due to the geometrical symmetry of the model. The formulation used in the previous Section, i.e. Eq. (4.66), was used for terminating the ports in matched loads of 50Ω . The directivity patterns calculated in the (x-y) and (z-x) plane respectively are illustrated in Figure 4.20(a) and 4.20(b). The results include that obtained by GMoM

and FEKO respectively. In Table 4.3, the results illustrated in Figures 4.19(a) to 4.20(b) are compared in terms of the error norm percentage defined in Section 4.3, i.e. $\epsilon_{\%}$.

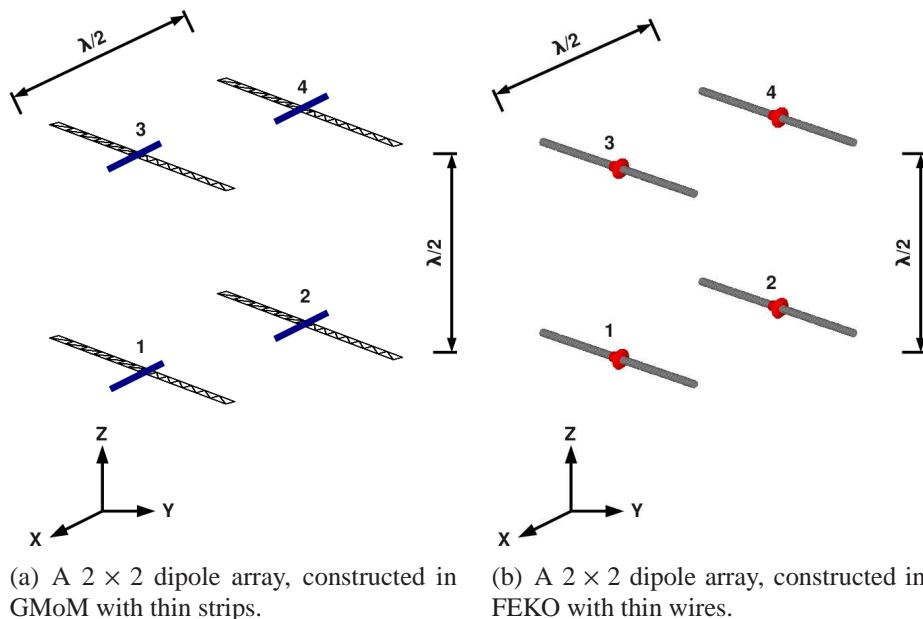


Figure 4.18: Comparing the geometry of a 2×2 dipole array configuration created in FEKO and GMoM respectively.

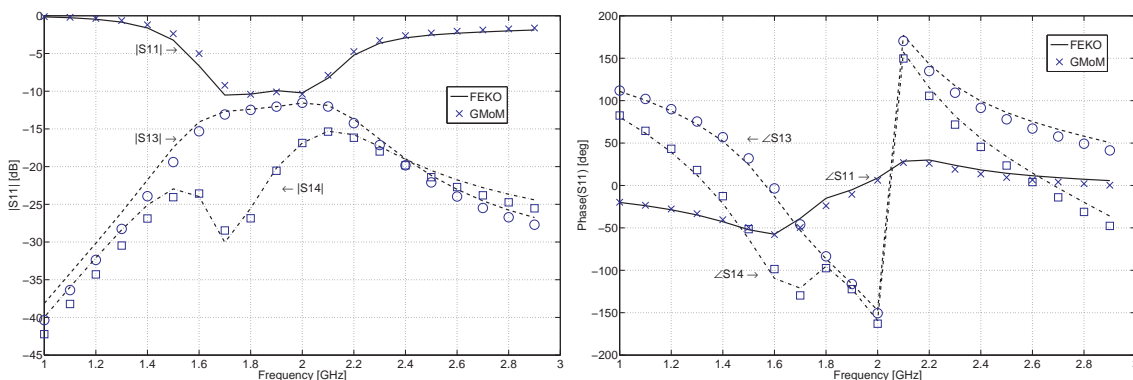


Figure 4.19: Comparing the S-parameters for a 2×2 dipole array configuration simulated with FEKO and GMoM respectively

From the results obtained in this Section, the accuracy that is obtained by GMoM compares sufficiently with FEKO.

4.7 Conclusions

In this Chapter, the underlying theory of the MoM when applied to PEC structures was presented in terms of the EFIE formulation. The numerical considerations and implementations

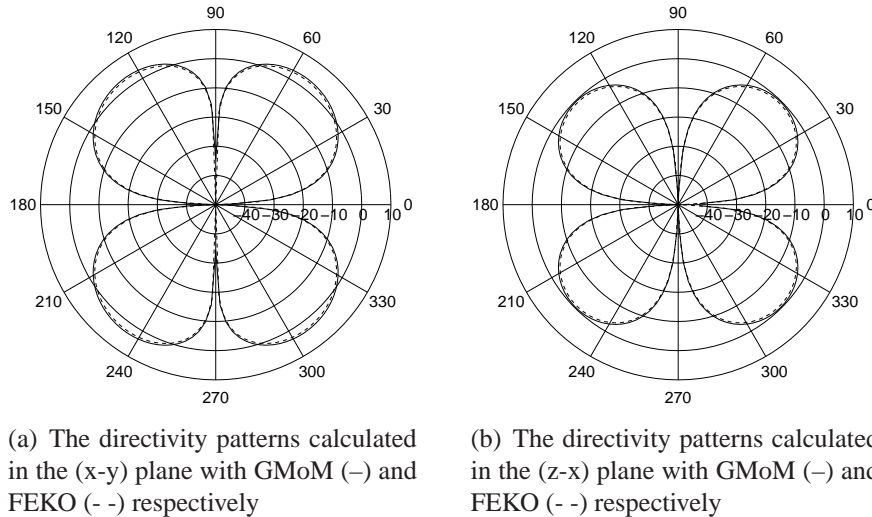


Figure 4.20: Comparing the directivity patterns for a 2×2 dipole array configuration simulated with FEKO and GMoM respectively. All the ports are excited with $V_s = 1$ V

Table 4.3: The normalised error percentage obtained by GMoM when compared to FEKO for various parameters calculated for a 2×2 dipole antenna array

Calculated quantity	$\epsilon_{\%}$
S_{11}	8.34%
$ S_{12} $	12.83%
$ S_{14} $	16.53%
maximum (x-y) plane directivity	23.34%
maximum (z-x) plane directivity	22.69%

of the MoM kernel developed by the author, namely GMoM, was discussed. In addition to the underlying integral formulations used in GMoM and their numerical analysis, the post-processing implementation was also discussed in terms of surface current visualisation and that of directivity patterns. The second part of the Chapter was focussed on extending GMoM, where needed, to incorporate a feeding-edge model with which one is able to add a voltage and source impedance to any arbitrary port defined by two non-boundary edges. The numerical results obtained by GMoM were compared to that of FEKO and also results available from [2] and verified that a sufficient accuracy can be obtained for radiating structures such as the Vivaldi antenna.

In the following Chapter, GMoM will be extended to incorporate the CBFM solver and analyse the efficiency thereof.

Chapter 5

The CBFM approach for solving the MoM matrix equation

In Section 2.4 on page 11, the reader was introduced to the CBFM approach from a scattering problem point of view. There, it was illustrated that for a scatterer consisting of discrete PEC plates, the simulation time and memory usage are significantly reduced by the CBFM formulation when compared to direct-techniques. The purpose of this Chapter is to illustrate how the CBFM can be applied to arrays of electrically interconnected periodic structures, such as the Vivaldi antenna array¹. The first section of this Chapter will focus on the general theory of the CBFM when applied to interconnected array configurations that is largely based on the research conducted in [39] and [45]. Numerical results will then be presented for a 3×1 and 7×1 linearly polarised array, to illustrate the accuracy of the CBFM. In the last part of the Chapter, the author will illustrate how domain decomposition can be applied to a general CBFM formulation as a means of further reducing the computational runtime associated with the method. The author's MoM formulation, GMoM, will be used for the CBFM implementation and also for results pertaining to a direct solver.

5.1 Applying the CBFM to an interconnected FPA

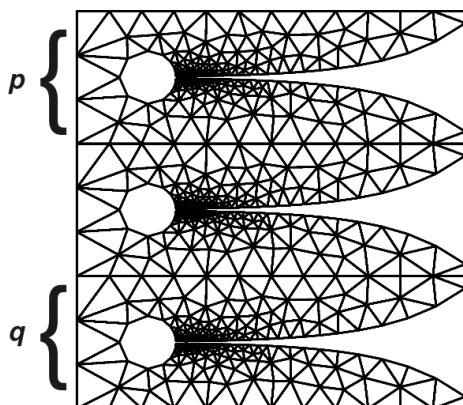


Figure 5.1: A 3×1 Vivaldi array depicted in terms of CBF subdomains p and q respectively.

¹In FPA configurations there is typically a high degree of periodicity, see for e.g. the *checker-board array* that is being developed by the ASKAP team in Australia [4].

5.1.1 Overview of the CBFM approach

To review the CBFM concept, specifically with interconnected radiating elements in mind, consider the 3×1 Vivaldi array depicted in Figure 5.1. The CBF sub-domains consists of a single array element, illustrated as p and q respectively.

Following the underlying methodology explained in Chapter 2, Section 2.4 on page 15, a set of CBFs can be generated for each of the sub-domains which leads to the formulation of a reduced matrix equation of the following form,

$$[Z_{CBFM}] \{I_{CBFM}\} = \{V_{CBFM}\} \quad (5.1)$$

If we apply a more versatile approach where the number of CBFs on the i th sub-domain is $K^{(i)}$, Eq. (5.1) represents a matrix equation of size $MK \times MK$, where $K = \sum_{m=1}^M K^{(m)}$. The reason for varying the number of CBFs on the sub-domains, is related to the placement of the antenna elements in the array, as will be discussed in the following Subsections. Furthermore, the quantity M represents the total number of CBFM sub-domains, which in the case of Figure 5.1, is equal to the number of antenna elements in the array, i.e. $M = 3$.

The reduced impedance matrix entries of Eq. (5.1) can be calculated as,

$$[Z_{CBFM}^{(p,q)}] = \langle [J_{CBFM}^{(p)}]_{LOC}^T, [Z_{RWG}^{(p,q)}] [J_{CBFM}^{(q)}]_{LOC} \rangle \quad (5.2)$$

where, $[J_{CBFM}^{(i)}]_{LOC}$ represents the *locally mapped* column-augmented CBFM expansion coefficients related to the i th sub-domain. This $N_{RWG}^{(i)} \times K^{(i)}$ matrix contains both the "primary" and "secondary" CBFM expansion coefficients for this domain.

Similarly, the CBFM excitation vector, $\{V_{CBFM}\}$, can be obtained as follows,

$$\{V_{CBFM}^{(p)}\} = \langle [J_{CBFM}^{(p)}]_{LOC}^T, \{V_{RWG}^{(p)}\} \rangle \quad (5.3)$$

where $\{V_{RWG}^{(p)}\}$, contains the entries of the primary excitation vector, $\{V_{RWG}\}$, of Eq. (2.2), corresponding to the RWGs on subdomain p .

By solving Eq. (5.1) for the unknown α coefficients contained in the CBFM solution vector, $\{I_{CBFM}\}$, the solution to the entire problem can then be calculated as,

$$\begin{aligned} \{I_{RWG}\}_{N_{RWG} \times 1} &= \sum_{m=1}^{K^{(1)}} \alpha_m^{(1)} \{J_{CBFM,m}^{(1)}\} + \sum_{m=1}^{K^{(2)}} \alpha_m^{(2)} \{J_{CBFM,m}^{(2)}\} \\ &+ \dots + \sum_{m=1}^{K^{(M)}} \alpha_m^{(M)} \{J_{CBFM,m}^{(M)}\} \end{aligned} \quad (5.4)$$

where each of the CBFM expansion coefficient vectors, $\{J_{CBFM,m}^{(i)}\}$, are mapped to their corresponding *global* indices in terms of the RWG formulation for the entire problem.

In this Chapter, the task of generating the "primary" and "secondary" CBFs for each sub-domain is restructured to account for the interconnectivity between the domains while also accounting for the placement of the elements in the array². A summary of the general CBFM formulation is illustrated in Figure 5.2. Each of the steps outlined in this diagram will be discussed in the following Subsections. The array configuration that forms the basis of the

²The *placement* in this regard refers to elements at or near the edge of the array and also centrally located elements.

discussion is the linearly polarised Vivaldi array depicted in Figure 5.1. It is however to be noted that the underlying theory remains the same when applied to any other antenna array structure. The concepts can also be extended and applied to dual-polarised arrays.

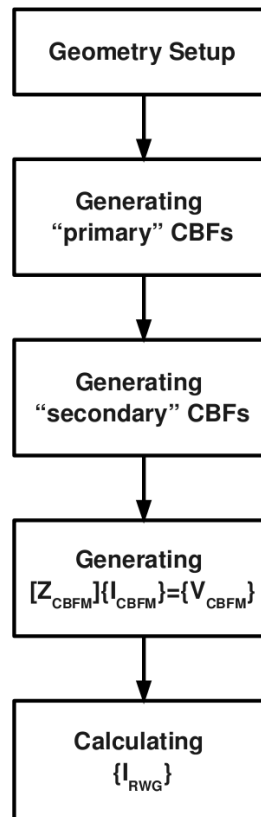


Figure 5.2: An overview of the general CBFM approach

5.1.2 Geometry setup

In order to rapidly construct the geometry of the array configuration, only one element is meshed, subsequently copied and then translated to its position in the actual array. This concept is illustrated in Figure 5.3 for a 7×1 Vivaldi array, where element 1 is meshed and translated to positions 2 to 7. Each of the antenna elements correspond to one primary CBFM sub-domain, that will each host a set of CBFs. The partitioning of the low-level RWG basis-functions is therefore kept identical for each array element and the polarity of the RWGs are chosen consistently at the inter-connections between the antennas.

5.1.3 Generating "primary" CBFs

To generate "primary" CBFs on each of the sub-domains illustrated in Figure 5.3 as 1 to 7, a set of sub-arrays are defined and extracted. For the array configuration illustrated in Figure 5.3, three types of sub-arrays are identified, as illustrated in Figure 5.4. Each of the sub-arrays will be used to construct a basis function for the antenna elements that has (a) one adjacent element, namely the two corner elements (1 and 7) and (b) those with two adjacent elements, i.e. the centre elements (2 to 6). The number of sub-arrays therefore corresponds to the number of

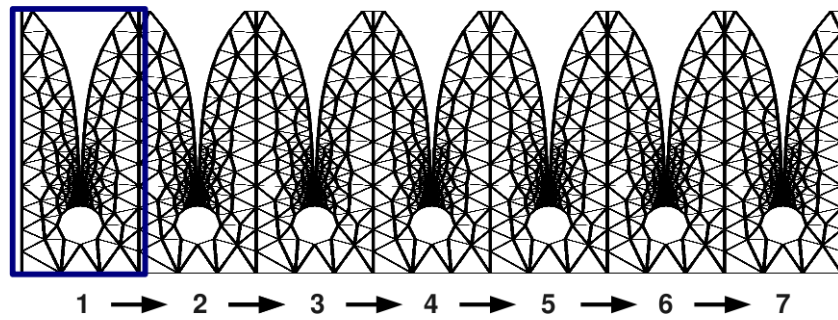


Figure 5.3: The discretization of a 7×1 Vivaldi array constructed by copying and translating one meshed element to its location in the array.

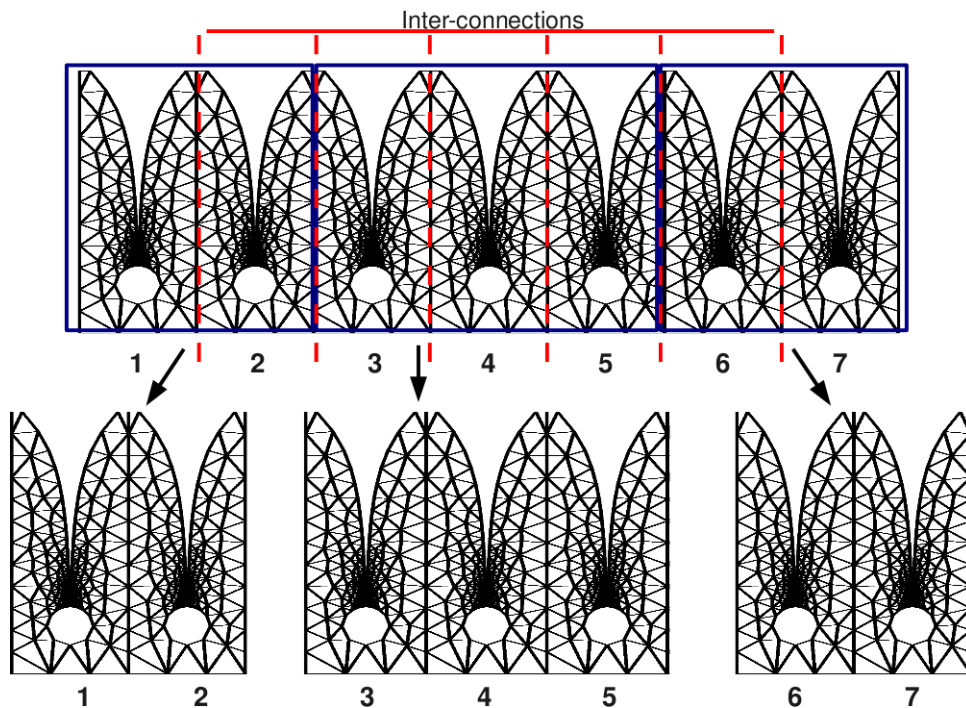


Figure 5.4: The extraction of sub-arrays from a 7×1 Vivaldi array for the construction of "primary" CBFs.

uniquely *extended* subdomains. The reason for extending the sub-domains is as follows: If a primary CBF is generated for a sub-domain consisting of a single element, current continuity may not be ensured at the interconnections between the elements (illustrated in Figure 5.4). Furthermore, by limiting the solution to that of a single element, inaccurate behaviour may be observed for the current near the fictitious edges of such a sub-domain³.

Primary CBFs are then constructed on each of the extended sub-domains by exciting each of the terminals in the sub-array in a sequential manner. The corresponding excitation vectors are then augmented column-wise to construct the "primary" excitation matrix, $[V_{sub}^{(i)}]$. The expansion coefficients of the "primary" CBFs, $[J_{CBFM,prim}^{(i)}]$, on the i th sub-array can then be calculated

³In [45], this is expressed as fictitious singularities that may arise at the interconnections.

as follows,

$$\left[Z_{RWG}^{(i,i)} \right] \left[J_{CBFM,prim,sub}^{(i)} \right] = \left[V_{sub}^{(i)} \right] \text{ for } i = 1, \dots, M' \quad (5.5)$$

where M' corresponds to the number of generating sub-arrays and $\left[Z_{RWG}^{(i,i)} \right]$ is the $N_{RWG,sub}^{(i)} \times N_{RWG,sub}^{(i)}$ sub-matrix extracted from the MoM matrix equation, Eq. (2.2) on page 8.

Following this manner, 2 "primary" CBFs are therefore constructed for the edge-located sub-arrays, and 3 for the sub-arrays representing the centre elements. Furthermore, the total number of RWG unknowns included in the i th sub-array, $N_{RWG,sub}^{(i)}$, is relatively small compared to that of the entire problem. The solution to Eq. (5.5) can therefore be obtained by means of LU-decomposition.

To account for the truncation effects at the outer edges of each sub-array, the following windowing function is applied to limit the support of each of the "primary" CBFs to that of a single element, depending on the location of the RWG elements in the sub-array.

$$[\Lambda(m, m)] = \begin{cases} 1 & \text{internal region} \\ 1/2 & \text{overlapping region} \\ 0 & \text{external region} \end{cases} \quad (5.6)$$

where $[\Lambda]$ is an $N_{RWG,sub}^{(i)} \times N_{RWG,sub}^{(i)}$ zero matrix, with the diagonal entries calculated as illustrated above. In Figure 5.5, the internal, external and overlapping regions are illustrated as (A), (B) and (C) respectively for each of the sub-arrays.

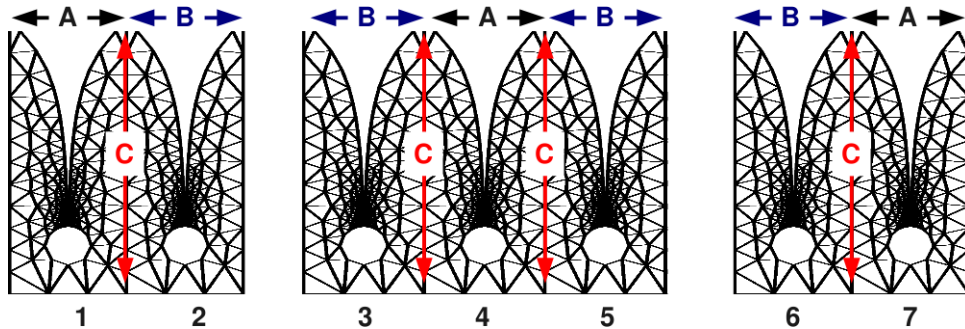


Figure 5.5: The windowing of the CBFM sub-arrays for a linearly polarised Vivaldi array.

The windowing function defined in Eq. (5.6) can then be multiplied with the "primary" CBFs on the i th sub-array to limit their support to the internal and overlapping region as follows,

$$\left[J_{CBFM,prim}^{(i)} \right] = \Lambda \left[J_{CBFM,prim,sub}^{(i)} \right] \quad (5.7)$$

where $\left[J_{CBFM,prim}^{(i)} \right]$ is an $N_{RWG,sub}^{(i)} \times K_{prim}^{(i)}$ matrix which can be converted to a size of $N_{RWG}^{(i)} \times K_{prim}^{(i)}$ by discarding the zero entries that are associated with the external region. In the aforementioned, $K_{prim}^{(i)}$ is the total number of "primary" CBFs on each sub-array.

The advantage of using the windowing technique explained in the previous paragraphs, is that the partially overlapping CBFs (after the mapping), sum with a proper weight in the overlapping region. The reason for this being that the corresponding partially overlapping windowing functions sum to unity across the entire problem domain.

The final step in the "primary" CBF generation, is to map the CBFs to the corresponding array element. Graphically, this is illustrated in Figure 5.6. The number of CBFs per sub-domain⁴ is also illustrated.

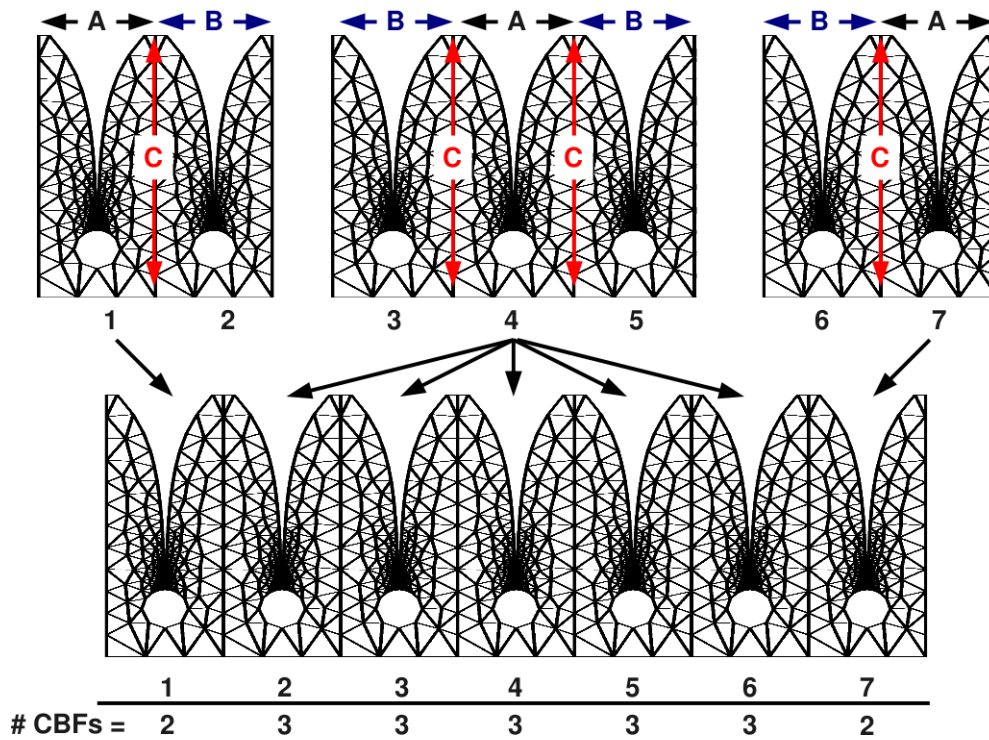


Figure 5.6: Mapping the windowed "primary" CBFs to the corresponding element positions in a linearly polarised Vivaldi array.

5.1.4 Generating "secondary" CBFs

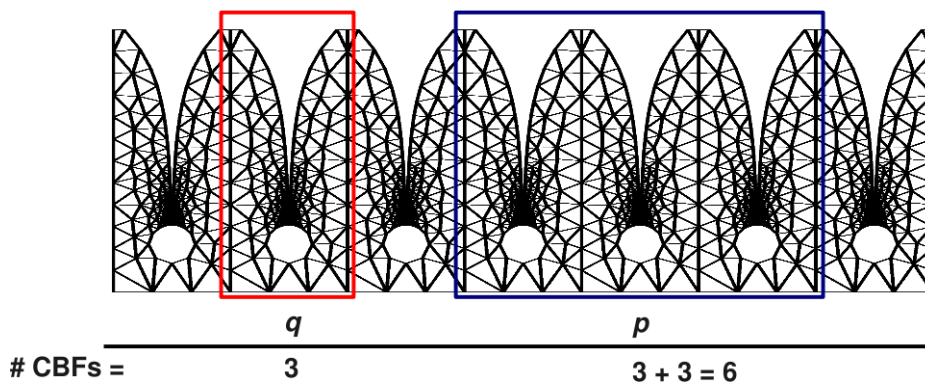


Figure 5.7: Generating the "secondary" CBFs for sub-array q by considering the primary CBFs on domain p for a linearly polarised Vivaldi array.

⁴In this context, the sub-domain refers to that defined by a single array element

When the interconnected array configuration consists of many elements, mutual coupling can play a non-negligible role. To account for this, the number of CBFs per sub-domain can be enlarged to achieve a more accurate representation of the final surface current. If considering only "primary" coupling effects, as explained in Section 2.4 on page 14, the CBFs that are generated following the methodology explained in the previous Subsection, may be used as distant current sources to the sub-arrays depicted in Figure 5.4.

To illustrate this concept, consider the array configuration in Figure 5.7, where the "secondary" CBFs for sub-array⁵ p are induced by the "primary" CBFs of sub-domain q . The increase in the number of CBFs defined on the p th sub-domain are also illustrated.

Following Eq. (2.13) on page 14, the secondary CBFs on the q th sub-array can be calculated as

$$\left[Z_{RWG}^{(p,p)} \right] \left[J_{CBFM,sec,sub}^{(p)} \right] = - \left[Z_{RWG}^{(p,q)} \right] \left[J_{CBFM,prim,sub}^{(q)} \right] \quad (5.8)$$

The matrix, $\left[Z_{RWG}^{(p,q)} \right]$, represents the coupling matrix between the q th sub-domain and p th sub-array respectively.

After accounting for the different "primary" source CBFs within a specified radius⁶ to the corresponding sub-array, the support of the resulting secondary CBFs, $\left[J_{CBFM,sec,sub}^{(p)} \right]$, are truncated with the windowing function defined in Equation (5.6) and appended to the already existing set of "primary" CBFs.

5.1.5 Generating and solving the reduced matrix equation

After mapping the truncated CBFs to their corresponding array elements, as depicted in Figure 5.6 on page 67, the reduced matrix equation (Eq. (5.1)) can be constructed as explained in Section 5.1. This matrix equation contains a relatively small number of unknowns when compared to the MoM matrix equation of the entire problem and can be solved directly without resorting to iterative techniques.

In the following Subsections, the formulations applied in Section 5.1 will be used to model a 3×1 and a 7×1 linearly polarised Vivaldi array respectively. The author's MoM formulation, GMoM, was used to generate the results.

5.2 Numerical Results

5.2.1 A linearly polarised 3×1 Vivaldi array

The surface current distribution for the three element linearly polarised Vivaldi array (Figure 5.1) was obtained with the direct and CBFM solution techniques implemented in GMoM respectively. The results are illustrated in Figures 5.8(a) and 5.8(b). The results are obtained at $f = 2$ GHz with all the ports of the array excited. The E and H plane directivity patterns were also calculated at this frequency for the active array and are illustrated in Figures 5.9(a) and 5.9(b). The array model is discretized with 3,474 RWG unknowns. For the CBFM, 7 primary CBFs were used as basis functions.

The relative error norm percentages, $\epsilon_{\%}$, are summarised in Table 5.1 for the calculated quantities. The vector, $\{I\}$, in Table 5.1 is the expansion coefficients of the RWG basis-functions of the entire problem.

⁵By identifying unique placements of the sub-arrays, the calculation of "secondary" CBFs can be limited to a small set of sub-arrays that needs to be taken into consideration.

⁶In [39] this radius is specified as equal to the width of two array elements.

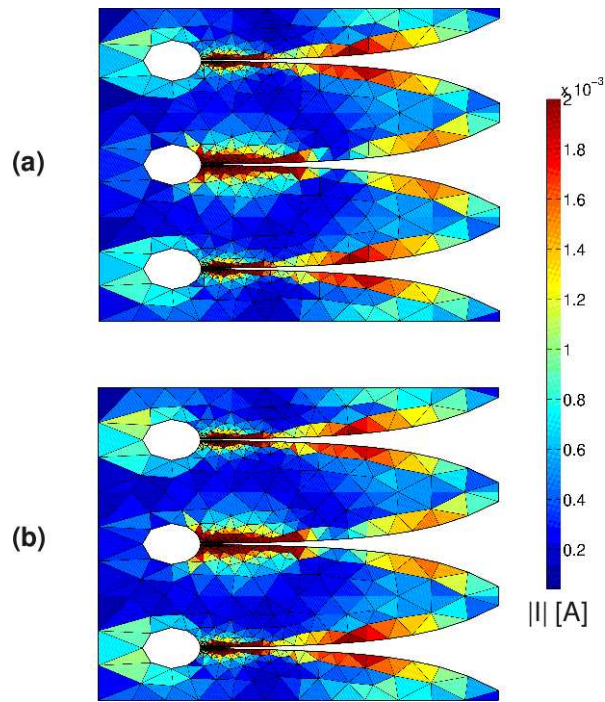
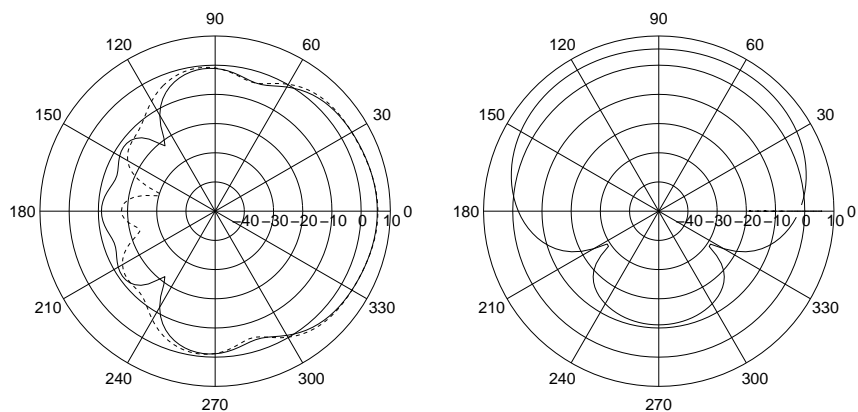


Figure 5.8: The surface current distribution on a 3×1 Vivaldi array calculated with (a) a direct solution technique and (b) the CBFM approach



(a) The E-plane directivity pattern in dB for a 3×1 Vivaldi antenna, calculated with a direct solver (- -) and the CBFM approach (-) respectively

(b) The H-plane directivity pattern in dB for a 3×1 Vivaldi antenna, calculated with with a direct solver (- -) and the CBFM approach (-) respectively

Figure 5.9: Comparing the E and H-plane directivity patterns for a 3×1 Vivaldi antenna at a frequency of 2 GHz, calculated with the CBFM and a direct solver.

From the results it is evident that the CBFM approach corresponds well to that associated with a direct solver. The formulation is also free of current singularities that may arise at the intersection between the antenna elements, which can be attributed to the windowing technique introduced in Section 5.1.3.

Table 5.1: The normalised error percentage obtained by GMoM when calculating various quantities for a 3×1 Vivaldi antenna with the CBFM and direct solution techniques.

Calculated quantity	$\epsilon_{\%}$
$\{I\}$	3.41%
maximum E-plane directivity	3.49%
maximum H-plane directivity	0.72%

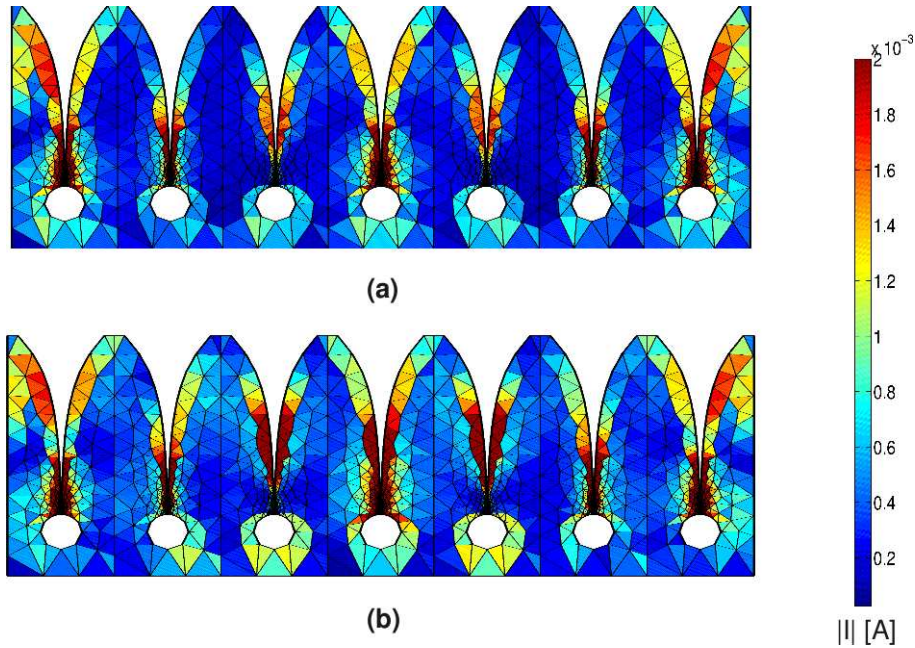


Figure 5.10: The surface current distribution on a 7×1 Vivaldi array calculated with (a) a direct solution technique and (b) the CBFM approach.

5.2.2 A linearly polarised 7×1 Vivaldi array

The geometry of the 7×1 Vivaldi array is presented in Figure 5.3 with the port numbering as illustrated. The surface current distribution for the array was calculated with the CBFM and direct methods respectively and are illustrated in Figure 5.10(a) and Figure 5.10(b). In Figures 5.11(a) and 5.11(b), the E and H-plane directivity patterns of the array are presented. The results were calculated at 2 GHz, with all the ports excited at the same time. The array is discretized with 8,122 RWG unknowns. The total number of CBFs is 19, and includes only primary excitation effects, i.e. from the applied voltage.

The relative error norm percentages, $\epsilon_{\%}$, for the calculated quantities are summarised in Table 5.2. The solution times obtained for the direct and CBFM approach when solving the 7×1 array configuration are illustrated in Table 5.3. The solution times were calculated on a desktop computer equipped with an Intel Core2 Duo 2.8 GHz processor with 4 GBytes of RAM. The run-times associated with the CBFM approach includes that obtained when the "primary" CBFs calculated in the previous Subsection are recycled to synthesise the 7×1 array.

For the larger array configuration, the relative error for the RWG expansion coefficient vector, I_{RWG} , increased to 11.77% which is higher than that obtained for the 3×1 array presented in the previous Subsection. The reason for this being that only "primary" CBFs are used in

the CBFM formulation. By incorporating "secondary" CBFs with the technique explained in Subsection 5.1.4, the accuracy can be improved even further.

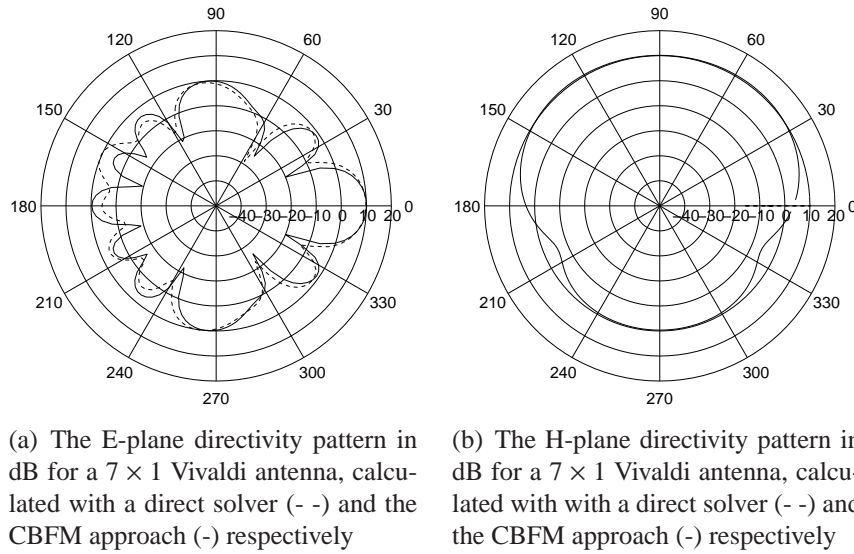


Figure 5.11: Comparing the E and H-plane directivity patterns for a 7×1 Vivaldi antenna at a frequency of 2 GHz, calculated with the CBFM and a direct solver.

Table 5.2: The normalised error percentage obtained by GMoM when calculating various quantities for a 7×1 Vivaldi antenna with the CBFM and direct solution techniques.

Calculated quantity	$\epsilon_{\%}$
$\{I\}$	11.77%
maximum E-plane directivity	2.57%
maximum H-plane directivity	4.76%

Table 5.3: The solution run-times obtained by the CBFM and direct solution techniques when calculating the surface current distribution for a 7×1 Vivaldi antenna at a centre frequency of 2 GHz. The results are generated on an Intel Core2 Duo 2.8 GHz processor equipped with 4 GBytes of RAM.

Method	Solution time [seconds]
Direct	226.8
CBFM without using pre-calculated sub-array data	43.40
CBFM using pre-calculated sub-array data	9.57

From Table 5.3 it can be seen that the CBFM solution time related to this problem size reduces by a factor of roughly 5 when compared to a direct solver. By using the resulting primary CBFs generated for the sub-arrays in the previous Subsection to synthesise the 7×1 configuration, the CBFM solution time reduces even further by a factor of nearly 22 while still maintaining a reasonable degree of accuracy.

In the remainder of this Chapter, the parallelisation of the CBFM will be considered as a means of further reducing the simulation runtime associated with this technique.

5.3 Parallelisation of the CBFM

In this Section, the parallelisation of the CBFM will be discussed from a general viewpoint, i.e. not related to any particular FPA configuration or scattering problem. Each of the steps outlined in Figure 5.2 for the general CBFM formulation will be considered, after which numerical results will be presented for the PEC scattering problem⁷ (Fig. 2.5) introduced in Chapter 2 on page 12. In each Subsection, reference will be made to the underlying MPI routines that can be used for the parallelisation.

5.3.1 Geometry Setup - from a domain decomposition point of view

The basic premise behind the CBFM is to divide a complex structure such as the PEC scatterer illustrated in Figure 2.5 or the Vivaldi array presented in Figure 5.3 into smaller sub-domains on which a set of "primary" and (if required) "secondary" CBFs can be generated. This inherent domain-decomposition makes the CBFM particularly well suited for parallelization schemes based on distributed programming models using the MPI standard [46].

The key concept behind the parallelisation of any algorithm is to ensure that the total computational work and memory usage are distributed more or less equally between the processes involved in the computation. In the context of the CBFM it is therefore optimal, from a computational point of view, to allocate a more or less equal number of sub-domains to each process [46].

The distribution of the memory requirement (dominated by storing the MoM impedance matrix) may be accomplished by noting that for each of the sub-domains, a number of sub-matrices of the form $[Z_{RWG}^{(p,q)}]$ will be required. When $p = q$, the sub-matrix represent the local interactions within the subdomain p or q and when $p \neq q$, it represents the mutual reaction integrals involving the RWGs on these domains respectively.

If we consider an example where the total number of sub-domains are $M = 4$, the MoM impedance matrix, Z_{RWG} , can be expressed in terms of these sub-matrices as,

$$[Z_{RWG}] = \begin{bmatrix} [Z_{RWG}^{(1,1)}] & [Z_{RWG}^{(1,2)}] & [Z_{RWG}^{(1,3)}] & [Z_{RWG}^{(1,4)}] \\ [Z_{RWG}^{(2,1)}] & [Z_{RWG}^{(2,2)}] & [Z_{RWG}^{(2,3)}] & [Z_{RWG}^{(2,4)}] \\ [Z_{RWG}^{(3,1)}] & [Z_{RWG}^{(3,2)}] & [Z_{RWG}^{(3,3)}] & [Z_{RWG}^{(3,4)}] \\ [Z_{RWG}^{(4,1)}] & [Z_{RWG}^{(4,2)}] & [Z_{RWG}^{(4,3)}] & [Z_{RWG}^{(4,4)}] \end{bmatrix} \quad (5.9)$$

If the task of generating the "primary" and "secondary" CBFs for each sub-domain is allocated to each of the processes according to their rank, i.e. a unique process identification number within the parallel process pool, the impedance matrix may be distributed amongst the processes as,

⁷The same underlying parallelization methods can however be applied to any FPA structure. The reason for selecting the scattering problem, is that the number of unknowns and number of sub-domains can easily be controlled for measurement purposes in a HPC environment.

$$[Z_{RWG}]_{LOC} = \begin{cases} \begin{bmatrix} [Z_{RWG}^{(1,1)}] & [Z_{RWG}^{(1,2)}] & [Z_{RWG}^{(1,3)}] & [Z_{RWG}^{(1,4)}] \\ [Z_{RWG}^{(2,1)}] & [Z_{RWG}^{(2,2)}] & [Z_{RWG}^{(2,3)}] & [Z_{RWG}^{(2,4)}] \end{bmatrix} & \text{Process ID} = 0 \\ \begin{bmatrix} [Z_{RWG}^{(3,1)}] & [Z_{RWG}^{(3,2)}] & [Z_{RWG}^{(3,3)}] & [Z_{RWG}^{(3,4)}] \\ [Z_{RWG}^{(4,1)}] & [Z_{RWG}^{(4,2)}] & [Z_{RWG}^{(4,3)}] & [Z_{RWG}^{(4,4)}] \end{bmatrix} & \text{Process ID} = 1 \end{cases} \quad (5.10)$$

where for illustration purposes two processes are used with identification numbers 0 and 1, each storing a local copy of the distributed MoM impedance matrix, $[Z_{RWG}]_{LOC}$.

Furthermore, the primary excitation vector, i.e. $\{V_{RWG}\}$, is of $O(N_{RWG})$ and can be stored locally in each process's address space⁸.

Distributing the impedance matrix in this manner amongst the processes, balances the computational load in terms of memory usage and also in terms of the computational burden associated with generating the CBFs, as will be discussed in the following Subsections. The example used in this Subsection will be extended in each of the following Subsections.

5.3.2 Calculating the "primary" CBFs

For the purpose of generating the primary CBFs, no inter-process communication is required between the process pool, as each process contains all the necessary information required to compute the set of CBFs associated with the sub-domains allocated to that particular process. Consider for example generating the primary CBF, $\{J_{CBFM,prim}^{(1)}\}$ on sub-domain $p = 1$,

$$[Z_{RWG}^{(1,1)}] \{J_{CBFM,prim}^{(1)}\} = \{V_{RWG}^{(1)}\} \quad (5.11)$$

Both $[Z_{RWG}^{(1,1)}]$ and $\{V_{RWG}^{(1)}\}$ are located in the address space of Process 0. Similarly the "primary" CBF of sub-domain $p = 3$, can be carried out at the same time on Process 1 without requiring any information from Process 0. Following this manner, all processes therefore compute their own set of "primary" CBFs concurrently.

5.3.3 Calculating the "secondary" CBFs

Generating the "secondary" CBFs requires that the excitations associated with the "primary" CBFs of all domains not included in the sub-domain scope of the corresponding process, be communicated to the process that includes the domain on which the secondary CBFs are required. As an example, consider the calculation of the "secondary" CBF on sub-domain $p = 1$, that is induced by the "primary" CBF on sub-domain $q = 3$, i.e.

$$[Z_{RWG}^{(1,1)}] \{J_{CBFM,sec}^{(1)}\} = -[Z_{RWG}^{(1,3)}] \{J_{CBFM,prim}^{(3)}\} \quad (5.12)$$

The only data not residing on Process 0 in Eq. (5.12) is that of the primary CBF, $\{J_{CBFM,prim}^{(3)}\}$, associated with sub-domain $q = 3$.

⁸When considering extremely large problems, it may be necessary to distribute the excitation vector amongst the processes. This can be accomplished by following exactly the same method as depicted in Eq. (5.10) for the impedance matrix.

A collective MPI broadcasting command, `MPI-Allgather`, can be used to ensure that each process has, in its address space, the "primary" CBFs of all the other sub-domains⁹. The transmittal of data between the processes occurs simultaneously when collective MPI-communication is used¹⁰.

When the necessary data is gathered by each process, Eq. (5.12) can then be calculated concurrently, where each process thereby constructs its own set of additional "secondary" CBFs.

5.3.4 Generating the reduced matrix equation

By using a Galerkin testing procedure with the newly generated CBFs, a low-rank reduced matrix equation can be constructed as explained in Section 5.1. Computationally, this step is very expensive, as it requires MK^2 complex matrix-vector multiplications, where M is the total number of CBFM sub-domains and K is the average number of CBFs per sub-domain. Each process can however compute its own segment of the reduced impedance matrix, which thereby significantly reduces the computational overhead associated with step. This is illustrated in the following example.

When considering the $M = 4$ sub-domain example introduced in Section 5.3, with a single "primary" CBF and three "secondary" CBFs calculated on each of the domains, the reduced impedance matrix can be expressed as,

$$[Z_{CBFM}] = \begin{bmatrix} [Z_{CBFM}^{(1,1)}] & [Z_{CBFM}^{(1,2)}] & [Z_{CBFM}^{(1,3)}] & [Z_{CBFM}^{(1,4)}] \\ [Z_{CBFM}^{(2,1)}] & [Z_{CBFM}^{(2,2)}] & [Z_{CBFM}^{(2,3)}] & [Z_{CBFM}^{(2,4)}] \\ [Z_{CBFM}^{(3,1)}] & [Z_{CBFM}^{(3,2)}] & [Z_{CBFM}^{(3,3)}] & [Z_{CBFM}^{(3,4)}] \\ [Z_{CBFM}^{(4,1)}] & [Z_{CBFM}^{(4,2)}] & [Z_{CBFM}^{(4,3)}] & [Z_{CBFM}^{(4,4)}] \end{bmatrix} \quad (5.13)$$

with each of the entries calculated as,

$$[Z_{CBFM}^{(p,q)}] = \langle [J_{CBFM}^{(p)}]^T, [Z_{RWG}^{(p,q)}] [J_{CBFM}^{(q)}] \rangle \quad (5.14)$$

All the data required by each process to calculate the reduced CBFM equation is located in the address space of that process, if the computation of Eq. (5.13) is distributed in exactly the same manner as that of the MoM impedance matrix (Eq. (5.10)), i.e.

$$[Z_{CBFM}]_{LOC} = \begin{cases} \begin{bmatrix} [Z_{CBFM}^{(1,1)}] & [Z_{CBFM}^{(1,2)}] & [Z_{CBFM}^{(1,3)}] & [Z_{CBFM}^{(1,4)}] \\ [Z_{CBFM}^{(2,1)}] & [Z_{CBFM}^{(2,2)}] & [Z_{CBFM}^{(2,3)}] & [Z_{CBFM}^{(2,4)}] \end{bmatrix} & \text{Process ID} = 0 \\ \begin{bmatrix} [Z_{CBFM}^{(3,1)}] & [Z_{CBFM}^{(3,2)}] & [Z_{CBFM}^{(3,3)}] & [Z_{CBFM}^{(3,4)}] \\ [Z_{CBFM}^{(4,1)}] & [Z_{CBFM}^{(4,2)}] & [Z_{CBFM}^{(4,3)}] & [Z_{CBFM}^{(4,4)}] \end{bmatrix} & \text{Process ID} = 1 \end{cases} \quad (5.15)$$

These segments of the reduced impedance matrix can then be gathered on a single process, typically designated as "root" or "master". The collective `MPI-Gather` routine can be used for the data-collection. The remainder of the CBFM algorithm is performed sequentially on the "root" process, and entails the direct solution of the reduced impedance matrix and the mapping

⁹From a computational manner

¹⁰Care should be taken to ensure that the data is re-assembled correctly depending on whether arrays are stored in *row-major* or *column-major* order

of the solution to the global problem domain to yield the discretised surface current distribution. The reason for performing the final steps of the CBFM formulation sequentially, is that the reduced data-sets are involved in the computations and that parallelisation will therefore lead to unnecessary communication overhead.

In the following Subsection, the scalability of the parallelised CBFM techniques explained in this and the previous Subsections are presented in terms of speed-up and efficiency measurements obtained in a HPC environment.

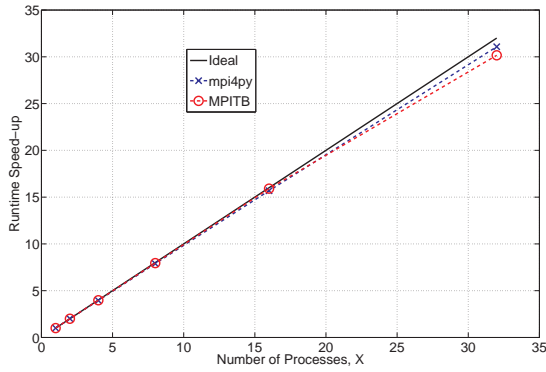
5.3.5 CBFM parallelisation results

In Figure 5.12(a), the measured speed-up related to the "primary" CBF generation is compared to that of the ideal case, i.e. $S = X$ where X is the number of processes used in the simulation. For comparison, the CBFM algorithm was implemented with two different MPI bindings, namely mpi4py (Python) and MPITB (MATLAB). The results are associated with a PEC scattering configuration (similar to that presented in Figure 2.5 on page 12) and is measured on the iQudu infrastructure of which an overview is given in Section 3.3 on page 22. The entire problem consists of roughly 15,000 RWG unknowns and a total of 1024 CBFs was used in the formulation.

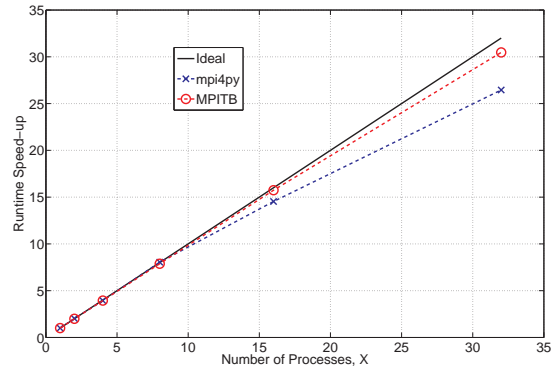
The speed-up results presented in Figure 5.12(a) illustrates that the parallel CBFM implementation pertaining to the generation of the "primary" CBFs scales well for both mpi4py and MPITB when compared to the ideal case. The reason for this being that no inter-process communication is required during this step.

The speed-up results associated with the generation of the "secondary" CBFs (Figure 5.12(b)) and that of the reduced matrix equation (Figure 5.12(c)) are subject to inter-process communication and thereby deviate from that obtained by the ideal case.

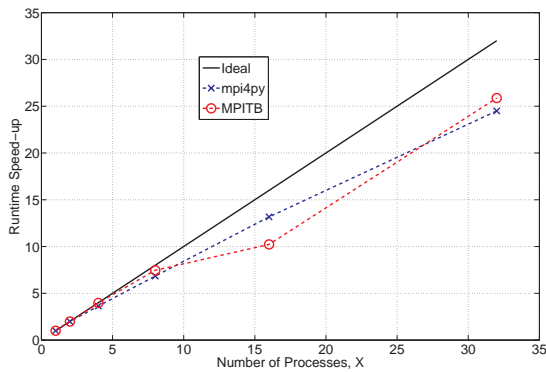
In Figure 5.12(d), the total speed-up and efficiency pertaining to all the CBFM steps are presented with the ideal results once again added for comparison. It is evident that the effect of inter-process communication introduced in the calculation of the "secondary" CBFs and also that of the reduced matrix equation leads to a deterioration in the scalability of the algorithm. Nonetheless, the total results obtained by mpi4py and MPITB still presents an adequate scalability when compared to the ideal case. For 32 processes, mpi4py obtains a speed-up of nearly 25 and MPITB obtains a speed-up of 22 (compared to the ideal case of $S = 32$). For all the simulations, the efficiency of the algorithm remains above 60%. The reason that mpi4py performs slightly better as obtained from the results, can be attributed to a higher latency associated with the collective MPI routines implemented with MPITB when compared to that of mpi4py, as was illustrated in Section 3.8.1 on page 27.



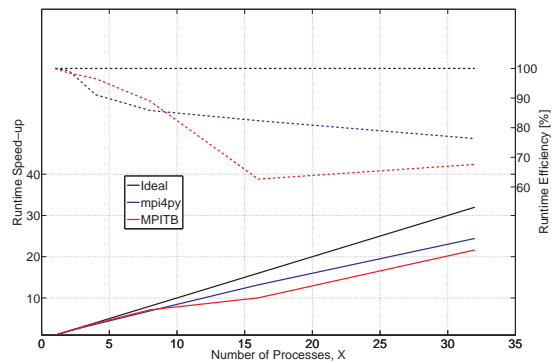
(a) The runtime speed-up associated with the "primary" CBFs setup, as obtained by mpi4py and MPITB respectively



(b) The runtime speed-up associated with the "secondary" CBFs setup, as obtained by mpi4py and MPITB respectively



(c) The runtime speed-up associated with the reduced matrix equation setup, as obtained by mpi4py and MPITB respectively



(d) The total runtime speed-up and efficiency, as obtained by mpi4py and MPITB for a general CBFM formulation

Figure 5.12: Comparing the runtime speed-up and efficiency obtained by mpi4py and MPITB when applied to a general CBFM formulation

5.4 Conclusions

In this Chapter, the CBFM formulation was introduced for interconnected radiating elements. The array configurations used in the discussions are that of a 3×1 and 7×1 linearly polarised Vivaldi array. Numerical results obtained by the CBFM solution for both of these arrays compared well to that pertaining to a direct solver. For the 7×1 array, a significant reduction in simulation runtime was observed, especially when using the pre-calculated "primary" CBFs of the 3×1 Vivaldi array to synthesise that of the 7×1 configuration.

As a means of further reducing the simulation runtime associated with the general CBFM formulation, the author explained the key concepts behind the parallelisations of this algorithm from a domain-decomposition point of view using MPI. Measured speed-up and efficiency results compared well to that corresponding to the ideal case and illustrates the high degree of scalability that can be obtained with a parallel CBFM implementation.

The underlying theory presented in this Chapter can be applied to a wide variety of interconnected FPA structures and also extended to incorporate dual polarised configurations. Simulation results can then be obtained in a fraction of the time (compared to that obtained by direct solvers), while still maintaining an adequate level accuracy.

Chapter 6

General Conclusions

In this thesis, a collection of methods was researched and developed for the efficient numerical simulation of large FPA configurations. In Chapter 2, an overview was presented for various well-known CEM techniques that are particularly suited to the simulation of large electromagnetic structures in general. These techniques included the use of HPC resources, the FMM and its multilevel extension the MLFMA and also macro-domain techniques such as the CBFM. The aforementioned methods were compared in terms of simulation runtimes and also memory usage. In Chapter 2 it was also illustrated that methods such as the CBFM and the MLFMA present better runtimes and memory usage when compared to direct solvers applied to problems discretized with a large number of unknowns. Methods such as the MLFMA however rely on iterative solvers, which may suffer from convergence problems if the underlying matrix equation is ill-conditioned. For this reason, the focus was directed towards improving the simulation runtime that incorporates direct solution methods, based on the full-wave MoM formulation, while still maintaining a sufficient degree of accuracy.

In Chapter 3, the efficiency of the full-wave MoM solver in FEKO was applied in a HPC environment to quantify the degree to which parallel computing can be used to simulate large FPA structures. The simulation results were focussed on measuring the speed-up and efficiency for various sized Vivaldi antenna arrays ranging from 550 to 33,624 RWG unknowns, i.e. arrays consisting of 1 to 64 elements. Absolute runtime could be improved by a factor of nearly 20 for a large array consisting of 64 Vivaldi elements by selecting an optimal number of nodes for the simulation and also a fast interconnect such as Infiniband. Practical considerations were given for selecting this optimal number of nodes in terms of parallel efficiency and memory usage in order to fully utilise the resources of the HPC infrastructure.

To further improve the runtime efficiency of the MoM, the focus was directed towards macro-domain techniques such as the CBFM. In order to evaluate the accuracy of this method, the author developed a MoM kernel (GMoM) that could be used to analyse an arbitrary collection of PEC structures (radiators and scatterers), as illustrated in Chapter 4. The accuracy of GMoM was validated by comparing results associated with a variety of problems to that obtained by FEKO and also with results available in the literature. For the validation, various quantities such as directivity patterns, RWG expansion coefficients and S-parameters were compared in terms of the relative error norm percentage, $\epsilon_{\%}$.

In Chapter 5, GMoM formed the underlying basis for implementing the CBFM formulation applied to an interconnected array of Vivaldi elements. The CBFM formulation was discussed in terms of its fundamental components, i.e. dividing the problem into a number of smaller sub-domains and approximating the induced current on each in terms of "primary" and "secondary" CBFS. These macro-domain basis functions can then be used to construct a reduced

matrix equation that is significantly smaller than that of the matrix equation associated with the problem as a whole. The advantage of this, is that direct solvers can be used for all steps related to the CBFM. Results presented for a 7×1 linearly polarised Vivaldi array illustrate that the simulation runtimes can be reduced by a factor of 22 when using pre-calculated CBFs computed for a 3×1 array configuration. The accuracy obtained by the CBFM for the 7×1 configuration was evaluated in terms of various criteria such as E and H-plane directivity patterns and also the RWG expansion coefficients of the discretized structure. For the latter, the CBFM obtained a relative error norm percentage of 11.77% when compared to that calculated with a direct solver. Future work can however be focussed on improving this accuracy by incorporating "secondary" CBFs, as discussed in Section 5.1.4. In Chapter 5, the parallelisation of a general CBFM formulation was also considered using the MPI implementations associated with the interpreted programming languages Python and Octave. Speed-up and efficiency results measured on the HPC infrastructure provided by the iQudu, compared well to those of the ideal cases. This serves to illustrate the high degree of scalability that a parallel CBFM formulation can obtain.

In conclusion, ongoing research such as that conducted by the author in this thesis may prove useful to develop a more efficient design process for FPA structures and perhaps even general electromagnetic problems associated with the SKA and MeerKAT.

Appendix A

Appendix A - Thin Strip MoM model of a Dipole antenna

The purpose of this Section is to evaluate the accuracy that is obtained by modelling a wire-radiator, such as a simple half-wavelength dipole antenna, with a thin strip discretized with RWG basis-functions.

A.1 The antenna model

A half-wavelength dipole antenna, resonant at $f_c = 2$ GHz, was modelled with a thin PEC strip and a wire model respectively. A feeding-edge excitation [35] was used to excite the thin strip dipole antenna as implemented in GMoM. The wire dipole-model was constructed with the commercial CEM simulation package FEKO, and was excited with a voltage source. The geometry is presented in Figures A.1 (a) and (b) respectively. The dimensions of each model are illustrated in Table A.1.

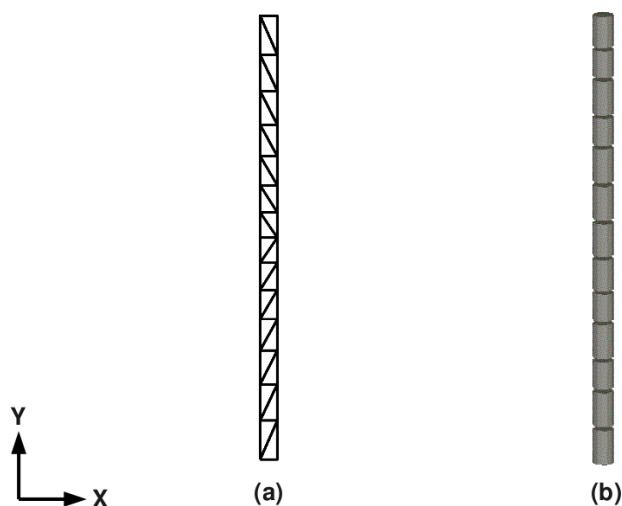


Figure A.1: The geometry of a half-wavelength dipole antenna modelled with (a) a thin strip discretized with RWG basis functions constructed in GMoM and (b) a wire-model constructed in FEKO.

When investigating the dimensions in Table A.1, it is noted that a wire-radius of 1.5 mm does not entirely conform to the equivalent radius of 1 mm (cf. [47]) for a strip-width of 4 mm.

Table A.1: Dimensions associated with the half-wavelength dipole models

	height(h) [mm]	width(w) [mm]
Strip Dipole Model	74.9	4
Wire Dipole Model	74.9	3

The reason for this, is that the formulated relationship between the strip-width and wire-radius in [47], i.e. $r_{eq} = w_{strip}/4$, holds for cases where the strip's length is orders of magnitude larger than its width.

A.2 Simulation results

The dipole antenna models were simulated with FEKO and GMoM respectively over a frequency range, $f_{min} = 1$ GHz to $f_{max} = 3$ GHz. The results are illustrated in Figure A.2 and Figure A.3 respectively, where the magnitude and phase of the input reflection coefficient are compared for each of the simulations.

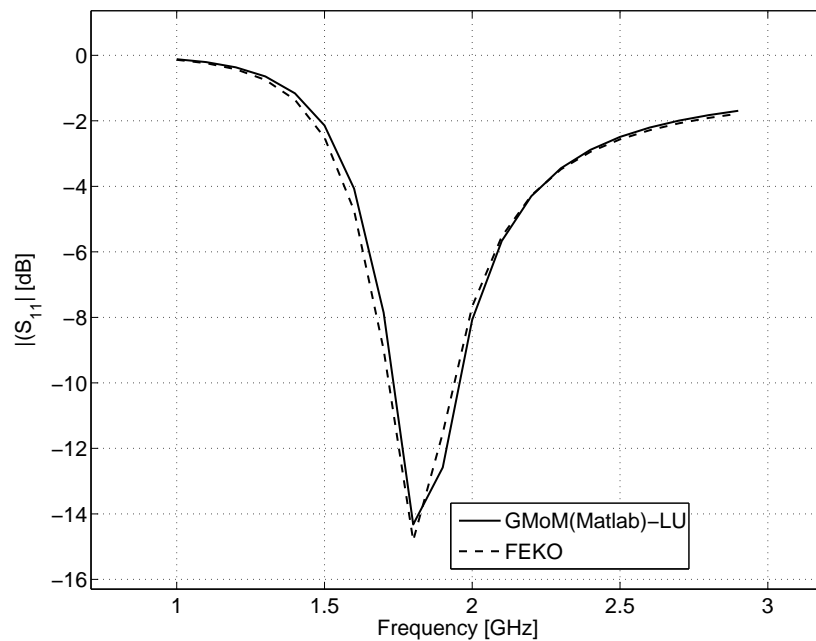


Figure A.2: Magnitude of the input reflection coefficient (S_{11}) in dB calculated with the RWG model (GMoM) and FEKO respectively.

By calculating the relative error norm percentage, $\epsilon_{\%}$, for the input reflection coefficients, it can be determined that the RWG strip dipole-model differs by approximately 10.88%, for a discretization of $\lambda/15^1$ at $f_{max} = 3$ GHz, from the results obtained by FEKO.

Excellent agreement is also observed between the E-plane directivity patterns of the RWG and FEKO dipole-models, as illustrated in Figure A.4.

¹This corresponds to 27 RWG basis-functions

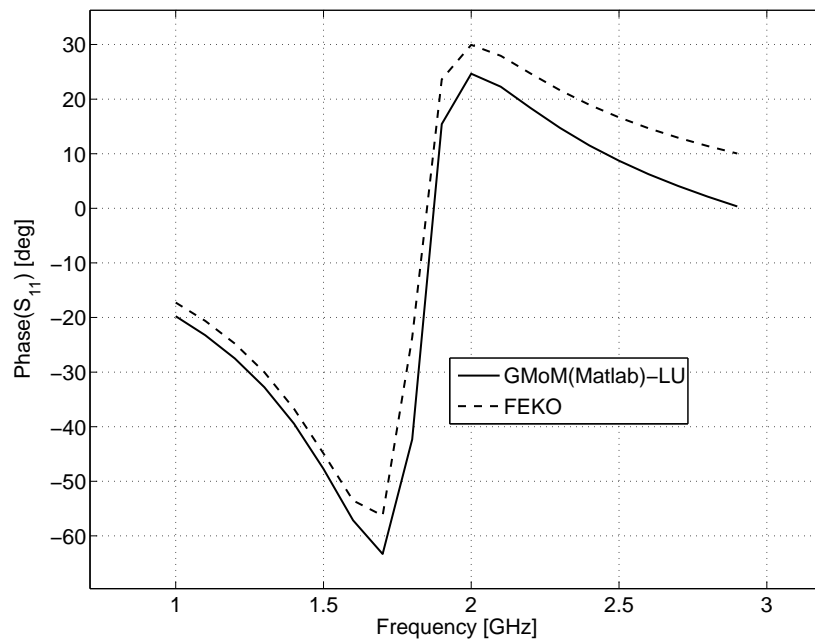


Figure A.3: Phase of the input reflection coefficient (S_{11}) in degrees calculated with the RWG model (GMoM) and FEKO respectively.

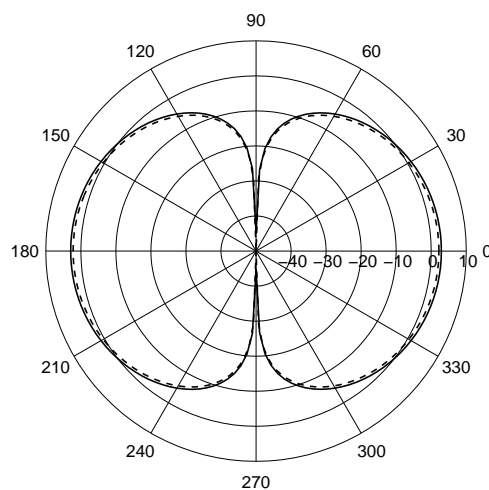


Figure A.4: Directivity pattern in dB, calculated with the RWG model (GMoM) (-) and FEKO (- -) respectively, at $f_c = 2$ GHz.

A.3 Conclusion

In this section, a half-wavelength dipole antenna was modelled with a thin wire constructed with FEKO and a thin strip model discretized with RWG basis functions (implemented in GMoM). Numerical results for the input reflection coefficient of the RWG strip-model differ by less than 11% compared to that obtained by the FEKO wire-model over a frequency range of $f_{min} = 1$ GHz to $f_{max} = 3$ GHz. The directivity patterns of the antenna models, also yield excellent agreement at the centre frequency.

Appendix B

Appendix B - Code Listing for the Ping-pong test

The ping-pong implementations used to measure the *bandwidth* and *latency* associated with the message passing interface (MPI) bindings of various programming languages are illustrated in Listings B.1 to B.3. For brevity, only the most relevant code sections are illustrated.

B.0.1 C implementation of the Ping-pong test

```
2  for (length = MIN_BYTE_SIZE; length <= MAX_BYTE_SIZE; \  
3      length = length + BYTE_STEP)  
4  {  
5      MPI_Barrier(MPI_COMM_WORLD);  
6      if (myId == MASTER)  
7      {  
8          startTime = MPI_Wtime();  
9          for (i = 0; i < NUM_OF_REPS; i++)  
10         {  
11             MPI_Send(buffer, length, MPI_BYTE, 1, \  
12                 PING, MPI_COMM_WORLD);  
13             MPI_Recv(buffer, length, MPI_BYTE, 1, \  
14                 PONG, MPI_COMM_WORLD, &status);  
15             endTime = MPI_Wtime();  
16         }  
17         endTime = MPI_Wtime();  
18         T = endTime - startTime;  
19     } else // Slave  
20     {  
21         for (i = 0; i < NUM_OF_REPS; i++)  
22         {  
23             MPI_Recv(buffer, length, MPI_BYTE, 0, \  
24                 PING, MPI_COMM_WORLD, &status);  
25             MPI_Send(buffer, length, MPI_BYTE, 0, \  
26                 PONG, MPI_COMM_WORLD);  
27         }  
28     }
```



```

30     if (myId == 0)
31     {
32         printf("%lf %g # length , time[sec]\n",
33             length , (T/(2*NUM_OF_REPS)));
34     }
35 }
36 }

```

Listing B.1: Ping-Pong program implemented in C

B.0.2 Octave implementation of the Ping-pong test

```

2 n = 1;
3 for NUM_OF_BYTES = linspace(MIN_BYTE_SIZE, MAX_BYTE_SIZE, ...
4     numDataPoints)
5     MPI_Barrier(MPI_COMM_WORLD);
6     if (rnk == ROOT_PROC)
7         T=clock;
8         for i=1:TRANSFER_REPEATS
9             MPI_Recv(rxData ,SLAVE_PROC,TAG, ...
10                MPI_COMM_WORLD);
11             MPI_Send(txData ,SLAVE_PROC,TAG, ...
12                MPI_COMM_WORLD);
13         end
14         T=etime(clock ,T);
15     else % SLAVE_PROC
16         for i=1:TRANSFER_REPEATS
17             MPI_Send(txData ,ROOT_PROC,TAG, ...
18                MPI_COMM_WORLD);
19             MPI_Recv(rxData ,ROOT_PROC,TAG, ...
20                MPI_COMM_WORLD);
21         end
22     end % MST / SLV
23
24     if (rnk == ROOT_PROC)
25         results(n,1) = NUM_OF_BYTES;
26         results(n,2) = T/2/TRANSFER_REPEATS;
27         fprintf(1,"%10d\t%9.6f\n", results(n,:));
28     end
29     n = n+1
30 end

```

Listing B.2: Ping-Pong program implemented in Octave

B.0.3 Python implementation of the Ping-pong test

```

2 n = 0
3 for NUM_OF_BYTES in range(MIN_BYTE_SIZE, \
4     MAX_BYTE_SIZE + BYTE_STEP, BYTE_STEP):
5     MPI.COMM_WORLD.Barrier()
6     if(myRank == ROOT_PROC):
7         startTime = MPI.Wtime()
8         for i in range(0,TRANSFER_REPEATS):
9             MPI.COMM_WORLD.Send([ txData , NUM_OF_BYTES, \
10                MPI.BYTE], SLAVE_PROC, 1)
11            MPI.COMM_WORLD.Recv([ rxData , NUM_OF_BYTES, MPI.BYTE],
12                SLAVE_PROC, 1, status)
13
14            endTime = MPI.Wtime()
15            rootTime = float(endTime - startTime) / (2.0 * \
16                float(TRANSFER_REPEATS))
17            rootTimes[n] = rootTime
18            rootMsgSize[n] = float(NUM_OF_BYTES)
19            logFile.write('Message Size [BYTES = %d] and \
20                [Time = %.6f]\n' % \
21                (NUM_OF_BYTES, rootTime))
22
23            if(myRank == SLAVE_PROC):
24                startTime = MPI.Wtime()
25                for i in range(0,TRANSFER_REPEATS):
26                    MPI.COMM_WORLD.Recv([ rxData , NUM_OF_BYTES, MPI.BYTE],
27                        ROOT_PROC, 1, status)
28                    MPI.COMM_WORLD.Send([ txData , NUM_OF_BYTES, MPI.BYTE],
29                        ROOT_PROC, 1)
30
31                endTime = MPI.Wtime()
32
33                slaveTime = float(endTime - startTime) / (2.0 * \
34                    float(TRANSFER_REPEATS))
35                slaveTimes[n] = slaveTime
36                slaveMsgSize[n] = float(NUM_OF_BYTES)

```

Listing B.3: Ping-Pong program implemented in Python

References

- [1] Davidson, D.B.: *Computational Electromagnetics for RF and Microwave Engineering*. Cambridge University Press, 2005.
- [2] S.M. Rao, D.R. Wilton, A.G.: Electromagnetic Scattering by Surfaces of Arbitrary Shape. *IEEE Transactions on Antennas and Propagations*, vol. 30, p. 409 ... 418, 1982.
- [3] The South-African Square Kilometre Array Project Page.
Available at: <http://www.ska.ac.za/>
- [4] The Australian SKA Project (ASKAP) Page.
Available at: <http://www.atnf.csiro.au/projects/askap/>
- [5] Veidt, B.: SKA Memo 71: Focal-Plane Architectures: Horn Clusters vs. Phased-Array Techniques. *SKA Memos*, 02 2006.
Available at: www.skatelescope.org/pages/memos
- [6] Ludick, D.: Focal Plane Antennas for the Square Kilometre Array, November 2007.
- [7] FEKO homepage.
Available at: <http://www.feko.info>
- [8] Karniadakis, G. and R.M., K.: *Parallel Scientific Computing in C++ and MPI*. Cambridge University Press Cambridge, 2003.
- [9] Moore, G.E.: Cramming more components onto integrated circuits. *Electronics*, vol. 38, 04 1965.
- [10] Hennessy, J. and Patterson, D.: *Computer architecture: a quantitative approach*. Morgan Kaufmann, 2003.
- [11] Rabaey, J.M.: *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, 1996.
- [12] Flynn, M.J.: *Some computer organizations and their effectiveness*, vol. 21. 1972.
- [13] Chen, R., Xu, K. and Ding, J.: Acceleration of MoM Solver for Scattering Using Graphics Processing Units (GPUs). *The cross-strait three wireless technology seminar, Taipei, Oriental Institute of Technology*, 2008.
- [14] Lay, D.C.: *Linear Algebra and its applications*. Greg Tobin, 2003.
- [15] Top 500 Supercomputer Sites. <http://www.top500.org>, 2009.
Available at: <http://www.top500.org>
- [16] N. Engheta, W. D. Murphy, V.R. and Vassiliou, M.S.: The fast multipole method (FMM) for electromagnetic scattering problems. *IEEE Transactions on Antennas and Propagations*, vol. 40, p. 634 ... 641, 1992.

- [17] Cipra, B.A.: The Best of the 20th Century: Editors Name Top 10 Algorithms. *SIAM News*, vol. 33, 1999.
- [18] Peterson, A., Ray, S. and Mittra, R.: *Computational methods for electromagnetics*. IEEE press New York, 1998.
- [19] R. Coifman, V.R. and Wandzura, S.: The fast multipole method (FMM) for electromagnetic scattering problems. *IEEE Transactions on Antennas and Propagations*, vol. 35, p. 7 ... 12, 1993.
- [20] Lu, W., Zhao, Q. and Cui, T.: Sub-Entire-Domain Basis Function Method for Irrectangular Periodic Structures. *Progress in Electromagnetic Research B*, vol. 5, p. 91...105, 2008.
- [21] Suter, E. and Mosig, J.: A Sub-Domain Multilevel Approach for the efficient MoM analysis of large planar antennas. *Microwave and Optical Technology Letters*, vol. 26, p. 270...277, 2000.
- [22] Matekovits, L., Laza, V. and Vecchi, G.: Analysis of Large Complex Structures With the Synthetic-Functions Approach. *IEEE Transactions on Antennas and Propagations*, vol. 55, p. 2509...2521, 2007.
- [23] Prakash, V. and Mittra, R.: Characteristic Basis Function Method: A new technique for efficient solution of Mehod of Moments Matrix Equations. *Microwave and Optical Technology Letters*, vol. 36, p. 95...100, 2003.
- [24] Maaskant, R., Tijhuis, A.G., Mittra, R., Ivashina, M., van Cappellen, W. and Arts, M.: Hybridization of Efficient Modeling Techniques for Fast Analysis of Large-Scale Antenna Structures in the Context of the Square Kilometre Array Project. *Proceedings of the 38th European Microwave Conference*, p. 837 ... 840, 2008.
- [25] Mittra, R. and Du, K.: Characteristic Basis Function Method For Iteration-Free Solution of Large Method of Moments Problems. *Progress In Electromagnetics Research*, vol. 6, no. B, p. 307...336, December 2008.
- [26] Chan, K., Lam, K. and Chan, C.: Modelling of Microstrip Reflector arrays using the Characteristic Basis Function Approach. *URSI EMTS*, p. 334...336, 2004.
- [27] *FEKO User's Manual Suite 5.4*, July 2007.
Available at: <http://www.feko.info>
- [28] Jakobus, U.: Parallel Computation of Electromagnetic Fields based on Integral Equations. *Transactions of the High Performance Computing Center Stuttgart (HLRS)*, 1998.
- [29] LAPACK - Linear Algebra PACKage. <http://www.netlib.org/lapack/index.html>.
Available at: <http://www.netlib.org/lapack/index.html>
- [30] ScaLAPACK Home Page. <http://www.netlib.org/scalapack/>.
Available at: <http://www.netlib.org/scalapack/>
- [31] Information on the CHPC.
Available at: <http://www.chpc.ac.za>
- [32] Davidson, D.: *Parallel Algorithms for Electromagnetic Moment Method Formulations*. Ph.D. thesis, University of Stellenbosch, November 1991.
- [33] Riesen, R., Brightwell, R. and Maccabe, A.B.: Measuring MPI Latency Variance. *Euro PVM and MPI, LNCS 2840*, p. 112...116, 2003.

- [34] Frese, H. and Knipp, H.: Performance Evaluation of MPI and MPICH on the Cray T3E. *Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)*, 1999.
- [35] Makarov, S.: MoM Antenna Simulations with *Matlab*: RWG Basis Functions. *IEEE Antennas and Propagations Magazine*, vol. 43, p. 100 ... 107, 2001.
- [36] Remacle, C.G.J.: An introduction to geometrical modelling and mesh generation with Gmsh. 2008.
- [37] Scott, J. and Peterson, A.: Quadrature Rules for Numerical Integration over Triangles and Tetrahedra. *IEEE Transactions on Antennas and Propagations*, vol. 38, p. 100...102, 1996.
- [38] Duvenant, D.: High Degree Efficient Symmetrical Gaussian Quadrature Rules for the Triangle. *International Journal for Numerical Methods in Engineering*, vol. 21, p. 1129...1148, 1985.
- [39] Maaskant, R., Mittra, R. and Tijhuis, A.G.: Fast Analysis of Large Antenna Arrays Using the Characteristic Basis Function Method and the Adaptive Cross Approximation Algorithm. *IEEE Transactions on Antennas and Propagations*, vol. 56, p. 3440 ... 3451, 2008.
- [40] Leat, C., N.V.Shuley and G.F.Stickley: Triangular-patch model of bowtie antennas: validation against Brown and Woodward. *IEEE Proceedings: Microwaves, Antennas and Propagation*, vol. 145, p. 465 ... 470, 1998.
- [41] Balanis, C.: *Antenna theory*. Wiley New York, 2005.
- [42] Shoon, J. and Schaubert, D.: A Parameter Study of Stripline-Fed Vivaldi Notch-Antenna Arrays. *IEEE Transactions on Antennas and Propagation*, vol. 47, 1999.
- [43] David, M.: Pozar. *Microwave Engineering*. 2005.
- [44] Stutzman, W. and G.A.Thiele: *Antenna Theory and Design*. John Wiley and Sons, 1981.
- [45] Maaskant, R., Mittra, R. and Tijhuis, A.G.: Application of Trapezoidal-Shaped Characteristic Basis Functions to Arrays of Electrically Interconnected Antenna Elements. *presented at the Int. Conf. Electromagn. Adv. Applicat. (ICEAA)*, 2007.
- [46] Lucente, E. and Monorchio, A.: A parallel iteration-free mom algorithm based on the characteristic basis function method. *Int. USRI Commission B - Electromagnetic Theory Symposium*, July 2007.
- [47] Butler, C.M.: The equivalent radius of a narrow conducting strip. *IEEE Transactions on Antennas and Propagations*, vol. 30, p. 755 ... 758, 1982.