3D to 2D Surface Mesh Parameterisation for the Purposes of Unstructured Transmission Line Modelling Method Simulations

Hayan Nasser, MEng

Thesis submitted to The University of Nottingham
for the degree of Doctor of Philosophy, September, 2015

Abstract

Small scale fabrication processes have led to the advent of very thin flexible devices such as RFID tags, flexible PCBs and smart clothing. In a geometrical sense, these present themselves as curved two dimensional surfaces embedded in a three dimensional domain. When simulating electromagnetic behaviour on these surfaces at low frequencies, a full 3D field model is not always necessary. Using 3D algorithms to solve these problems can result in a large portion of the computer memory and runtime being used to mesh and simulate areas of the domain that present little electromagnetic activity.

The theme of this thesis is concerned with the improvement of the runtime and memory consumption of electromagnetic simulations of these surfaces. The main contributions of this work are presented as an investigation into the feasibility of applying a 2D Unstructured Transmission Line Modelling method (UTLM) simulation to open, curved surfaces embedded in 3D space, by providing a one-to-one mapping of the geometry to a 2D flat plane.

First, an investigation into the various methods of how a computer represents unstructured meshes in its memory is presented, and how this affects the runtime of the simulation. The underlying mesh data structures used to represent the geometrical problem space can have a huge impact on the efficiency and memory consumption of the simulation. This investigation served to demonstrate that it is not just simply the optimisation of the simulation algorithms that facilitate improvements to the runtime and memory consumption of a simulation. How a computer understands

the connectivity of the mesh can have far greater impacts to the computational resources available.

The concepts of surface parameterisation are then introduced; a process of mapping curved surfaces embedded in a three dimensional domain to a flat two dimensional plane. By providing a one-to-one mapping of the geometry from the 3D domain to the 2D flat plane, a low frequency 2D unstructured TLM simulation can be applied, negating the need for 3D algorithms. Because this mapping is one-to-one, the results of the simulation can then be mapped back to 3D space for visualisation. Parameterisations will almost always introduce distortion to angle and area, and minimising this distortion is paramount to maintaining an accurate simulation. Test cases were used to measure the extent of this distortion, and the investigation concluded that Angle Based Flattening (ABF) and Least Squares Conformal Mapping (LSCM) methods resulted in the best quality parameterisations. Simulations were then conducted on these test cases as a demonstration of how UTLM can be performed on 2D surfaces, embedded in a 3D domain.

Acknowledgements

I would like to profoundly thank my supervisors Professor Phillip Sewell, Professor Trevor Benson, Dr Ana Vukovic and Dr Steve Greedy for their immense amount of guidence and support that enabled me to complete this work.

I extend my thanks to everyone in the lab - thank you all for the fruitful discussions. You all made this period of my life great fun.

I would also like to thank my family and friends. I am extremely grateful for your infinite encouragement, support and patience throughout.

Many thanks to the EPSRC, who provided the funding for this work.

Principal Symbols

ε	-	Permittivity
μ	-	Permeability
E	-	Electric Field
H	-	Magnetic Field
I	-	Current
J	-	Bessel Function
k	-	Wavenumber
V	-	Voltage
Z	-	Impedence
Y	-	Addmittance
F	-	Parameterisation Energy Functional
$\mathbf{p}$	-	Point in 3D space
$\mathbf{u}$	-	Point in 2D parameter space
(x, y, z)	-	Cartesian coordinates in 3D space
(u, v)	-	Parameter coordinates in 2D space

Acronyms

TLM	-	Transmission Line Modelling
UTLM	-	Unstructured Transmission Line Modelling
FDTD	-	Finite Difference Time-Domain
FEM	-	Finite Element Method
FFT	-	Fast Fourier Method
TM	-	Transverse Magnetic
DHP	-	Discrete Harmonic Parameterisation
DAP	-	Discrete Authalic Parameterisation
LSCM	-	Least Squares Conformal Mapping
MIPS	-	Most Isometric ParameterisationS
ABF	-	Angle Based Flattening

Publications and Awards

The following paper has arisen from the work described in this thesis:

Hayan Nasser, S. Greedy, T. Benson, A. Vukovic, P. Sewell, "3D to 2D Surface Mesh Parameterization for Unstructured Transmission Line Method Simulations" Computational Electromagnetics (ICCEM), 2015 IEEE International Conference on, pp. 338 - 340, 2015

The following awards have arisen from the work presented in this thesis:

Hayan Nasser, P. Sewell, A. Vukovic, T. Benson, S. Greedy, "3D to 2D Reduction for Electrical and Thermal Simulations", Electrical Systems and Optics Division Research Competition, University of Nottingham, *First Prize*, 2013.

Contents

Chapter 1

Introduction

Small scale fabrication processes have led to the advent of very thin flexible devices such as RFID tags, flexible PCBs and smart clothing. In a geometrical sense, these present themselves as curved two dimensional surfaces embedded in a three dimensional domain. When simulating electromagnetic behaviour on these surfaces at low frequencies, a full 3D field model is not always necessary. Using 3D algorithms to solve these problems can result in a large portion of the computer memory and runtime being used to mesh and simulate areas of the domain that present little electromagnetic activity.

The theme of this thesis is concerned with the improvement of the runtime and memory consumption of electromagnetic simulations of these surfaces. The main contributions of this work are presented as an investigation into the feasibility of applying a 2D unstructured Transmission Line Modelling method (TLM) simulation to open, curved surfaces embedded in 3D space, by providing a one-to-one mapping of the geometry to a 2D flat plane.

1.1 Background

Within the field of electromagnetics, simulation becomes an indispensable tool, playing a key role in the design stage of many devices and systems. And for good reason. Electromagnetism deals with invisible phenomena including profound manifestations such as light, motion and heat. The underlying complexity in solving these time varying and media dependent problems compels engineers to look for tools, facilitating the design and testing of devices and systems, on all scales.

Comprehensive methods are available which facilitate the solutions of Maxwell's equations [1.1] [1.2]. Such methods include the Finite-Difference-Time-Domain method (FDTD) [1.3], the Transmission Line Modelling Method [1.4], and the Finite Element Method (FEM) [1.5].

The work in this thesis is concerned with the application of TLM. TLM has been a well-documented area of study for several decades [1.4, 1.6–1.9], offering a robust simulation technique continually being developed and improved in terms of its numerical efficiency and breadth of application [1.10] [1.11] [1.12].

TLM algorithms exhibit an isomorphism with Maxwells equations and the propagation of voltage signals contained within a network of transmission lines applied to a discretised surface or volume. Ensuring synchronism between electric and magnetic fields between spatial nodes of the mesh is equivalent to providing electrical parameters which provide continuity of voltage and current from one node to another. Each length of electrical transmission line between nodes constitutes a single large equivalent circuit, reducing the solution of Maxwell's equations to solving this transmission line network.

The focus of the early work on TLM was on structured, regular Cartesian meshes in one, two and three dimensions [1.4]. The regularity of these meshes provide straight

forward implementations of the numerical algorithms that govern the simulation, in addition to simple representations of the mesh connectivity in computer memory. However, the limitations of structured meshes become evident when attempting to describe curved geometry. The inevitable stair-casing effect, where the curvature of the geometry boundary cannot be accurately represented using rectangular elements introduces quantisation errors [1.7]. This error can be alleviated through the use of multigridding [1.13], a method which employs meshes of different sampling densities, in addition to developments of rectangular grid methods, which allow a varying degree of mesh density [1.14]. These methods can be viewed as an unstructured approach; nodes are now spaced in an irregular fashion across the problem domain. While they do provide a better approximation of curvature, a high mesh density at the curved boundaries is required to reduce the numerical noise. These boundaries would be better described using unstructured triangular meshes in two dimensional space, and tetrahedral cells in a three dimensional setting.

The use of unstructured triangular and tetrahedral meshes have become highly popular amongst the computational scientific and engineering community [1.15] [1.16]. As the geometric complexity of problem spaces increase, these meshes provide greater flexibility in approximating complex geometry by providing piecewise linear boundary descriptions, with the superior ability to grade the mesh with arbitrary cell sizes. However, the use of triangular or tetrahedral meshes does not come without some drawback. The size and location of the cells that constitute the mesh are no longer regular, in addition to each cell having different simulation parameters. This means explicit pre-processing of the mesh connectivity is now necessary, which necessitates a negative contribution to the overall simulation efficiency.

Computational efficiency is a limitation to all numerical methods, which manifests itself in both runtime and memory consumption. It is therefore important to pay attention to using the available resources in the most effective manner. Meshing

a problem space such that it accurately describes fine features is paramount in achieving accurate simulation results. However, the memory cost of storing the connectivity information of a mesh, in addition to the essential simulation parameters of each mesh element, can become expensive. This becomes especially true when considering very thin flexible devices in the 3D domain, such as thin RFID tags, flexible printed circuit boards and smart clothing.

Motivated by improvements to microscale and nanoscale fabrication techniques, the introduction of these devices have been commercially realised. It is not uncommon to see these devices with a thickness of a millimetre or less [1.17]. In a geometrical sense, these present themselves as curved, open surfaces embedded in a three dimensional domain. To date, these devices are usually simulated using full 3D vectorial computational EM methods. However, using 3D algorithms in order to mesh and simulate electromagnetic behaviour on these surfaces can result in large portions of the runtime and memory being used to simulate areas of the domain that present little electromagnetic activity.

If techniques exist which facilitate a mapping of these surfaces from the 3D domain to an *equivalent* 2D reciprocal flat surface, computational efficiency can be greatly improved. This is because we are now able to exploit the more efficient and simpler 2D algorithms to simulate complex EM behaviour, where meshing the whole 3D domain would otherwise by necessary.

By providing a one-to-one mapping of the geometry from the 3D domain to the 2D flat plane, a 2D unstructured TLM simulation can be applied. Because this mapping is one-to-one, the results of the simulation can then be mapped back to 3D space for visualisation.

The process of mapping surfaces between 3D space and 2D is known as *surface parameterisation*. Surface parameterisation has been used for centuries; a practice

used by cartographers in order to represent our spherical world as a flat map [1.18]. However, the driving force for the development of the first parameterisation methods for discrete surfaces has been the computer graphics industry. Parameterising a 3D model to a flat plane allows textures to be easily applied in 2D before being mapped back to the original surface, enhancing the visual richness of polygonal models [1.19] [1.20]. More recently, the application of mesh parameterisation has been used to facilitate other processes such as remeshing [1.21] [1.22], mesh subdivision [1.23] and CAD repair [1.24] [1.25].

Ideally a mapping should be isometric; that is, preserving both angle and area of the individual faces that constitute a mesh. However this is not possible apart from in special cases. This explains why cartographers have so many different projections of the globe. For example Figure 1.1 shows two conceptual projections of our world. Part a) is an area preserving parameterisation, and b) conceptually shows an angle preserving parameterisation.

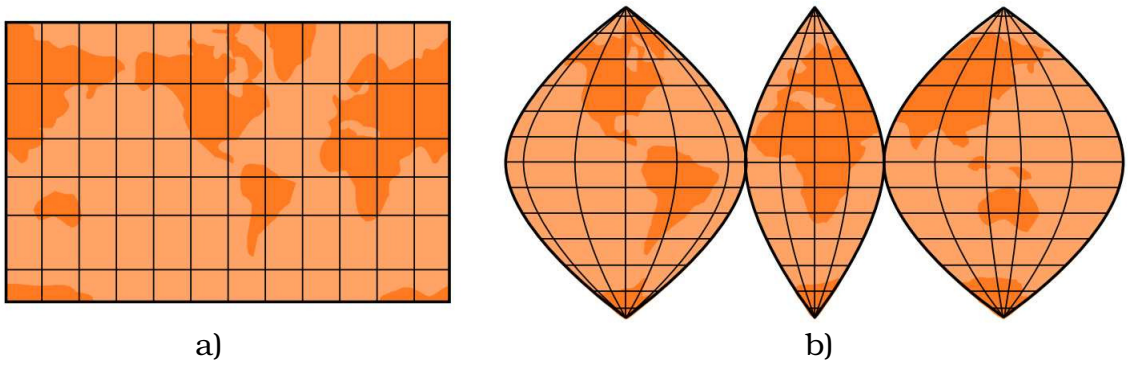

Figure 1.1: Conceptual parameterisations of our world: a) is an area preserving parameterisation, b) is an angle preserving parameterisation

Parameterisations will almost always induce distortion in the form of a change in either angles (known as conformal mappings), or area (authalic mappings), and a good parameterisation for an application is one that minimizes this distortion in some sense. Many different ways have been proposed to do this, all varying only by

the metric of distortion considered and the minimization processes used.

Early techniques of mesh parameterisation follow a simple strategy; define a suitable boundary in the parameter domain, and then proceed to minimise an edge-based energy function to determine the location of the vertices in 2D space. A simple analogy for this energy is to consider a triangular mesh as a physical spring model, where each edge connecting the vertices of the mesh in 3D space is replaced with a spring [1.26]. If the vertices at the boundary are pulled, the springs would relax in a state of minimal energy along the plane, where we assume each spring to be ideal in the sense that the rest length is zero. The new position of the vertices in the now 2D domain are taken to be the parameter points [1.27].

The question that remains is how to choose the spring constants in this spring model. Early techniques define the spring coefficients based on barycentric coordinates [1.28]. Different methods of defining barycentric coordinates, each resulting in a different spring constant, and hence a different parameterisation [1.29] [1.30] [1.31].

The beauty of using any of these choices is that the coefficients, analogous to the spring constant, assigned to each edge depends only on angles and distances between vertices of a mesh. This results in a system that is linear to solve, facilitating a fast and efficient implementation. However a major draw back of this technique is the necessity to define a boundary in 2D parameter space. In order to ensure a valid one-to-one mapping, the boundary is constrained to a regular convex polygon, such as a square or circle. It follows that the extent of the induced distortion depends on how closely the boundary of the geometry in 3D space matches the boundary in the parameter domain. If the 3D surface mesh has a boundary that differs significantly from the specified boundary of the planar domain, these fixed boundary techniques can induce significant stretch at the parameter of the 2D domain [1.32].

In a bid to reduce the distortion at the boundary of the parameter domain, techniques have been derived which integrate the computation of the boundary vertices as part of the solution [1.33] [1.34] [1.35]. While these methods are successful in reducing distortion at the boundary, the minimisation process is no longer linear, and require a greater computational effort. These methods are explored in Chapter 4.

Choosing a parameterisation technique which effectively and efficiently reduces distortion is important. Distortion of angle and area will impact the TLM parameters; affecting the accuracy and stability of the simulation. This forms the the basis of the work carried out in this thesis.

1.2 Organisation of the Thesis

The work presented in this thesis focuses on the unstructured two-dimensional TLM. Chapter 2 provides the background theory to the TLM method. The main concept and methodology of the TLM method is first presented for the one and two dimensional structured implementation for Cartesian meshes. This is then built upon in the form of a derivation of the two dimensional triangular node and its transmission line equivalent for the two dimensional Unstructured TLM. The TLM method is then contrasted with the FEM and FDTD methods.

The use of unstructured meshes allow for superior descriptions of complex geometries compared to their structured kindred, however size and location of the cells that constitute the mesh are no longer regular, in addition to each cell having different simulation parameters. Fundamental to every scientific simulation is how a computer represents and understands the connectivity of a descritised geometry. How a computer understands the connectivity of the mesh, in terms of the adjacency of

mesh elements, has a profound impact on the runtime of the simulation. Expensive searching in order to locate neighbouring nodes in order to traverse the mesh will result in a longer runtime. Counter to this, a full understanding of the mesh connectivity allows the mesh to be traversed with ease, yielding much a faster runtime, but this greater knowledge of the connectivity is also served with a greater impact on the memory consumption. There are existing methods which offer differing paradigms for representing meshes in memory, which allow for a trade off between runtime and memory consumption of a simulation. These are discussed in Chapter 3. Here, an over arching description of the various methods is presented. Following on from the core theory of TLM in chapter 2, how the Unstructured TLM algorithms are represented in data structures is also discussed in Chapter 3, with a validation of these algorithms and the software framework developed for this research, by means of a comparison between the simulation results and analytical solutions of a rectangular resonator.

Chapter 4 begins the discussion of surface parameterisation by providing a background to the methods used to produce a one-to-one mapping of an open surface embedded in the three dimensional domain to a flat two dimensional plane. Achieving an isometric parameterisation, that is, a parameterisation with no distortion is only achievable for special cases. Distortion of area or angle will impact the TLM parameters used for simulation, so a method which minimizes the induced distortion is required. An investigation into how the various parameterisation techniques influence the distortion of a mesh is provided in Chapter 5, by measuring the extent of shear and stretch.

Chapter 6 then provides simulation results for selected test cases, to demonstrate how surface parameterisation can be used with UTLM. Chapter 7 subsequently presents the conclusions of this work, followed by a discussion of the future work that stems from this investigation.

References

[1.1] Joel H. Ferziger, *Numerical Methods for Engineering Applications*, 2nd ed. Wiley, 1998.

[1.2] Matthew N.O. Sadiku, *Numerical Techniques in Electromagnetics*, 2nd ed. CRC Press, 2000.

[1.3] K. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media," *IEEE Trans. Antennas Propag.*, vol. 14, no. 3, pp. 302 – 307, 1966.

[1.4] C. Christopoulos, *The Transmission-Line Modeling Method: TLM.* IEEE Publications,U.S., 1995.

[1.5] A. C. Polycarpou, "Introduction to the Finite Element Method in Electromagnetics," *Synth. Lect. Comput. Electromagn.*, vol. 1, no. 1, pp. 1–126, 2005.

[1.6] P. Johns and R. Beurle, "Numerical Solution of 2-Dimensional Scattering Problems Using a Transmission-Line Matrix," *Proc. Inst. Electr. Eng.*, vol. 118, no. 9, p. 1203, 1971.

[1.7] P. Sewell, J. Wykes, T. Benson, C. Christopoulos, D. Thomas, and A. Vukovic, "Transmission-Line Modeling using Unstructured Triangular Meshes," *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 5, pp. 1490–1497, 2004.

[1.8] P. Sewell, T. M. Benson, C. Christopoulos, D. W. P. Thomas, A. Vukovic, and J. G. Wykes, "Transmission-Line Modeling (TLM) Based upon Unstructured Tetrahedral Meshes," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 6 I, pp. 1919–1928, 2005.

[1.9] C. Christopoulos, *The Transmission-line Modeling (TLM) Method in Electromagnetics*. Morgan & Claypool Publishers, 2006.

[1.10] P. Johns, "Application of the Transmission-Line-Matrix Method to Homogeneous Waveguides of Arbitrary Cross-Section," *Proc. Inst. Electr. Eng.*, vol. 119, no. 8, p. 1086, 1972.

[1.11] M. Ahmadian, "Transmission Line Matrix (TLM) Modelling of Medical Ultrasound," Ph.D. dissertation, University of Edinburgh, 2004.

[1.12] G. Guillaume, P. Aumond, B. Gauvreau, and G. Dutilleux, "Application of the Transmission Line Matrix Method for Outdoor Sound Propagation Modelling - Part 1: Model Presentation and Evaluation," *Appl. Acoust.*, vol. 76, pp. 113–118, 2014.

[1.13] J. Wlodarczyk, "New Multigrid Interface for the TLM Method," *Electron. Lett.*, vol. 32, no. 12, p. 1111, 1996.

[1.14] J. L. Herring and C. Christopoulos, "Solving Electromagnetic Field Problems Using a Multiple Grid Transmission-Line Modeling Method," *IEEE Trans. Antennas Propag.*, vol. 42, no. 12, pp. 1654–1658, 1994.

[1.15] J. Y. Wu and R. Lee, "The Advantages of Triangular and Tetrahedral Edge Elements for Electromagnetic Modeling with the Finite-Element Method," *IEEE Trans. Antennas Propag.*, vol. 45, no. 9, pp. 1431–1437, 1997.

[1.16] A. Bossavit and L. Kettunen, "Yee-Like Schemes On a Tetrahedral Mesh, With Diagonal Lumping," *Int. J. Numer. Model. Electron. Networks Devices Fields*, vol. 12, no. 1/2, pp. 129–142, 1999.

[1.17] Teledyne, *Flexible Circuit Design Guide*, 4th ed. Teledyne Electronic Technologies, 2013.

[1.18] C. Ptolemy, *Geographia*. Editore Alfredo Firmin Didot, 1883.

[1.19] C. Bennis, J.-M. Vézien, and G. Iglésias, "Piecewise Surface Flattening For Non-Distorted Texture Mapping," *ACM SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 237–246, 1991.

[1.20] J. Maillot, H. Yahia, and A. Verroust, "Interactive Texture Mapping," *Proc. 20th Annu. Conf. Comput. Graph. Interact. Tech.*, pp. 27–34, 1993.

[1.21] T. Lee, M. Leok, and N. H. McClamroch, "Geometric Numerical Integration for Complex Dynamics of Tethered Spacecraft," *Proc. 2011 Am. Control Conf.*, no. February, pp. 1885–1891, 2011.

[1.22] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parametrization of Surface Meshes," *Eurographics*, vol. 21, no. 2, 2002.

[1.23] A. Lee, H. Moreton, and H. Hoppe, "Displaced Subdivision Surfaces," *Proc. 27th Annu. Conf. Comput. Graph. Interact. Tech.*, pp. 85–94, 2000.

[1.24] B. Allen, B. Curless, and Z. Popović, "The Space of Human Body Shapes," *ACM Trans. Graph.*, vol. 22, no. 3, p. 587, 2003.

[1.25] V. Kraevoy and A. Sheffer, "Template-Based Mesh Completion," *Eurographics Symp. Geom. Process.*, pp. 13–22, 2005.

[1.26] K. Hormann, "Parameterizing Triangulations," phD, University of Erlangen, 2001.

[1.27] K. Hormann, K. Polthier, and A. Sheffer, "Mesh Parameterization," *ACM SIGGRAPH ASIA 2008 courses - SIGGRAPH Asia '08*, no. August, pp. 1–87, 2008.

[1.28] M. S. Floater, K. Hormann, and G. Kós, "A General Construction of Barycentric Coordinates Over Convex Polygons," *Adv. Comput. Math.*, vol. 24, no. 1-4, pp. 311–331, 2006.

[1.29] W. T. Tutte, "Convex Representations of Graphs," *Proc. London Math. Soc.*, vol. 10, no. February 1959, 1960.

[1.30] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '95*, vol. 6, pp. 173–182, 1995.

[1.31] M. S. Floater, "Parametrization and Smooth Approximation of Surface Triangulations," *Comput. Aided Geom. Des.*, vol. 14, no. 3, pp. 231–250, 1997.

[1.32] A. Sheffer, E. Praun, and K. Rose, "Mesh Parameterization Methods and Their Applications," *Found. Trends Comput. Graph. Vis.*, vol. 2, no. 2, pp. 105–171, 2006.

[1.33] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Trans. Graph.*, vol. 21, no. 3, 2002.

[1.34] K. Hormann and G. Greiner, "MIPS : An Efficient Global Parametrization Method," *Curve Surf. Des.*, pp. 153–162, 2000.

[1.35] A. Sheffer and E. de Sturler, "Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening," *Eng. Comput.*, vol. 17, no. 3, pp. 326–337, 2001.

Chapter 2

Transmission Line Modelling

2.1 Introduction

In order to simulate electromagnetic behaviour a simulation technique is required. The simulation tool of choice for this investigation is the Transmission Line Modelling Method (TLM). In this chapter the background and theory for this simulation method is presented. First, an overview of TLM is discussed, before detailing the concepts of the one dimensional (1D) structured TLM method. This foundation is then developed into the two dimensional (2D) structured TLM method. The core theory of the unstructured 2D TLM method is then presented, leading into a discussion and comparison with the Finite Difference Time Domain (FDTD) Method and the Finite Element Method (FEM).

TLM is a well-established, robust simulation technique continually being developed and improved in terms of its numerical efficiency and breadth of application [2.1] [2.2] [2.3] [2.4]. The TLM algorithms exhibit an isomorphism with Maxwell's equations and the propagation of voltage signals contained within a network of transmission lines applied to a descritised surface or volume. Ensuring synchronism between electric and magnetic fields between spatial nodes of the mesh is equivalent to providing

electrical parameters which induce continuity of voltage and current from one node to another. Each length of transmission line between nodes constitutes an equivalent LC network, reducing the solution of Maxwell's equations to solving this transmission line network. Solving the electromagnetic problem explicitly like this provides a computationally efficient algorithm, where stability can be guaranteed before the runtime of the simulation [2.2].

This thesis focuses on the manipulation of unstructured triangular meshes, and it is the unstructured TLM algorithms that are used to simulate electromagnetic behaviour. However TLM was not originally derived for unstructured descritisation of geometries. Instead TLM began life as an implementation for structured meshes in one, two and three dimensions. The details of these implementations are covered extensively in [2.1] and overviewed here as an aid for better understanding the Unstructured TLM method.

The benefits that Unstructured TLM can offer over its structured kindred are two fold. Firstly, the limitations of structured meshes become evident when used to describe smooth curved geometry. The inevitable boundary stair-casing effect, where the curvature of the boundary cannot be accurately represented using rectangular elements (Figure 2.1 a)), introduces quantization errors [2.2]. The introduction of unstructured mesh elements can help to alleviate this error with the freedom of node placement along the boundary, allowing a better description of the geometry, as shown in Figure 2.1 b).

One could make an argument that simply increasing the density of structured elements along the boundary will allow a more accurate boundary description. As demonstrated in Figure 2.1 c), a multigridded approach can be employed [2.5] whereby meshes of different sampling densities are used. In addition, rectangular grid methods have been derived, which allow a mesh to have a varying degree of node density [2.6]. However, these approaches can require the use of very dense

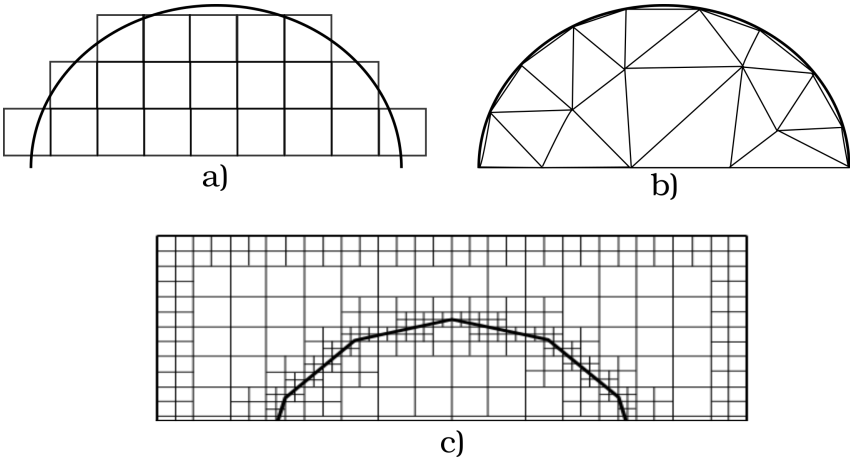

Figure 2.1: Example of boundary approximation using a structured Cartesian approach (a) and unstructured triangular approach (b). Multigridding is also an option (c)

meshes to reduce the amount of nonphysical noise, affecting not only the memory consumption, but the runtime of the simulation [2.3]. As the geometric complexity of problem spaces increases, unstructured meshes provide greater flexibility and efficiency in approximating complex geometry by providing piecewise linear boundary descriptions, improving the accuracy of the simulation [2.2].

In order to understand the Unstructured TLM method [2.2] the following section provides a foundation by first explaining the original 1D and 2D Structured TLM approach, before moving on to the formulations for unstructured triangular meshes.

2.2 1D Structured TLM

The evolution of the TLM simulation in time involves the propagation of voltage impulses along the link lines. When pulses arrive at the node, the received pulses are scattered and reflected back onto the link lines connected to the node. This is shown in Figure 2.2, where V_L^i and V_R^i are pulses incident to the node n from the

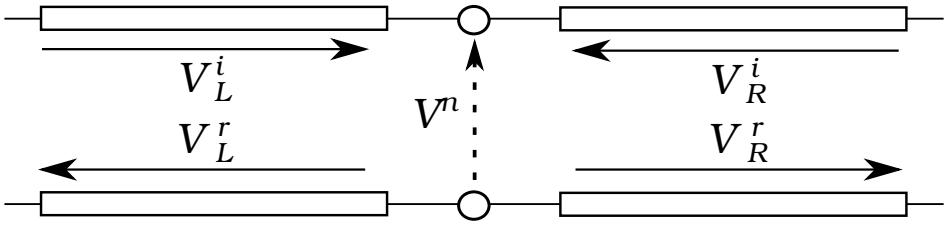

Figure 2.2: The left (L) and right (R) incident (V^i) and reflected (V^r) voltage impulses for a 1D TLM node, where V^n is the nodal voltage.

left and right respectively, V_n is the total voltage and V_L^r and V_R^r are the reflected voltages from the node. Restricting the problem space to one dimension effectively means that the signal is passed along a single line of nodes. A node is the connection point for the network of transmission lines (referred to as link lines) that model the electromagnetic propagation in the simulation. By explicitly setting the capacitance and inductance of the link lines, the material of the simulated medium can be altered. The parameters of free space are modelled by the inductance and capacitance of the transmission line as follows [2.1]:

$$L = \mu_0.\Delta l, \quad C = \varepsilon_0.\Delta l, \tag{2.1}$$

where μ_0 and ε_0 are the permeability and permittivity of free space respectively, L is the inductance and C is the capacitance per length of transmission line Δl. In order to reduce dispersion, Δl is typically modelled to be less than $\frac{\lambda}{10}$, where λ is the wavelength of interest [2.1].

The transmission line is characterised by the characteristic impedance as [2.1]:

$$Z_0 = \sqrt{\frac{L}{C}}. \tag{2.2}$$

The time step, Δt, is the time the voltage takes to propagate through a section of transmission line of length Δl. It is also related to the inductance and capacitance in the following relationship,

$$\Delta t = \frac{\Delta l}{c} = \sqrt{LC}, \tag{2.3}$$

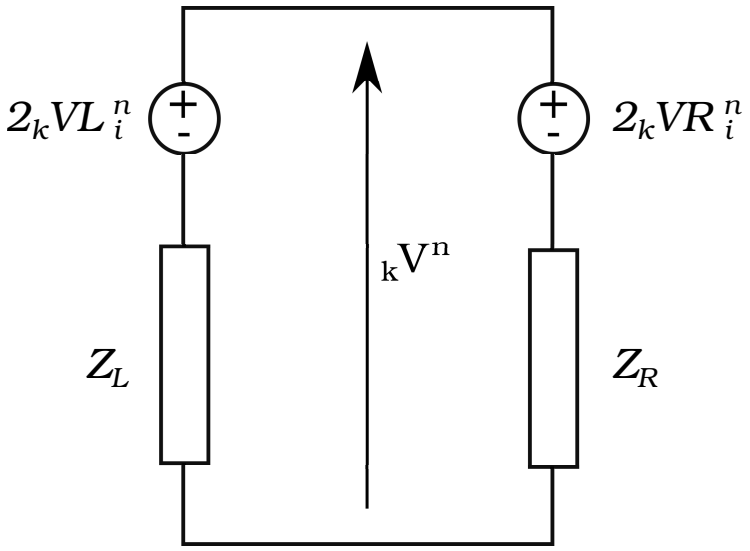

Figure 2.3: 1D Thevenin equivalent circuit at node n and at time k

where c is the velocity of the wave propagation in free space and is given by:

$$c = \frac{1}{\sqrt{\frac{L}{\Delta l}.\frac{C}{\Delta l}}} = \frac{\Delta l}{\sqrt{LC}}. \tag{2.4}$$

The Thevenin equivalent circuit for the one dimensional node, at a time step k is shown in Figure 2.3. This calculates the voltage at the node n due to the pulses incident at the node from the left (VL_i) and right (VR_i) at the time step k. The voltage at the nodes for the one dimensional configuration is [2.1]:

$$_kV^n = \frac{\frac{2_kVL_i^n}{Z_L} + \frac{2_kVR_i^n}{Z_R}}{\frac{1}{Z_L} + \frac{1}{Z_R}}, \tag{2.5}$$

where Z_L and Z_R refer to the impedance of the transmission lines to the left and right respectively, and the subscript k indicates the time step. At the node, incident voltages are scattered, and the resultant reflected voltages are directly calculated

from [2.1]:

$$ {}_kVL_r^n \;=\; {}_kV^n \;-\; {}_kVL_i^n, \quad {}_kVR_r^n =_k V^n \;-\; {}_kVR_i^n. \tag{2.6} $$

This is known as the scatter process which occurs for every node constituting the problem space. The reflected voltage pulses simply become the incident voltage pulses on the neighbouring nodes at the next time step, $k+1$, i.e,

$$ {}_{k+1}VR_i^n \;=\; {}_kVL_r^{n+1}, \qquad\qquad {}_{k+1}VL_i^n \;=\; {}_kVR_r^{n-1}, \tag{2.7} $$

in the process known as the connection process. The scatter and connect processes are repeated a predetermined number of time steps, or until steady state is reached.

Modelling different materials other than air can be achieved by modifying a stub component added to the lossless transmission line connecting the nodes [2.1]. A capacitor open circuit is used to model dielectric permittivity ($\epsilon \neq \epsilon_0$) and inductor short circuit stubs are used to model magnetic permeability ($\mu \neq \mu_0$). For example, the dielectric medium can be modelled by adding an extra capacitance in the TLM model. The characteristic impedance of the open circuit stub that models the capacitance C_s, is given by [2.1]:

$$ Z_c = \frac{\Delta t}{2C_s}, \tag{2.8} $$

where Δt is the time step of the simulation. The effect of the open circuited stub acts to introduce a delay to the propagation speed at the point it is placed, accounting for the reduced velocity of light through a dielectric medium.

2.3 2D Structured TLM

In 2D structured TLM, all of the nodes are arranged in a regular Cartesian grid with identical spacing of the nodes. The benefit of this regular spacing is the simplification of the time step considerations for the simulation. Each link line is the same length and as a consequence the time taken for each voltage pulse to travel between the nodes will be the same through uniform material.

Each mesh element is now constituted by four transmission lines emanating from the node, as shown in Figure 2.4, allowing the description of more complicated propagation problems. This 2D node is known as the *series node*. This models a Transverse Electric configuration, where V_1 and V_3 shown in Figure 2.4 correspond to E_y, and V_2 along with V_4 relate to E_x with the loop current I_z corresponding to H_z. The fact that there are now four ports per mesh cell means that the nodal voltage is updated in a slightly more complicated fashion, however the method is an extension of the principle shown for the one dimensional setup.

To calculate the voltages and currents, an equivalent Thevenin circuit is constructed as shown in Figure 2.5. Each transmission line is replaced by an equivalent source of double the incident voltage ($2_kV_m^i$) where i represents the incident signal, k indicates the current time step, and $m = 1, 2, 3, 4$ is the index of the transmission line. The factor of 2 is applied here because of the round trip of the transmission line.

The loop current is calculated as [2.1],

$$_kI = \frac{2_kV_1^i - 2_kV_2^i - 2_kV_3^i + 2_kV_4^i}{4Z_{TL}}, \tag{2.9}$$

Then the magnetic field is,

$$_kH_z = \frac{I}{\Delta l} = \frac{2_kV_1^i - 2_kV_2^i - 2_kV_3^i + 2_kV_4^i}{4\Delta l Z_{TL}}, \tag{2.10}$$

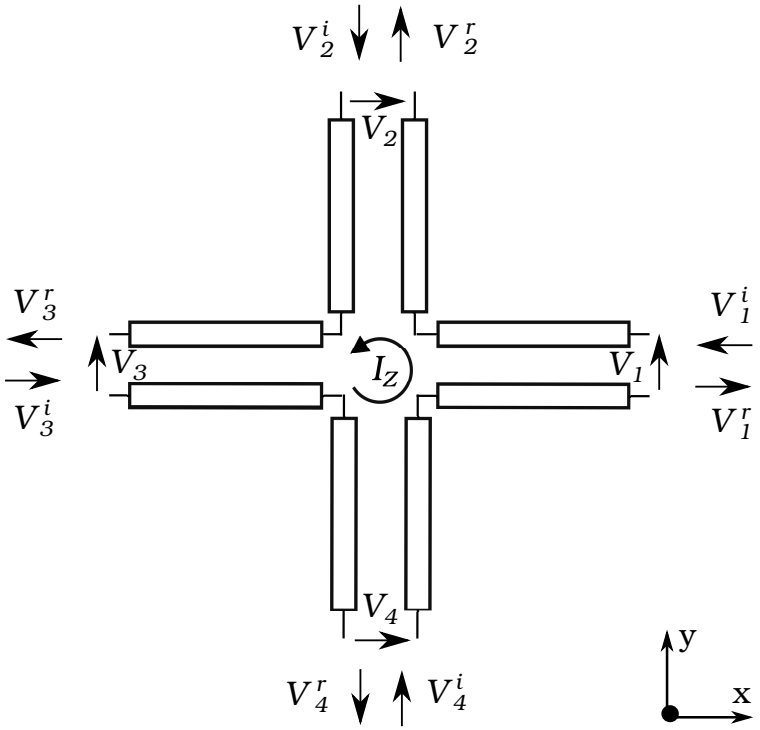

Figure 2.4: 2D series TLM node

and the electric field components are calculated as,

$$ {}_kE_x = -\left(\frac{{}_kV_2^i + {}_k\,V_4^i}{\Delta l}\right), \tag{2.11} $$

$$ {}_kE_y = -\left(\frac{{}_kV_1^i + {}_k\,V_3^i}{\Delta l}\right), \tag{2.12} $$

where Δl is the length of the transmission line.

In the scatter process the voltages at each port are reflected back to the original node. The reflected voltage is related to the incident voltage by,

$$ {}_kV^r \;=\; {}_kV \;-\; {}_kV^i. \tag{2.13} $$

As an example, the reflected voltage at port 1 can be calculated as,

$$ {}_kV_1^r = {}_k\,V - {}_k\,V_1^i = 2{}_kV_1^1 - IZ_{TL} - {}_k\,V_1^i \tag{2.14} $$

$$ {}_kV_r^1 = 0.5({}_kV_1^i + {}_k\,V_2^i + {}_k\,V_3^i - {}_k\,V_4^i). \tag{2.15} $$

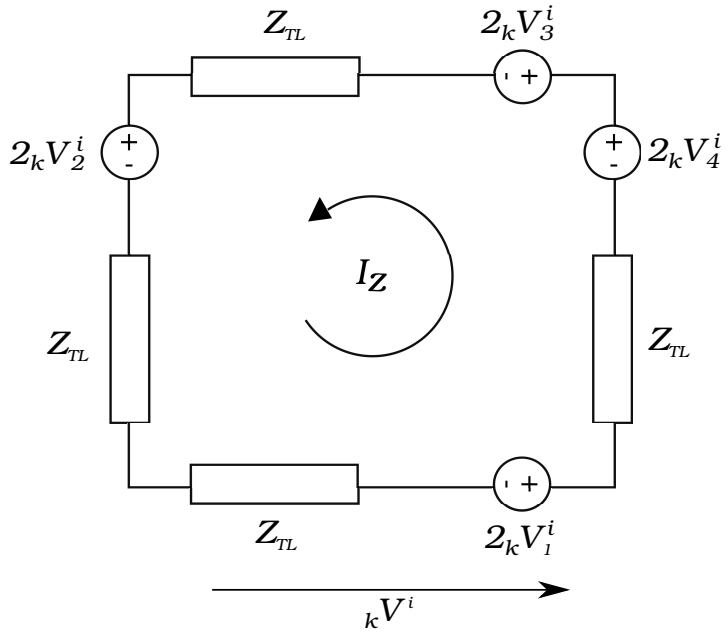

Figure 2.5: Thevinen equivalent circuit for 2D structured series node

The reflected voltages for the four transmission lines can be neatly expressed by the scattering matrix $\boldsymbol{S}$:

$$\boldsymbol{V^r} = \boldsymbol{S}\boldsymbol{V^i}, \tag{2.16}$$

with:

$$\boldsymbol{S} = 0.5 \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix}, \tag{2.17}$$

and where:

$$\boldsymbol{V}^r = [V_1^r, V_2^r, V_3^r, V_4^r]^T, \tag{2.18}$$

represents the reflected voltages from the node, and,

$$\boldsymbol{V}^i = [V_1^i, V_2^i, V_3^i, V_4^i]^T, \tag{2.19}$$

represents the incident voltages at the node.

There is another configuration, known as the *shunt node* [2.1] which models the Transverse Magnetic (TM) field, with field components H_x, H_y and E_z. The shunt node, will allow the simulation of the Transverse Magnetic (TM) field, shown in Figure 2.6. Here V_1 and V_3 relates to H_x, and V_2 and V_4 to H_y and the loop voltage corresponds to E_z. The updating procedures for the shunt node are similar to the series node. The incident signals enter the node to be scattered and the connection phase provide reflected pulses for the next time step.

Figure 2.7 shows the Thevenin equivalent circuit for the shunt node. The voltage V_z across the parallel transmission lines is calculated by:

$$V_z = 0.5(V_1^i + V_2^i + V_3^i + V_4^i); \tag{2.20}$$

and the currents I_x and I_y are calculated as,

$$I_x = \frac{kV_2^i - {}_k V_4^i}{Z_{TL}}, \tag{2.21}$$

$$I_y = \frac{{}_kV_3^i - {}_k V_1^i}{Z_{TL}}. \tag{2.22}$$

In the same manner as for the series node the scattering process provides the reflected pulse with the exception that $\boldsymbol{S}$ is now replaced with:

$$\boldsymbol{S} = 0.5 \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix} \tag{2.23}$$

This means that the main difference between the series and shunt node update is

the scattering matrix which decides which incident voltage is subtracted from the incident nodes to provide the reflected voltage pulses for the next time step.

Altering the material properties of the medium is achieved in much the same way as the 1D structured case, through the addition of capacitive and inductive stubs to model media where $\varepsilon \neq \epsilon_0$ or $\mu \neq \mu_0$.

This description is by no means complete, and the reader is referred to [2.7]. The concepts have been introduced here for structured configurations such that they can be further expanded in the following chapter.

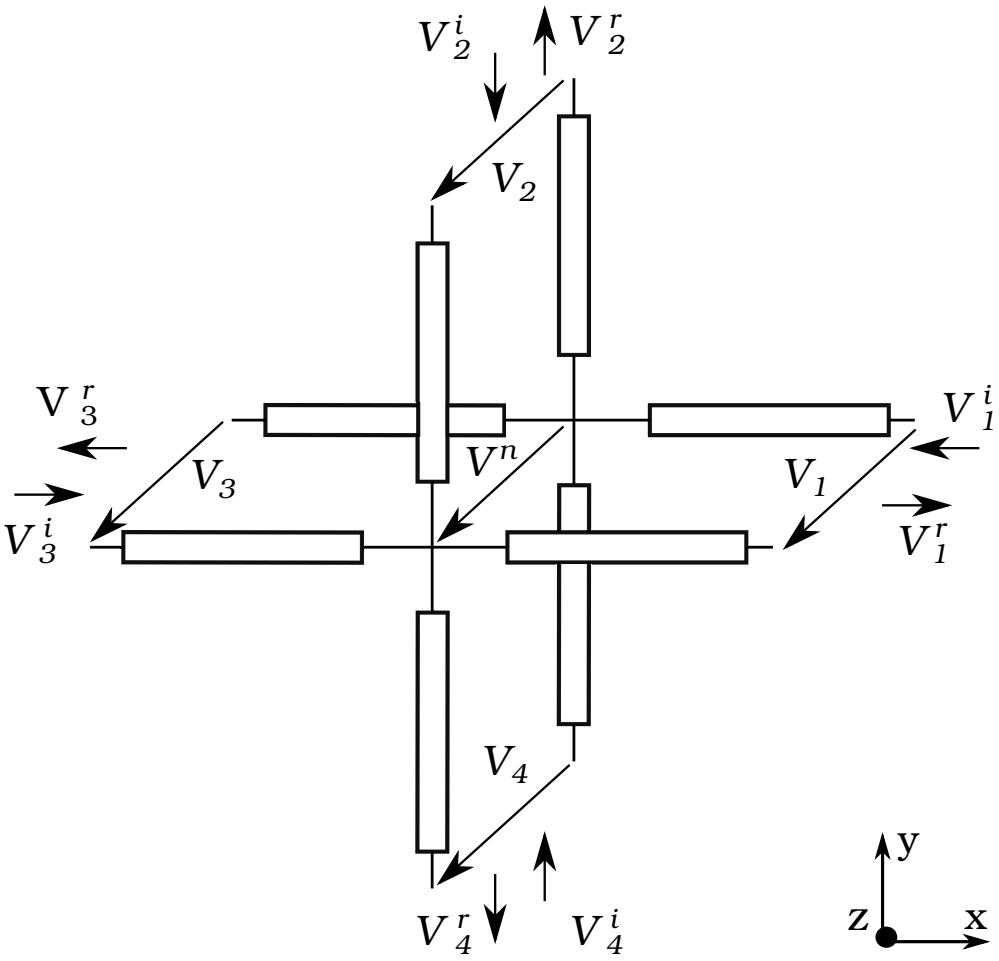

Figure 2.6: 2D shunt TLM node

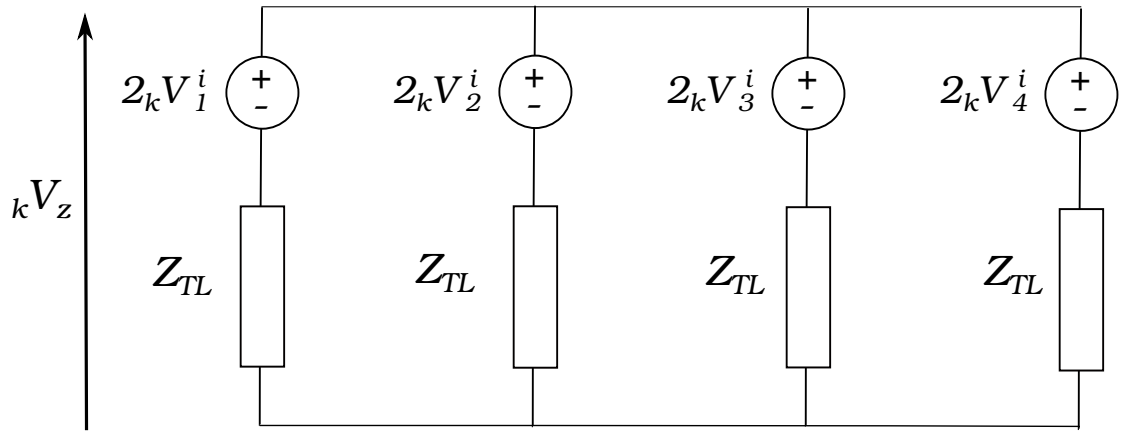

Figure 2.7: Thevinen equivalent circuit for 2D structured shunt node

2.4 2D Unstructured TLM (UTLM)

The challenge imposed on UTLM by triangular meshes is the varying length of the link lines that constitute the TLM network. The link lines are no longer constant length, laying neatly in x and y directions as with the structured case. Instead link lines are now orientated as dictated by the triangles. In addition, the time step of the simulation is derived from the shortest link length across the entire mesh. One poorly conditioned triangle can result in a simulation that requires a number of time steps that is orders of magnitude more than a well conditioned mesh through out. An ill-conditioned mesh refers to triangles that are too long and thin (known as slivers). It is therefore paramount to ensure that these poor quality mesh elements are not part of the descritisation.

Choosing a triangulation that is Delaunay dictates a requirement that no vertex lies within the circumcircle of a triangle within the triangulation. As such, a Delaunay mesh maximises the minimum angles of a triangle in a bid to avoid these slivers. It has been also shown that a Delaunay triangulation minimises the error for field approximation over a domain [2.2]. For Unstructured TLM, the positions of the nodes are dictated by the circumcentres of each triangle constituting a descritisation

which conforms to the Delaunay mesh criteria [2.8].

In order to provide a reference for the terms used in what follows, Figure 2.9 [2.7] shows a diagram of an unstructured TLM node. The dual graph of the Delaunay triangulation, called the Voronoi mesh [2.8] [2.9], ensures the connectivity of adjacent circumcentres of the Delaunay mesh and hence represents the transmission lines of the TLM network. Figure 2.10 shows how the ports relate to the Voronoi dual mesh of the Delaunay triangulation, which governs the connectivity of TLM network. We can see that it is the tessellation of triangles that define the Unstructured TLM network. Each node represented by the triangle circumcentre is connected to a port, which acts as a switching point for the propagation of signals from node to node.

Although a triangulation that conforms to the Delaunay criteria does aid the removal of sliver triangles in the mesh, a Delaunay triangulation is not necessarily the end-all solution to creating a well conditioned mesh. Figure 2.8 shows two Delaunay triangles, an equilateral triangle, with the circumcentre and hence TLM node in the centre, along side a right angle triangle. Both of these triangles conform to the Delaunay criteria; no vertex of the mesh lies within the circumcircle of a triangle. However in the case of the right angle triangle, the circumcentre lies on the hypotenuse, resulting in a zero length link line. A mesh constituting entirely equilateral triangles is the holy grail of meshes for the purposes of the unstructured TLM method, providing a nice equal distribution of link lengths; though this is rarely achievable. Methods have been investigated to relieve a mesh of links approaching zero in length by either combining two triangles to form an element that is a four port node [2.10] or employing mesh relaxation methods [2.11] to maximise the time step of the impending TLM simulation.

In a cylindrical coordinate description of the Transverse Magnetic mode (TM), the only non-zero field components are E_z, H_r and H_θ [2.7]. The node is constructed

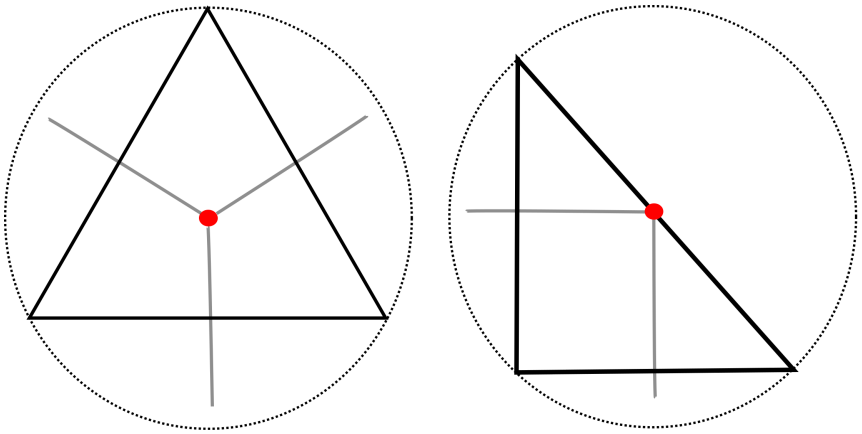

Figure 2.8: Two valid Delaunay triangles (black lines), but an Equilaterial triangle (left) has its circumcentre central to the bounds of the triangle. A right angle triangle (right) has its circumcentre laying on the hypotenuse. The Voronoi dual mesh (in gray) represent the link lines of the transmission line network.

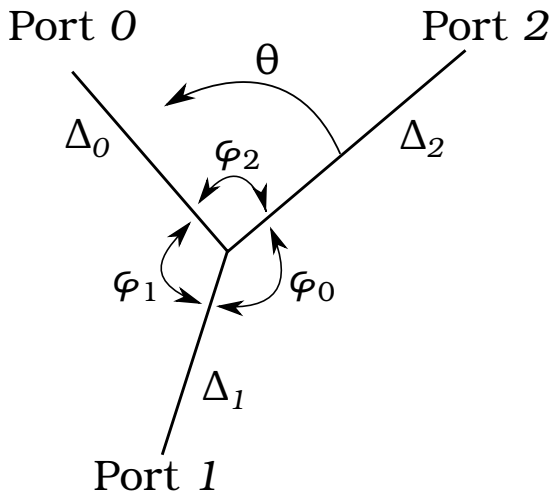

Figure 2.9: Basic Notation and structure of an Unstructured TLM node.

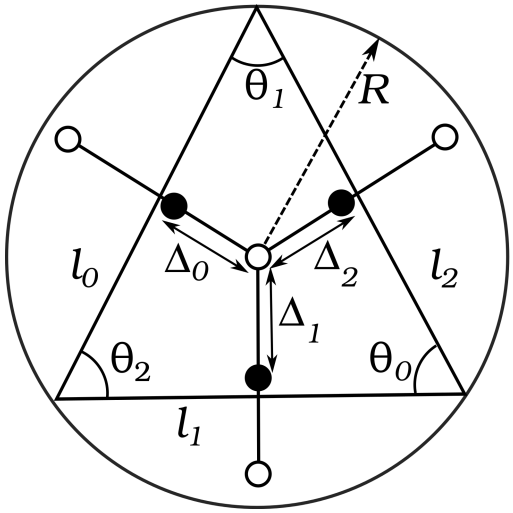

Figure 2.10: A triangle constituting a triangular mesh with embedded terms for the Unstructured TLM node.

by first considering the system in cylinderical coordinates, meaning that the field components to be considered are E_z and H_θ [2.2].

The tangential electric field at the nodes can be expressed in terms of the superposition of the first three cylindrical modes [2.7]:

$$E_z(r,\theta) = X_{c0}J_0(k\Delta) + \frac{2X_{c1}}{k}cos(\theta)J_1(k\Delta) + \frac{2X_{s1}}{k}sin(\theta)J_1(k\Delta) \tag{2.24}$$

where J_i are the Bessel functions of the order i, X are modal amplitudes of the field [2.2] and k is the wave number with θ representing the angle shown in Figure 2.9.

In the same manner we can display the tangential magnetic field at a node [2.7]:

$$-j\omega\mu_0 H_\theta(r,\theta) = \frac{\partial E_z(r,\theta)}{\partial r} \tag{2.25}$$

Using the assumption that the argument of the Bessel functions in (2.24) is much less than 1, $(k\Delta \ll 1)$, the Bessel functions can be approximated as [2.12]:

$$J_0(k\Delta_i) \cong 1 - \frac{(k\Delta_i)^2}{4} \tag{2.26}$$

$$J_1(k\Delta_i) \cong \frac{k\Delta_i}{2}, \tag{2.27}$$

Then equation (2.24) for port 1 can be expressed as:

$$E_{z1} = [1 - \frac{(k\Delta_i)^2}{4}]X_{c0} + \frac{\Delta_1}{2}\frac{2X_{c1}}{k} + (0)X_{s1}, \tag{2.28}$$

where the zero coefficient of the last term is due to $sin(\theta)$ at Port 1 being equal to zero. If the terms of the order $(k\Delta)^2$ are neglected (due again to the small size of $k\Delta$ relative to 1) then all that remains is:

$$E_{z1} = X_{c0} + \Delta_1 X_{c1} \tag{2.29}$$

The same procedure can be followed for the remaining two ports, and expressing them in a matrix $\boldsymbol{T_e}$ relates the electric field to the modal components $\boldsymbol{X}$ [2.7]:

$$\boldsymbol{E_z} = \boldsymbol{T_e}\boldsymbol{X} \tag{2.30}$$

where:

$$\boldsymbol{E_z} = [E_{z1}, E_{z2}, E_{z0}]^T, \quad \boldsymbol{X} = [X_{c1}, X_{c2}, X_{s1}]^T \tag{2.31}$$

$$\boldsymbol{T_e} = \begin{bmatrix} 1 & \Delta_1 & 0 \\ 1 & \Delta_2\cos(\varphi_0) & \Delta_2\sin(\varphi_0) \\ 1 & \Delta_0\cos(\varphi_0+\varphi_1) & \Delta_0\sin(\varphi_0+\varphi_1) \end{bmatrix} \tag{2.32}$$

The magnetic field can be treated in the same way, and also formulated into a matrix relation between $\boldsymbol{H_\theta}$ and $\boldsymbol{X}$. By replacing E_z in equation (2.24) with the magnetic field at the node in equation (2.25) yields:

$$-j\omega\mu_0 H_\theta(r,\theta) = X_{c0}\frac{\partial J_0(k\Delta)}{\partial r} + \frac{2X_{c1}}{k}\cos(\theta)\frac{\partial J_1(k\Delta)}{\partial r}. \tag{2.33}$$

Again, using a small argument approximation the derivatives of the Bessel functions can be approximated as [2.12]

$$\frac{\partial j_0(k\Delta)}{\partial r} \cong -\frac{k^2\Delta}{2} \tag{2.34}$$

$$\frac{\partial j_1(k\Delta)}{\partial r} \cong \frac{1}{2}, \tag{2.35}$$

which allows equation (2.33) to be now expressed, for Port 1, as [2.7]:

$$-j\omega\mu_0 H_{\theta 1} = -\frac{k^2\Delta_1}{2}X_{c0} + \frac{2X_{c1}}{k}\cos(0)\frac{k}{2} + \frac{2X_{s1}}{k}sin(0)\frac{k}{2} \tag{2.36}$$

After evaluating the *sin* and *cos* terms, in addition to the multiplication by Δ_1 (the distance from the triangle circumcentre to Port 1), the following expression is what remains:

$$-j\omega\mu_0\Delta_1 H_{\theta 1} = -\frac{k^2\Delta_1^2}{2}X_{c0} + \Delta_1 X_{c1} \tag{2.37}$$

Again, the same practice can be applied to the remaining components of $\boldsymbol{H_\theta}$ such that a relationship in the form of a matrix $\boldsymbol{T_h}$ between the magnetic field and $\boldsymbol{X}$ from equation (2.30) is expressed as [2.7]

$$j\omega\mu_0\Delta^D\boldsymbol{H_\theta} = \boldsymbol{T_h}\boldsymbol{X} \tag{2.38}$$

where:

$$\boldsymbol{H_\theta} = [H_{\theta 1}, H_{\theta 2}, H_{\theta 0}]^T \tag{2.39}$$

$$\boldsymbol{T_h} = \begin{bmatrix} -\frac{(k\Delta_1)^2}{2} & \Delta_1 & 0 \\ -\frac{(k\Delta_2)^2}{2} & \Delta_2\cos(\varphi_0) & \Delta_2\sin(\varphi_0) \\ -\frac{(k\Delta_0)^2}{2} & \Delta_0\cos(\varphi_0+\varphi_1) & \Delta_0\sin(\varphi_0+\varphi_1) \end{bmatrix} \tag{2.40}$$

$$\Delta^D = \begin{bmatrix} \Delta_1 & 0 & 0 \\ 0 & \Delta_2 & 0 \\ 0 & 0 & \Delta_0 \end{bmatrix} \tag{2.41}$$

Rearranging equation (2.30) such that the modal components can be obtained from the electric field,

$$\boldsymbol{X} = \boldsymbol{T_e^{-1}}\boldsymbol{E_z}, \tag{2.42}$$

and substituting this into equation (2.37), a new relation between $\boldsymbol{H_\theta}$ and $\boldsymbol{E_z}$ is obtained [2.7]:

$$j\omega\mu_0\Delta^D\boldsymbol{H_\theta} = \boldsymbol{T_h}\boldsymbol{T_e^{-1}}\boldsymbol{E_z} \tag{2.43}$$

In order to provide the magnetic and electric field relation at the three ports of the unstructured node, all that is required is to write equation (2.43) out in full, which can be achieved in the following process:

Step one is to multiply the inverse of matrix Δ^D with $\boldsymbol{T_h}$ [2.7]:

$$\Delta^{D-1}\boldsymbol{T_h} = \begin{bmatrix} -\frac{(k\Delta_1)^2}{2} & 1 & 0 \\ -\frac{(k\Delta_2)^2}{2} & \cos(\varphi_0) & \sin(\varphi_0) \\ -\frac{(k\Delta_0)^2}{2} & \cos(\varphi_0+\varphi_1) & \sin(\varphi_0+\varphi_1) \end{bmatrix} \tag{2.44}$$

Inverting the matrix $\boldsymbol{T_e}$ gives [2.7]:

$$\boldsymbol{T_e^{-1}} = \frac{1}{det(\boldsymbol{T_e})} \begin{bmatrix} \Delta_2\Delta_0 s_1 & \Delta_1\Delta_0 s_2 & \Delta_1\Delta_2 s_0 \\ \Delta_2 s_0 - \Delta_0 s_{0+1} & \Delta_0 s_{0+1} & -\Delta_2 s_0 \\ \Delta_0 c_{0+1} - \Delta_2 c_0 & \Delta_1 - \Delta_0 c_{0+1} & \Delta_2 c_0 - \Delta_1 \end{bmatrix} \tag{2.45}$$

where $s_i = sin(\varphi_i)$ and $\cos_i = \cos(\varphi_i)$ and

$$det(\boldsymbol{T_e}) = \Delta_2\Delta_0 s_1 + \Delta_1\Delta_0 s_2 + \Delta_2\Delta_1 s_0 \tag{2.46}$$

Multiplying equations (2.44) and (2.45) together and dividing through by $j\omega\mu_0$ results in:

$$\begin{aligned} \Delta^{D-1}\boldsymbol{T_h}\boldsymbol{T_e^{-1}} = {} & \frac{1}{j\omega\mu_0(\Delta_2\Delta_0 s_1 + \Delta_1\Delta 0 s2 + \Delta_2\Delta_1 s_0)} \\ & \left(-\frac{k^2}{2} \begin{bmatrix} \Delta_2\Delta_0\Delta_1 s_1 & \Delta_1\Delta_0\Delta_1 s_2 & \Delta_2\Delta_1\Delta_1 s_0 \\ \Delta_2\Delta_0\Delta_2 s_1 & \Delta_1\Delta_0\Delta_2 s_2 & \Delta_2\Delta_1\Delta_2 s_0 \\ \Delta_2\Delta_0\Delta_0 s_1 & \Delta_1\Delta_0\Delta_0 s_2 & \Delta_2\Delta_1\Delta_0 s_0 \end{bmatrix} \right. \\ & \left. + \begin{bmatrix} \Delta_2 s_0 + \Delta_0 s_2 & -\Delta_0 s_2 & -\Delta_2 s_0 \\ -\Delta_0 s_1 & \Delta_1 s_0 + \Delta_0 s_1 & -\Delta_1 s_0 \\ -\Delta_2 s_1 & -\Delta_1 s_2 & \Delta_2 s_1 + \Delta_1 s_2 \end{bmatrix} \right) \end{aligned} \tag{2.47}$$

If we take into account that:

$$k^2 = \omega^2\mu\varepsilon, \tag{2.48}$$

equation (2.47) can be tidied up a little to give [2.7]:

$$\Delta^{D-1}\boldsymbol{T_h}\boldsymbol{T_e^{-1}} = j\omega\frac{\varepsilon_0}{2}\frac{\begin{bmatrix}\Delta_2\Delta_0\Delta_1 s_1 & \Delta_1\Delta_0\Delta_1 s_2 & \Delta_2\Delta_1\Delta_1 s_0\\ \Delta_2\Delta_0\Delta_2 s_1 & \Delta_1\Delta_0\Delta_2 s_2 & \Delta_2\Delta_1\Delta_2 s_0\\ \Delta_2\Delta_0\Delta_0 s_1 & \Delta_1\Delta_0\Delta_0 s_2 & \Delta_2\Delta_1\Delta_0 s_0\end{bmatrix}}{\Delta_2\Delta_0 s_1 + \Delta_1\Delta_0 s_2 + \Delta_2\Delta_1 s_0} + \frac{1}{j\omega\mu_0}\frac{\begin{bmatrix}\Delta_2 s_0 + \Delta_0 s_2 & -\Delta_0 s_2 & -\Delta_2 s_0\\ -\Delta_0 s_1 & \Delta_1 s_0 + \Delta_0 s_1 & -\Delta_1 s_0\\ -\Delta_2 s_1 & -\Delta_1 s_2 & \Delta_2 s_1 + \Delta_1 s_2\end{bmatrix}}{\Delta_2\Delta_0 s_1 + \Delta_1\Delta_0 s_2 + \Delta_2\Delta_1 s_0} \tag{2.49}$$

Finally, multiplying by $\boldsymbol{E_z}$ gives [2.7]:

$$\begin{bmatrix}H_{\theta 1}\\ H_{\theta 2}\\ H_{\theta 3}\end{bmatrix} = j\omega\frac{\varepsilon_0}{2}\frac{\begin{bmatrix}\Delta_2\Delta_0\Delta_1 s_1 & \Delta_1\Delta_0\Delta_1 s_2 & \Delta_2\Delta_1\Delta_1 s_0\\ \Delta_2\Delta_0\Delta_2 s_1 & \Delta_1\Delta_0\Delta_2 s_2 & \Delta_2\Delta_1\Delta_2 s_0\\ \Delta_2\Delta_0\Delta_0 s_1 & \Delta_1\Delta_0\Delta_0 s_2 & \Delta_2\Delta_1\Delta_0 s_0\end{bmatrix}}{\Delta_2\Delta_0 s_1 + \Delta_1\Delta_0 s_2 + \Delta_2\Delta_1 s_0}\begin{bmatrix}E_{z1}\\ E_{z2}\\ E_{z0}\end{bmatrix} + \frac{1}{j\omega\mu_0}\frac{\begin{bmatrix}\Delta_2 s_0 + \Delta_0 s_2 & -\Delta_0 s_2 & -\Delta_2 s_0\\ -\Delta_0 s_1 & \Delta_1 s_0 + \Delta_0 s_1 & -\Delta_1 s_0\\ -\Delta_2 s_1 & -\Delta_1 s_2 & \Delta_2 s_1 + \Delta_1 s_2\end{bmatrix}}{\Delta_2\Delta_0 s_1 + \Delta_1\Delta_0 s_2 + \Delta_2\Delta_1 s_0}\begin{bmatrix}E_{z1}\\ E_{z2}\\ E_{z0}\end{bmatrix} \tag{2.50}$$

The above equation shows that the magnetic field is related to the electric field through an admittance matrix. The first term on the RHS of equation (2.50) is capacitive and the second term is inductive. The TLM method facilities the modelling of propagating electromagnetic waves by means of a circuit analogy and therefore the equation needs to be mapped to a circuit such that the voltages and currents provide an equivalence to the electric and magnetic field respectively. The voltage at the ports can be used directly to represent the electric field. The magnetic field however is not continuous at the boundaries of the triangles if directly mapped to the current. A constant (α) is introduced such that the current remains continuous

across the boundaries of triangles. With the introduction of this constant, the magnetic field can be mapped to the current giving the relationship below [2.7]:

$$\begin{bmatrix} E_{z1} \\ E_{z2} \\ E_{z0} \end{bmatrix} \rightarrow \begin{bmatrix} V_1 \\ V_2 \\ V_0 \end{bmatrix}, \begin{bmatrix} \alpha s_1 & 0 & 0 \\ 0 & \alpha s_2 & 0 \\ 0 & 0 & \alpha s_0 \end{bmatrix} \begin{bmatrix} H_{\theta 1} \\ H_{\theta 2} \\ H_{\theta 0} \end{bmatrix} \rightarrow \begin{bmatrix} I_1 \\ I_2 \\ I_0 \end{bmatrix} \tag{2.51}$$

mapping (2.51) into (2.50) leads to [2.7]:

$$\begin{aligned} \begin{bmatrix} I_1 \\ I_2 \\ I_0 \end{bmatrix} &= j\omega\alpha\frac{\varepsilon_0}{2}\frac{\begin{bmatrix} \Delta_2\Delta_0\Delta_1 s_1 s_1 & \Delta_1\Delta_0\Delta_1 s_2 s_1 & \Delta_2\Delta_1\Delta_1 s_0 s_1 \\ \Delta_2\Delta_0\Delta_2 s_1 s_2 & \Delta_1\Delta_0\Delta_2 s_2 s_2 & \Delta_2\Delta_1\Delta_2 s_0 s_2 \\ \Delta_2\Delta_0\Delta_0\Delta_1 s_1 s_0 & \Delta_1\Delta_0\Delta_0 s_2 s_0 & \Delta_2\Delta_1\Delta_0 s_0 s_0 \end{bmatrix}}{\Delta_2\Delta_0 s_1 + \Delta_1\Delta_0 s_2 + \Delta_2\Delta_1 s_0} \\ &+ \frac{\alpha}{j\omega\mu_0}\frac{\begin{bmatrix} \Delta_2 s_0 s_1 + \Delta_0 s_2 s_1 & -\Delta_0 s_2 s_1 & -\Delta_2 s_0 s_1 \\ -\Delta_0 s_1 s_2 & \Delta_1 s_0 s_2 + \Delta_0 s_1 s_2 & -\Delta_1 s_0 s_2 \\ -\Delta_2 s_1 s_0 & -\Delta_1 s_2 s_0 & \Delta_2 s_1 s_0 + \Delta_1 s_2 s_0 \end{bmatrix}}{\Delta_2\Delta_0 s_1 + \Delta_1\Delta_0 s_2 + \Delta_2\Delta_1 s_0} \begin{bmatrix} V_1 \\ V_2 \\ V_0 \end{bmatrix} \end{aligned} \tag{2.52}$$

A circuit topology with three ports with the current and voltage relationship shown in equation (2.52) would give the desired model for an unstructured node. However we are presented with a caveat. Circuits consisting of capacitors and inductors are reciprocal. That is, the admittance matrix must be symmetrical. This symmetry is necessary to ensure the current entering a port from a node is equal to the current travelling to the adjacent node.

In order to do this, the matrices in the expression must be symmetrical, such that an accurate mapping to a possible circuit can be achieved [2.13]. This symmetry guarantees that the input current to a port matches the output current to the adjacent port. The inductive term in the expression meets the condition of reciprocity;

that is, the matrix is equivalent to its transpose. However the capacitive term does not. If the voltages at the port are equal ($V_1 = V_2 = V_0$) the inductive term would disappear, as all components are equal to zero, leaving only the capacitive term, which is not symmetric and therefore not suitable for a circuit representation. Under this condition, the capacitive term can be reformulated, by removing factors of each row of $\Delta_i s_i$ to leave terms that match the denominator of the first matrix term, which can then be removed. We can then replace the first matrix expression with [2.7]:

$$j\omega\alpha\frac{\varepsilon_0}{2}\begin{bmatrix}\Delta_1 s_1 & 0 & 0\\ 0 & \Delta_2 s_2 & 0\\ 0 & 0 & \Delta_0 s_0\end{bmatrix}\begin{bmatrix}V_1\\ V_2\\ V_0\end{bmatrix} \tag{2.53}$$

Then the equation (2.52) can be expressed as:

$$\begin{bmatrix}I_1\\ I_2\\ I_0\end{bmatrix} = j\omega\alpha\frac{\varepsilon_0}{2}\begin{bmatrix}\Delta_1 s_1 & 0 & 0\\ 0 & \Delta_2 s_2 & 0\\ 0 & 0 & \Delta_0 s_0\end{bmatrix}\begin{bmatrix}V_1\\ V_2\\ V_0\end{bmatrix} + \frac{\alpha}{j\omega\mu_0}\frac{\begin{bmatrix}\Delta_2 s_0 s_1 + \Delta_0 s_2 s_1 & -\Delta_0 s_2 s_1 & -\Delta_2 s_0 s_1\\ -\Delta_0 s_1 s_2 & \Delta_1 s_0 s_2 + \Delta_0 s_1 s_2 & -\Delta_1 s_0 s_2\\ -\Delta_2 s_1 s_0 & -\Delta_1 s_2 s_0 & \Delta_2 s_1 s_0 + \Delta_1 s_2 s_0\end{bmatrix}}{\Delta_2\Delta_0 s_1 + \Delta_1\Delta_0 s_2 + \Delta_2\Delta_1 s_0}\begin{bmatrix}V_1\\ V_2\\ V_0\end{bmatrix} \tag{2.54}$$

where the first term is symmetric as desired. Figure 2.11 demonstrates the circuit representation of this network equation, noting the circuit parameters are given by [2.7]:

$$L_i = \frac{\mu_0\Delta_i}{\alpha s_i}, \quad C_i = \frac{\varepsilon_0\Delta_i\alpha s_i}{2}, \tag{2.55}$$

where ε_0 and μ_0 are the free space permittivity and permeability respectively, Δ_i, is the link line length, $s_i = sin(\varphi_i)$ and α is the mapping constant.

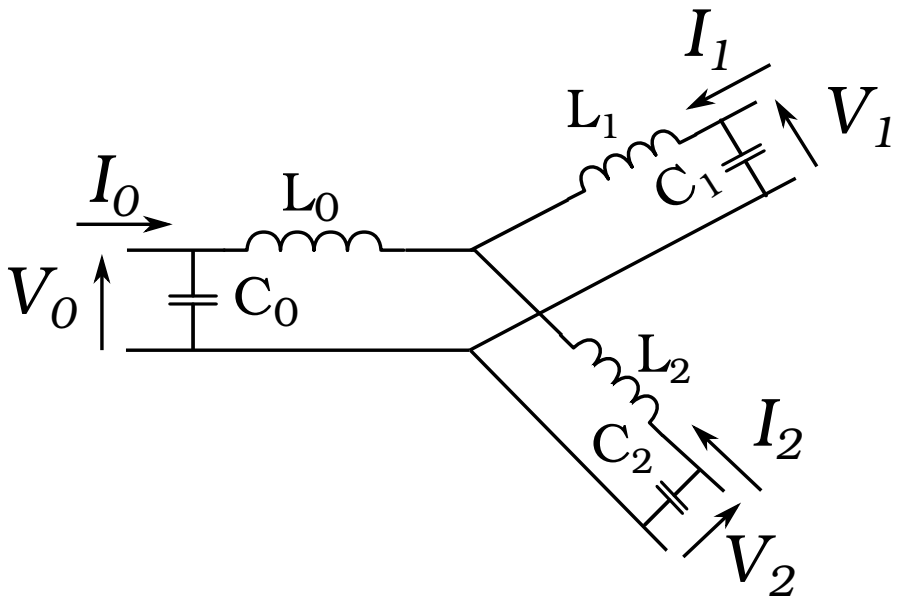

Figure 2.11: Circuit implementation of the network equation in equation (2.54)

The remaining criteria left to solve is the continuity of fields. The electric field can be directly mapped as the voltage, but a constant was introduced in (2.51) to ensure the continuity of the magnetic field across triangle boundaries, which now needs to be defined. Looking back at Figure 2.10, we can observe that [2.7]:

$$\alpha = 2Rs_i = l_i \tag{2.56}$$

where R is the circumradius, and l_i is a boundary edge of the triangle. Choosing α to be l_i for each individual node gives:

$$I_i = 2Rs_iH = l_iH_i. \tag{2.57}$$

The length l_i is the same for two adjacent triangles sharing the same edge and by setting the constant α to this edge length for which the representative link line intersects, the continuity of the magnetic field from node to node can be ensured.

The transmission line circuit facilitates the formulation of a time-domain algorithm. Figure 2.12 depicts the transmission line equivalent of the circuit representation shown in Figure 2.11, using inductive link lines and open circuit capacitive stubs

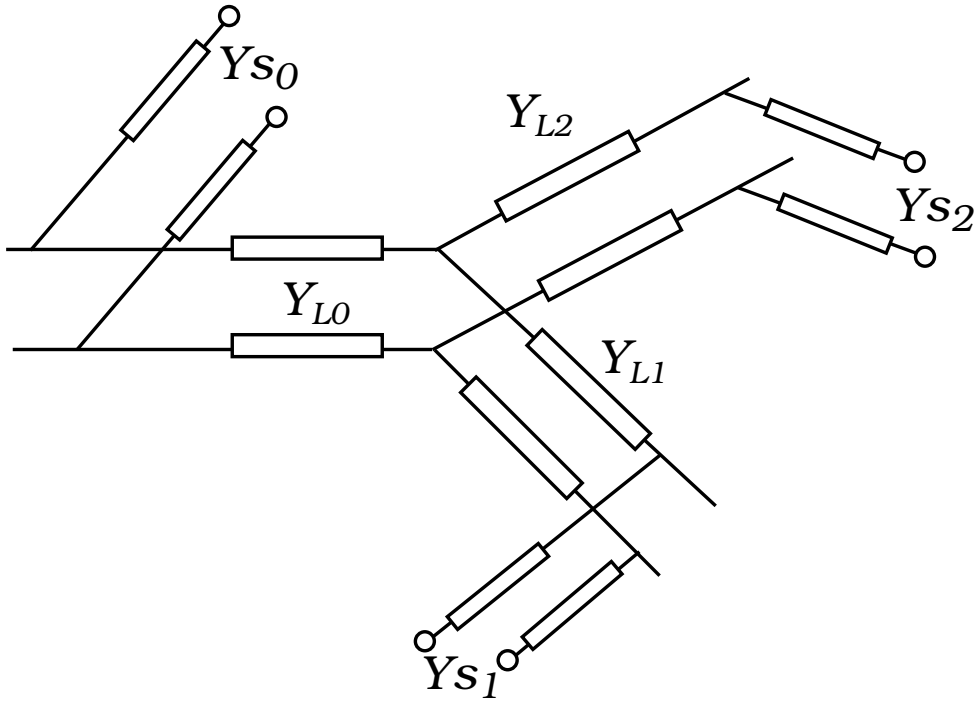

Figure 2.12: Transmission line equivalent of the circuit in Figure 2.11

[2.7]. A signal travelling between two adjacent nodes takes time Δt. It therefore takes the signal $\frac{\Delta t}{2}$ to travel from the node centre to a port. The time step of the simulation is calculated based on the shortest transmission line within the TLM network and is calculated as [2.7]:

$$\Delta t = \Delta_{min}\sqrt{2\mu\varepsilon}. \tag{2.58}$$

where Δ_{min} is the shortest link length, ε is the material permittivity and μ is the material permeability. Capacitive stubs facilitate the synchronisation of the voltage as it propagates along transmission lines of nonuniform length, such that they reach the ports at the same time. The stubs are inserted to slow down the pulse in all lines that are greater than the minimum link length. The propagation velocity along a link line is defined as [2.1]:

$$u_i = \frac{\Delta l}{\Delta t/2} = \frac{1}{\sqrt{L_i C_i}}. \tag{2.59}$$

It follows then that the capacitance is:

$$C_i = \frac{1}{u_i^2 L_i} = \frac{(\Delta t^2)}{4(\Delta l)L_i} \tag{2.60}$$

The characteristic impedance of the link line is:

$$\sqrt{\frac{L_i}{C_i}} = \sqrt{\frac{4(\Delta l)^2 L_i^2}{\Delta t^2}} = \frac{2L_i \Delta l}{\Delta t} = \frac{2L_i}{\Delta t}. \tag{2.61}$$

Using equation (2.55) the characteristic impedance of the transmission line can then be expressed as,

$$Z_{Li} = \frac{2L}{\Delta t} = \frac{2\mu\Delta_i}{2Rsin\varphi_i \Delta t} = \frac{2\mu\Delta_i}{l_i \Delta t} \tag{2.62}$$

where μ is the material permeability ($\mu = \mu_0\mu_r$), Δ_i is the length of the link line ($i = 0, 1, 2$), l_i is the corresponding triangle edge length of the port, and the time to travel between nodes is Δt. Figure 2.12 shows the transmission line equivalent circuit representation of the circuit implementation in Figure 2.11. It follows that the admittance of the link lines in Figure 2.12 is:

$$Y_{Li} = \frac{l_i \Delta t}{2\mu\Delta_i}. \tag{2.63}$$

The admittance of the open circuit capacitive stub is calculated using a similar process. From equation (2.59) the inductance is

$$Ł_i = \frac{1}{u_i^2 C_i} = \frac{(\Delta t)^2}{4C_i(\Delta l)^2}. \tag{2.64}$$

and the characteristic impedance of the stub is:

$$Z_{si} = \sqrt{\frac{L_i}{C_i}} = \sqrt{\frac{\Delta t^2}{4(\Delta l)^2 C_i^2}} = \frac{\Delta t}{2C_i \Delta l} = \frac{\Delta t}{2C_i}. \tag{2.65}$$

Now, using the capacitance from equation (2.55) the stub admittance Y_{si} in Figure 2.12 is calculated as:

$$Y_{si} = \frac{2C}{\Delta t} - Y_{Li} = \frac{2(2R\sin\varphi_i \varepsilon \Delta_i)}{2\Delta t} - \frac{l_i \Delta t}{2\mu\Delta_i} \tag{2.66}$$

and therefore:

$$Y_{si} = \frac{l_i \varepsilon \Delta_i}{2\Delta t} - \frac{l_i \Delta t}{2\mu \Delta_i} \tag{2.67}$$

where ε is the material permitivity ($\varepsilon = \varepsilon_0 \varepsilon_r$), Δ_i is the length of the link line and l_i is the triangle edge length for the corresponding port, and time step Δt.

It follows that modelling different materials involves altering ε and μ in equations (2.63) and (2.67) to provide the desired admittances.

The scatter and connect processes are conducted using exactly the same methodology as the 2D sturctured shunt node (2.16):

$$\begin{bmatrix} V_1^r \\ V_2^r \\ V_0^r \end{bmatrix} = \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} V_1^i \\ V_2^i \\ V_0^i \end{bmatrix} \tag{2.68}$$

The scatter process occurs at the node, where incident voltages are reflected, and subsequently these reflected voltages become the incident voltage pulses on the neighbouring nodes at the next time step during the connection phase. This process is then repeated until the desired simulated time is reached.

This chapter has provided an overview of the core TLM theory. How these algorithms are applied to the unstructured mesh for a full TLM simulation is discussed in the following chapter, on data structures.

2.5 Alternative Simulation Methods To TLM

Outside of the realm of Transmission Line Modelling for the purposes of electromagnetic simulations, there exists other very well established simulation methods.

Two alternatives to TLM are the Finite-Difference Time-Domain (FDTD) method, and the Finite Element Method (FEM). The focus of FEM is providing a simulation platform for the frequency domain, where as FDTD and TLM are primarily focused on simulations in the time domain.

This section provides a brief comparison between these popular simulation methods, such that the reader has an impression of the main differences between other widely used techniques and the TLM methodology.

2.5.1 Finite-Difference Time Domain (FDTD) Method

The FDTD method is a long standing simulation technique for electromagnetics. Initially developed by Kane Yee [2.14] in the 1960s, the method provides the user with a direct solution to Maxwell's equations, which was originally implemented for structured Cartesian descritisations of the problem space, however further FDTD implementations are now available allowing simulations applied to irregular, Cartesian meshes [2.15].

Central to the FDTD method is the cuboidal Yee cell, or Yee lattice [2.14](which describes the relative position the field components hold in relation to the cuboidal centre. As shown in Figure 2.13 the electric field components are located on the edges of the, cuboid whereas the magnetic field components are normal to the faces of the cuboid.

Like TLM, the evolution of the algorithms are carried out in an iterative fashion, however, where TLM updates the electric and magnetic field each full time step, updates to the field components in FDTD are required to be alternately updated at a half time step [2.16]. The magnetic field is updated at the mid-point of the time step interval, with electric field updating at full time step intervals. It follows

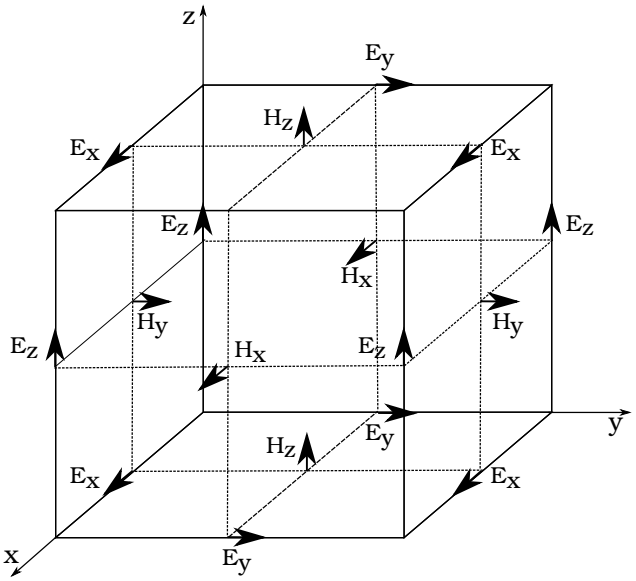

Figure 2.13: The Yee Cell, demonstrating the separation of the components of the electric and magnetic field. The components are updated at half time step intervals with electric and magnetic fields being updated alternatively.

that in order to maintain stability of the simulation, the following condition must be met [2.17]:

$$v_{max}\Delta t \leq \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)^{-\frac{1}{2}}, \tag{2.69}$$

where v_{max} is the maximum velocity of light through the medium of the problem space, and Δx, Δy, Δz are the node separation distances in the three spatial dimensions, which also relate to the dimensions of the Yee cell in Figure 2.13.

Like TLM, FDTD in its structured form is prone to the stair-casing effect at the boundary when attempting to describe curved geometry. In a bid to relieve this problem, unstructured triangular methods have been more recently developed, which allow more versatile and accurate descriptions of the problem space [2.18] [2.19].

2.5.2 Finite Element Method (FEM)

FEM like the FDTD method began life in the 1960s. Not only is this a powerful tool used for solving a diverse range of problems in computational electromagnetics, such as waveguides and microstrip problems [2.20] [2.21] [2.22], but it has proven its worth across number of other scientific and engineering disciplines. Structural and stress analysis [2.23] and fluid dynamics [2.24] are just a tiny proportion of the applications of the FEM method. The Finite Element Method really refers to analysing a problem space through the finite descritisation of the geometry and the solution to partial differential equations [2.25]. It will also be shown in Chapter 4, a Finite Element approach exists for mapping a surface in the three dimensional domain to a two dimensional flat plane.

FEM, like FDTD and TLM involve descritising the problem space into elements, to which an evolution of an algorithm is applied [2.26]. FEM was originally formulated as an unstructured algorithm, always allowing for the accurate representation of complex geometry.

The Finite Element Method uses the concept of basis functions focused upon certain elements of the mesh. These elements could relate to the vertices constituting the mesh, various points along the edge of a mesh triangle or combinations of the two. The basis function is an approximation based upon polynomials of the function values within individual triangles of the mesh. In this way the triangles can, based upon the values defined at the triangle edges or vertices, derive values that would be present inside the triangle. These functions could be piecewise linear, or, for modelling higher order partial differential equations, polynomial piecewise linear basis functions are used.

Once the basis functions are determined then a system of equations are formulated for each individual mesh element. The simulation can then be solved iteratively in

time, in the same manner as the FDTD method and TLM. FEM is primarily solved in the frequency domain, however algorithms do exist for time domain solutions using FEM [2.27]. It should be pointed out that frequency domain methodologies have also been developed for FDTD [2.28] as well as TLM [2.29].

FDTD and TLM always use the same method to calculate the field values at each time step. FEM however requires the solution to a sparse matrix, which itself determines the method used to solve it. Each local basis function per mesh element will contribute to the global solution to the problem.

2.5.3 Performance Differences Between TLM, FDTD and FEM

The methods in this section have all been independently validated to be adequate tools for approximating the solution to electromagnetic problems with good degrees of accuracy. In order to contrast the methods against one another however, there needs to be a measure of comparison. A measure that is regularly used for time domain simulations is numerical dispersion. This is the approximation error obtained when simulating electromagnetics using descritised space [2.30], which is due to the inability of a simulation to represent the field variance exactly at each mesh element.

Comparisons between FDTD and TLM have been previously investigated in the literature. A comprehensive study [2.31] analytically and theoretically demonstrated that dispersion between TLM and FDTD are akin. A study into the effects of dispersion using rectangular wave guides was performed in [2.30] where it was shown that TLM demonstrated significantly less numerical dispersion than the basic FDTD method. When the experiment was repeated using a higher order FDTD implementation however, the results demonstrated a good comparison with TLM. Further

examples in the literature also pointed out a fine congruence between TLM and FDTD [2.32] [2.33].

The fact FEM predominantly functions in the frequency domain for electromagnetic simulations makes the comparisons somewhat difficult when contrasting with time domain techniques. However the comparison exists when selecting the correct methodology for the problem at hand. The authors of [2.34] concluded that a choice between TLM and FEM is problem dependent. Where a wide band of frequencies is of interest, a time domain TLM simulation with subsequent frequency analysis via Fourier Transform obtained results over a broad frequency band with a saving in computational time compared with FEM. In contrast, a direct frequency domain simulation in FEM demonstrated an advantage when observing a reduced number of frequencies.

Until the advent of unstructured TLM, a difference between TLM and FEM was how the problem domain was descritised. TLM required the domain to be divided into structured Cartesian cells, where FEM always had the freedom to descritise the environment in an unstructured simplicial fashion. This has of course now changed with TLM providing a means to be applied to triangular and tetrahedral descritisation of a problem space.

2.6 Summary

This chapter introduced the TLM method as a computational electromagnetics simulation tool, first through the 1D and 2D structured approach for Cartesian meshes, building to the 2D unstructured TLM algorithm. The description of the unstructured approached covered the isomorphism of Maxwell's Equations with a capacitive and inductive circuit such that the propagation of electric and magnetic field can be

analysed using voltage and hence current traversing a network of transmission lines in two dimensions. It has been shown that the voltage pulses are updated in time through scatter and connection processes at each node that constitutes a mesh and at each time step. The model as then expanded to allow for the consideration of different materials through the application of a stub element.

Considerations of the mesh quality were also provided, where it has been decided that a Delaunay triangulation for the unstructured case was a best fit; this maximises the minimum angle of a triangle, preventing sliver" elements. This triangulation method does not provide a perfect solution however, where the link lines of the TLM network can still approach zero with certain triangle shapes that still conform to the Delaunay criteria. This mainly is attributed to right angle triangles. The maximum node separation that can be achieved is through the use of a uniform equilateral triangular mesh, however this is rarely obtained.

Finally, a brief description and comparison between TLM and other popular simulation methodologies were provided. Results from the literture demonstrate that TLM is comparable to FDTD as an alternative time domain simulation tool.

The representation of unstructured meshes which UTLM is applied to by a computer has profound consequences on the runtime and memory consumption of simulation. The different methods of representing unstructured meshes in computer memory is discussed in the following chapter. Additionally, how the core theory of UTLM presented in this chapter can be applied algorithmically to a mesh for a simulation is overviewed.

References

[2.1] C. Christopoulos, *The Transmission-Line Modeling Method: TLM.* IEEE Publications,U.S., 1995.

[2.2] P. Sewell, J. Wykes, T. Benson, C. Christopoulos, D. Thomas, and A. Vukovic, "Transmission-Line Modeling using Unstructured Triangular Meshes," *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 5, pp. 1490–1497, 2004.

[2.3] P. Sewell, T. M. Benson, C. Christopoulos, D. W. P. Thomas, A. Vukovic, and J. G. Wykes, "Transmission-Line Modeling (TLM) Based upon Unstructured Tetrahedral Meshes," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 6 I, pp. 1919–1928, 2005.

[2.4] M. Ahmadian, "Transmission Line Matrix (TLM) Modelling of Medical Ultrasound," Ph.D. dissertation, University of Edinburgh, 2004.

[2.5] J. L. Herring and C. Christopoulos, "The Use of Graded and Multigrid Techniques in Transmission Line Modelling," *Second Int. Conf. Comput. Electromagn.*, pp. 142–145, 1994.

[2.6] ——, "Solving Electromagnetic Field Problems Using a Multiple Grid Transmission-Line Modeling Method," *IEEE Trans. Antennas Propag.*, vol. 42, no. 12, pp. 1654–1658, 1994.

[2.7] C. Christopoulos, *The Transmission-line Modeling (TLM) Method in Electromagnetics.* Morgan & Claypool Publishers, 2006.

[2.8] J. R. Shewchuk, "Reprint of: Delaunay Refinement Algorithms for Triangular Mesh Generation," *Comput. Geom. Theory Appl.*, vol. 22, pp. 21–74, 2014.

[2.9] L. Guibas and J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi," pp. 74–123, 1985.

[2.10] L. Khashan, A. Vukovic, P. Sewell, and T. M. Benson, "Assessment of Accuracy and Runtime Trade-offs in Unstructured TLM Meshes for Electromagnetic Simulations," *Antennas Propag.*, no. November, pp. 518–523, 2013.

[2.11] P. Sewell, T. M. Benson, A. Vukovic, and S. Cole, "Mesh Optimisation Methods for Unstructured Transmission-Line Modelling," *IET Sci. Meas. Technol.*, vol. 7, no. 1, pp. 32–40, 2013.

[2.12] D. Pozar, *Microwave Engineering*. John Wiley & Sons, 1998.

[2.13] Chirlian P. M, *Basic Network Theory*, 1st ed. McGraw-Hill, 1969.

[2.14] K. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media," *IEEE Trans. Antennas Propag.*, vol. 14, no. 3, pp. 302 – 307, 1966.

[2.15] M. Okoniewski, E. Okoniewska, and M. A. Stuchly, "Three-Dimensional Subgridding Algorithm for FDTD," *IEEE Trans. Antennas Propag.*, vol. 45, no. 3, pp. 422–429, 1997.

[2.16] A. Taflove, *Computational Electrodynamics, the Finite-Difference Time-Domain Method*. Artech House, 1995.

[2.17] I. Transactions and O. N. Microwave, "Three-Dimensional Finite-Difference the Analysis of Microwave-Device," *IEEE Trans. Microw. Theory Tech.*, vol. 35, no. 8, pp. 688–696, 1987.

[2.18] A. Bossavit and L. Kettunen, "Yee-Like Schemes On a Tetrahedral Mesh, With Diagonal Lumping," *Int. J. Numer. Model. Electron. Networks Devices*

Fields, vol. 12, no. 1/2, pp. 129–142, 1999.

[2.19] R. B. Wu, "Hybrid Finite-Difference Time-Domain Modeling of Curved Surfaces Using Tetrahedral Edge Elements," *IEEE Trans. Antennas Propag.*, vol. 45, no. 8, pp. 1302–1309, 1997.

[2.20] Y. Lu and F. Fernandez, "An Efficient Finite Element Solution of Inhomogeneous Anisotropic and Lossy Waveguides," *IEEE Trans. Microw. Theory Tech.*, vol. 41, no. 6, pp. 1215–1223, 1993.

[2.21] H. N. Chait, F. Sandy, and J. Saunders, "Numerical Solution of Dielectric Loaded Waveguides: I-Finite-Element Analysis," *IEEE Trans. Microw. Theory Tech.*, vol. MTT-18, no. 12, pp. 1124–1131, 1970.

[2.22] M. Hano, "Finite-Element Analysis of Dielectric-Loaded Waveguides," *IEEE Trans. Microw. Theory Tech.*, vol. MTT-32, no. 10, pp. 1275–1279, 1984.

[2.23] M. Mahendran, "Applications of Finite Element Analysis in Structural Engineering," in *Int. Conf. Comput. Aided Eng.*, Chennia, India, 2007, pp. 38–46.

[2.24] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method: Solid Mechanics*, 6th ed. Elsevier Butterworth Heinemann, 2000.

[2.25] A. C. Polycarpou, "Introduction to the Finite Element Method in Electromagnetics," *Synth. Lect. Comput. Electromagn.*, vol. 1, no. 1, pp. 1–126, 2005.

[2.26] J. Oden, *Historical Comments on Finite Elements.* ACM, 1990.

[2.27] D. Dibben and N. Metaxas, "Frequency Domain vs. Time Domain Finite Element Methods for Calculation of Fields in Multimode Cavities," *IEEE Trans. Magn.*, vol. 33, no. 2, pp. 1468–1471, 1997.

[2.28] T. K. Sarkar, M. Manela, V. Narayanan, and A. R. Djordjevic, "Finite difference Frequency-Domain Treatment of Open Transmission Structures," *IEEE Trans. Microw. Theory Tech.*, vol. 38, no. 11, pp. 1609–1616, 1990.

[2.29] D. Johns, "New Frequency-Domain TLM Method for the Numerical Solution of Steady-State Electromagnetic Problems," *IEEE Proc. - Sci. Meas. Technol.*, vol. 141, no. 4, p. 310, 1994.

[2.30] A. Centeno, "A Comparison of Numerical Dispersion in FDTD and TLM Algorithms," *Asia-Pacific Conf. Appl. Electromagn. 2003. APACE 2003.*, no. Apace, pp. 128–131, 2003.

[2.31] M. Krumpholz, C. Huber, and P. Russer, "Field Theoretical Comparison of FDTD and TLM," *IEEE Trans. Microw. Theory Tech.*, vol. 43, no. 8, pp. 1935–1950, 1995.

[2.32] W. Gwarek, "On the Relationship Between TLM and Finite-Difference Methods for Maxwell's Equations (Comments)," *IEEE Trans. Microw. Theory Tech.*, vol. 35, no. 9, pp. 872–873, 1987.

[2.33] C. Eswarappa and W. J. R. Hoefer, "Bridging the Gap Between TLM and FDTD," *IEEE Microw. Guid. Wave Lett.*, vol. 6, no. 1, pp. 6–8, 1996.

[2.34] J. Carpes, W.P., G. Ferreira, A. Raizer, L. Pichon, and A. Razek, "TLM and FEM Methods Applied in the Analysis of Electromagnetic Coupling," *IEEE Trans. Magn.*, vol. 36, no. 4, pp. 982–985, 2000.

Chapter 3

Mesh Data Structures

3.1 Introduction

Fundamental to every scientific simulation is the representation of a geometrical problem space by a computer. This chapter discusses the data structures that allow a computer to understand and represent the problem space in memory such that algorithms which govern a simulation can be effectively applied. Further, the algorithms used to carry out a successful Unstructured TLM simulation are discussed, and how these are applied to the mesh data structures, expanding on the TLM theory from Chapter 2. Finally a study into how the choice of data structure impacts the computational resources of a simulation is provided.

Chapter 2 spoke about how unstructured meshes can better describe curved boundaries of a problem space with fewer mesh elements than a structured mesh. The draw back of this is the increase in complexity of the connectivity information of the mesh. Mesh elements are no longer uniformly spaced as they are with a regular Cartesian descritisation. Instead, a cloud of vertices requires organisation in terms of edge connectivity, and by extension, how these edges connect to describe the mesh faces. The underlying mesh data structures used to represent the geometrical prob-

lem space can have a large impact on the efficiency and memory consumption of the simulation [3.1]. Choosing a mesh data structure requires taking into consideration the mesh topology, as well as providing forethought to the algorithms necessary for simulating behaviour in the problem space [3.2]. The requirements can be generally divided into:

- **Topological Requirements** - What kind of mesh needs to be represented by the data structure? Fundamentally, meshes can be categorised as manifold or non-manifold. A manifold mesh is a mesh representing a surface that can be split along its various edges and subsequently unfolded such that the mesh lays flat without overlapping pieces [3.1]. Figure 3.1a) shows a simple example of a manifold mesh. This definition of a manifold mesh can be further expanded to mean the edges that constitute the mesh are simply connected. That is, no more than two faces share an edge, and no more than two faces share a vertex without an edge. Counter to this manifold case, a non-manifold mesh describes a mesh with the exact opposite definition; a configuration that cannot be unfolded into a continuous flat piece [3.3] [3.4]. Figure 3.1b) shows a configuration which conforms to this description; the mesh is non-manifold as more than two faces share an edge. Further Figure 3.1c) depicts a non-manifold mesh where more than two faces share a vertex without an edge. In addition to the deciding if manifold or non-manifold meshes are to be represented by the data structure, other decisions need to be made. For example, is the topology of individual mesh faces purely triangular, or will the mesh be a combination of triangular and quadrilateral structures? Depending on the application, perhaps arbitrary polygons need to be represented.

- **Algorithmic requirements** - What are the algorithms operating on the mesh? Is efficient access to local neighbourhoods of vertices, edges, and faces necessary? Will the mesh be static or will its geometry or connectivity change

over time? Is the only algorithmic requirement simply rendering the mesh? Perhaps there will be special memory requirements, for example, if the data set is extremely large.

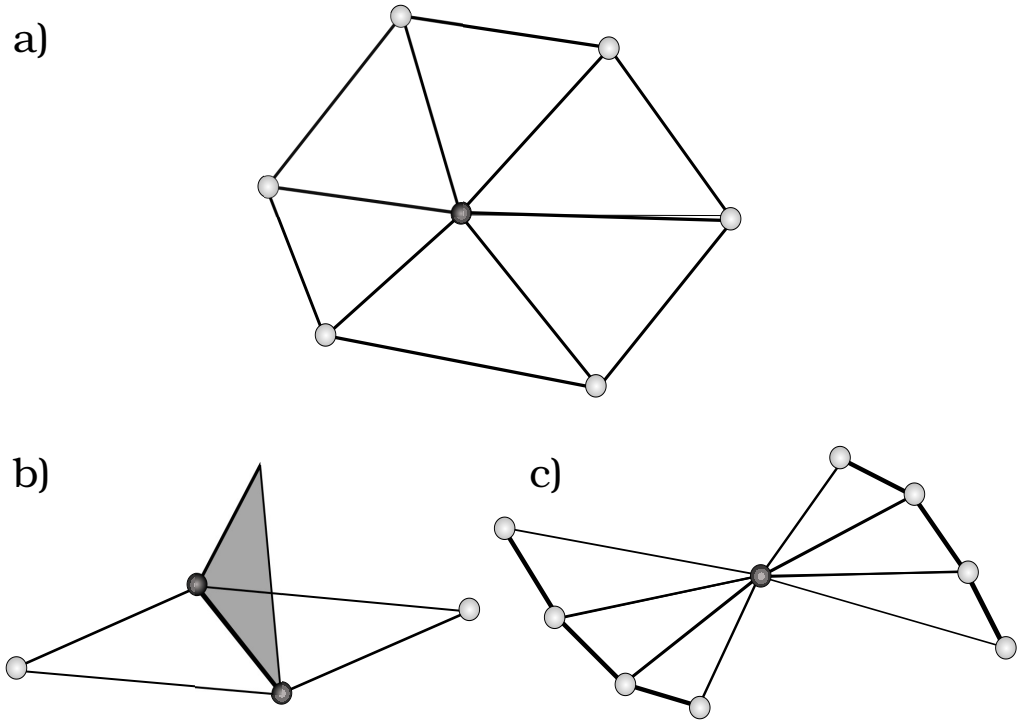

Figure 3.1: Examples of manifold and non-manifold meshes: a) A simple manifold mesh. b) Non-manifold mesh: Three or more faces share an edge. c) Non-Manifold: Two or more faces share a single vertex but no edge. This is strictly non-manifold, but most data structures can represent these meshes.

As Unstructured TLM is being implemented, purely triangular meshes will need to be represented, which describe physically realistic problem spaces and hence only manifold meshes will be required. Algorithmically, the time step evolution of the TLM simulation will require access to neighbouring elements constituting the mesh. Reducing the need for case distinctions in the form of *if-then* searches will have a big impact on the runtime of the simulation. However this will always come as a trade-off with memory consumption. How the adjacency information is stored and accessed will impact the computational expense of the simulation in terms of memory and runtime [3.5].

Evaluating a data structure requires measuring various criteria such as; the time taken to preprocess the construction of the mesh, the time taken to answer a specific query and the time to perform an operation as well as the memory consumption [3.4]. Before discussing the specifics of data structures, a method of theoretically approximating the memory consumption of the various structures will be defined. This will be provided using the Euler-Poincaré Formula.

3.2 The Euler-Poincaré Characteristic

The Euler-Poincaré formula characterises an interesting relationship between the number of vertices, the number of edges and the number of faces of a closed mesh [3.6]:

$$V - E + F = 2(1 - g), \tag{3.1}$$

where V is the number of vertices, E is the number of edges, F the number of faces and g is the genus of the geometry. The genus describes how many holes or *handles* the geometry has; a sphere has a genus of 0, a torus has a genus of 1, and a double torus has a genus of 2. For most practical applications, the genus is small compared to the number of elements in a mesh, and the right hand side of the equation can be assumed to be negligible. Given that each mesh triangle is bounded by three edges and that each interior edge is incident to two triangles, the following interesting mesh relationships can be derived [3.1]:

- The number of triangles is approximately twice the number of vertices: $F \approx 2V$,
- The number of edges is approximately three times the number of vertices: $E \approx 3V$, and

- The average vertex valance (number of incident edges) is assumed to be 6.

These relationships will become important when estimating the resource consumption of the data structures beginning with the face-based data structure in the following section.

3.3 Face-Based Data Structures

One of the simplest ways to represent a surface mesh involves storing a set of individual polygonal faces represented by their vertex positions [3.7]. Considering the simple case of a triangular mesh, storing 3 vertex positions per mesh element, the impact on memory consumption using 32-bit single precision numbers to represent the coordinates yields a requirement of 36 bytes per triangle in 3 dimensional space (3 coordinate components per vertex, 3 vertices per face with each vertex coordinate component requiring 4 bytes in memory). Referring back to Equation (3.1), the number of faces is approximately twice the number of vertices. This data structure consumes, on average, 72 bytes per vertex.

Outside of data representations in computer memory, data exchange formats such as the stereolithography (STL) file format, a standard format for representing CAD models as triangulated surfaces, use this representation [3.8]. An example of how these meshes are represented in this file format, and as a structure in memory is shown in Table 3.1a). This data structure does not represent the mesh connectivity. Because the connectivity cannot be explicitly accessed, and the vertices and associated information are replicated as many times as the degree of the vertices, this is not a fit representation for most algorithmic applications.

We can avoid this redundancy by using an indexed vertex approach (shown in Table 3.1b) where by the data structure involves storing an array of vertices and

a) Triangles

x_{11}	y_{11}	z_{11}	x_{12}	y_{12}	z_{12}	x_{13}	y_{13}	z_{13}
x_{21}	y_{21}	z_{21}	x_{12}	y_{22}	z_{22}	x_{23}	y_{23}	z_{23}
...			...			...		
...			...			...		
x_{f1}	y_{f1}	z_{f1}	x_{f2}	y_{f2}	z_{f2}	x_{f3}	y_{f3}	z_{f3}

b) Vertex List

v_1	x_1	y_1	z_1
v_2	x_2	y_2	z_2
...		...	
...		...	
v_n	x_v	y_v	z_v

Face List

f_1	v_1	v_2	v_3
f_2	v_4	v_2	v_1
...		...	
...		...	
f_n	v_n	v_n	v_n

Table 3.1: a: Face-set structure for triangles, individual triangles are represented by vertex positions. Vertex positions are not indexed. b: Indexed face-set structure for triangles. vertices are indexed in an array and then associated with a face

encodes polygons as a set of indices into an array. In the case of again using a 32-bit single precision representation to store the vertex coordinates and face indices, in a triangular mesh 12 bytes are used for each vertex and for each triangle. As a result, 12 bytes per vertex are used and in addition to 12 bytes per face, consuming a total of 36 bytes per vertex, due to the removed redundancy of duplicated coordinates; only half of the face-set approach.

File formats such as OFF, OBJ and VMRL [3.9] all use the indexed face-set data structure approach, due to the simple and efficient storage ideal for static data representation such as colour or texture mapping. These formats are typically used for computer graphics applications, due to the ability of attaching such attributes to vertices and faces. But without the connectivity information, this data structure

is not efficient enough for scientific simulations. Large and expensive searching is required to gain knowledge of neighbouring mesh elements, and to recover local adjacency information for vertices and faces. For large meshes, this overhead can be considerable. There is usually a set of operations used by algorithms, including TLM, which relate to the access of specific mesh elements and how the mesh is traversed [3.1] [3.10]:

- A representation of vertices, edges and faces
- Access to individual vertices, edges and faces, including the enumeration of all elements constituting the mesh.
- Given an edge, access to the vertices at the end points; this is essential for calculating the midpoint of an edge, for example, and referencing adjacent faces.
- Access to incident faces of an edge.
- Orientated traversal of edges of a face, i.e finding the next (or previous) edge in a face.
- Given a vertex, at least one incident face or edge must be accessible. This allows the one-ring neighbourhood for a manifold mesh (Figure 3.1a)) to be accessed.
- Allow the storage of any custom data at vertices, edges or faces.

A standard face-based data structure for triangular meshes that includes connectivity information, involves the storage of references to its three vertices, as well as references to its neighbouring triangles. Figure 3.2 demonstrates how the individual objects that describe the mesh relate to each other. A Vector object defines the coordinates of each individual vertex. A vertex object stores its position, and refer-

ence to one of its incident faces. Each face is subsequently defined by the storage of an array of references to its three vertices shown at (1) in Figure 3.2, in addition to the adjacency information through the use of references to its neighbouring faces, shown at (2). Based on this connectivity information, circulation around a vertex in order to access its one-ring neighbourhood is now possible, in addition to now being able to perform the operations listed above.

The drawback with this data structure is that it does not explicitly store edges, meaning no information can be attached to the edges. Further, enumerating the one-ring neighbourhood of a centre vertex requires a large number of case distinctions by means of *if-then* branching. This can result in a large amount of time spent simply searching for the correct adjacent face, especially for large meshes. For scientific simulations which require iterating over the entire mesh, often thousands of times, this can be a burden on the runtime of simulations. Finally, although we are restricting this project to triangular meshes, if this data representation is to be

Vector
x y z

Vertex	
Vector	position
FaceRef	face

Face		
(1)	VertexRef	vertex [3]
(2)	FaceRef	neighbour [3]

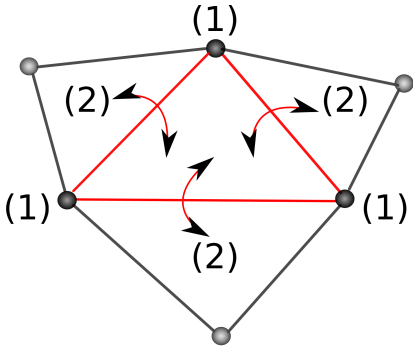

Figure 3.2: Connectivity information for a Face-Based data structure: a Vector defines the coordinates, the Vertex object stores its position as well as a reference to a connecting Face and each Face stores an array of references to its three vertices (1) and adjacent faces (2).

used for general polygonal meshes, the data type for faces no longer has constant size, which makes the implementation more complex and less efficient. An attempt to resolve some of these drawbacks is provided by the edge-based data structure, described in the next section.

3.4 Edge-Based Data Structures

Logically, data structures for general polygon meshes are edge based, since the connectivity primarily relates to the mesh edges. Well known edge based data structures are the winged-edge [3.11] and quad edge data structure [3.12]. What follows is the winged-edge data structure which is primarily used for triangular meshes [3.13], however it can be expanded to cater for arbitrary polygonal meshes.

In the winged edge data structure, when dealing with manifold meshes of surfaces, each edge has exactly 2 incident faces, and each face is orientated such that each edge receives a direction from its incident faces. This ordering can be clockwise or anticlockwise, provided that consistency of this orientation is enforced across the mesh. Figure 3.3 shows an example of a clockwise orientation of edges about a face, with the direction represented by the dashed lines. Here, edge a has incident faces 1 and 2, and the traversal of each face induces a predecessor edge and a successor edge. From Figure 3.3, the predecessor and successor edges of edge a with respect to *face 1* are d and b respectively, with the predecessor and successor edges with respect to *face 2* being edges c and e respectively.

Once the orientation of the edge is defined, one can assemble 9 pieces of information from an edge; its index, the start and end vertices, the left and right faces (i.e. the wings), the predecessor and successor edges when traversing the left face, and the predecessor and successor edges when traversing the right face. These are

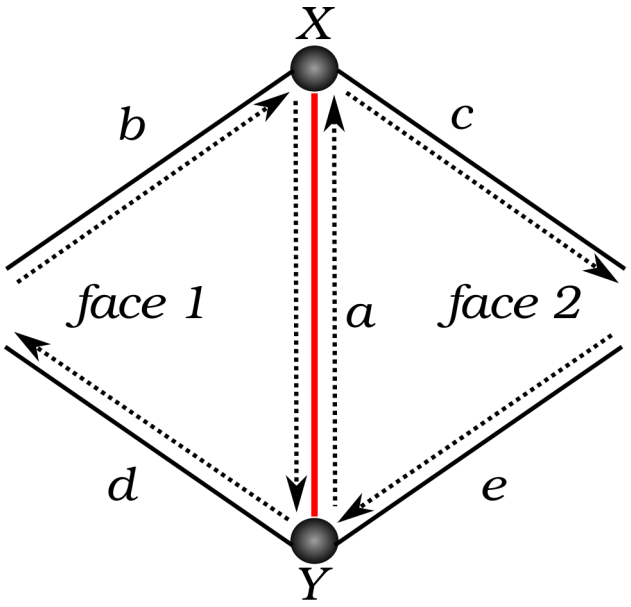

Figure 3.3: Winged-Edge data structure. Each face has a fixed orientation, for the clockwise orientation shown, edge a has incident faces *face 1* and *face 2*. The predecessor and successor edges of edge a with respect to *face 1* are d and b respectively, and predecessor and successor edges with respect to *face 2* being edges c and e respectively.

Edge	**Vertex**		**Face**		**Left Traversal**		**Right Traversal**	
Name	Start	End	Left	Right	Predecessor	Successor	Predecessor	Successor
a	X	Y	*1*	*2*	d	b	c	e

Table 3.2: The connectivity information known to an edge, based on the representation from Figure 3.3

summarised in Table 3.2, which relate to the configuration in Figure 3.3.

The connectivity of the winged edge data structure of a mesh is described using 3 essential lists. The Vertex list, Face list and Edge list. These are represented in Figure 3.4, where each entry in the Vertex list contains its position in space, and a reference to an incident edge. Each entry in the Face list only contains a reference to an incident edge in the Edge list. For example, the Vertex list entry for vertex X

in Figure 3.3 may reference any one of edges a, b or c. Similarly, the incident edge of *face 1* may be edge b, d, or a. There is one entry for each edge in the Edge list. Referring to Figure 3.4, each edge entry consists of references to its two end vertices (1), a reference to the two adjacent faces (2), the next and previous edges of *face 1* and *face 2* shown at (3) and (4), which are defined by the chosen orientation of the faces.

Referring back to the formula in equation (3.1), the number of faces is approximately twice the number of vertices, and the number of edges is approximately 3 times the number of vertices. This means in terms of memory consumption, this representation theoretically leads to 16 bytes per vertex (12 bytes for storing the coordinates and 4 bytes per reference), 32 bytes per edge (8 references in total) and 4 bytes per face to reference an edge. This totals 120 bytes per vertex constituting a mesh. The increased knowledge of connectivity impacts the available memory resources as it requires more than the 72 bytes per vertex used by the face-based structure.

Vector		
x	y	z

Vertex	
Vector	position
EdgeRef	edge

Face	
EdgeRef	edge

Edge		
(1)	VertexRef	vertex [2]
(2)	FaceRef	face [2]
(3)	EdgeRef	next [2]
(4)	Edgeref	previous [2]

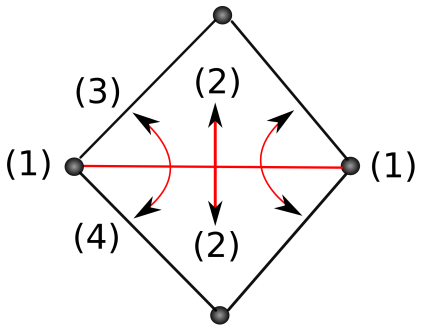

Figure 3.4: Winged-Edge data structure showing the connectivity information stored. Each edge references its two vertices, the two adjacent faces and the next and previous edges

An edge based data structure can represent arbitrary polygonal meshes, however

traversing the one-ring neighbourhood still requires distinctions between the various constituting vertices. Is the vertex at the head or the tail of the edge for example? It should be noted that the number of case-distinctions in order to enumerate the elements constituting the mesh is fewer than the face-based structure. An expansion of this winged edge data structure exists which facilitates the traversal of the entire mesh without any *if-then* branching, which is addressed in the following section, with the Halfedge data structure.

3.5 Halfedge-Based Data Structures

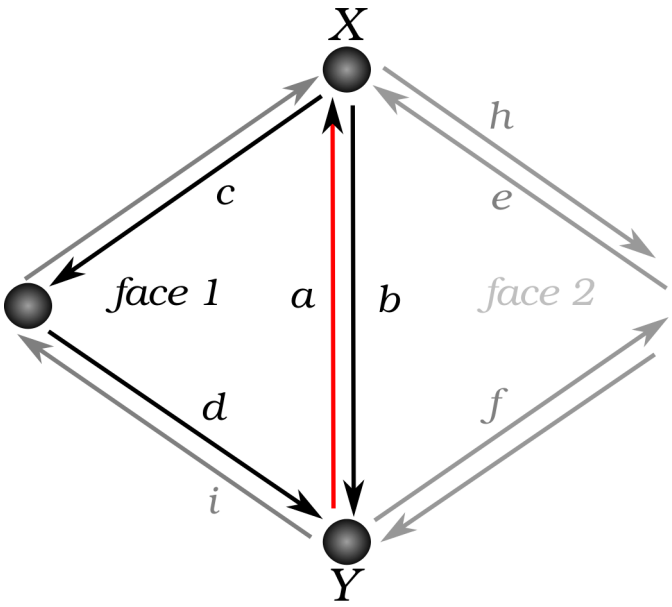

Figure 3.5: The Halfedge data structure. Each halfedge has a fixed anticlockwise orientation. The halfedge a references incident *face 1* and the next and (optionally) previous halfedges c and b respectively in addition to this opposite halfedge b.

As the name suggests, in a halfedge data structure [3.14] the edges are split in half such that the edge now becomes two orientated halfedges. By splitting the edges in such a way, it is possible to traverse an entire polygonal mesh without any case distinctions in the code to derive the mesh conectivity during runtime, as is required

by the data structures previously discussed [3.15].

Figure 3.5 shows an example of a halfedge representation. In a halfedge data structure, halfedges are oriented consistently in a counter clockwise direction around each face and along each boundary of the mesh. Three tables encapsulate the connectivity in the form of Face Tables, Vertex Tables, and Halfedge Tables. Each face stores a reference to one of its halfedges. With respect to *face 1* in Figure 3.5, this can be any of the halfedges a, c or d. Each vertex references one outgoing halfedge; that is, a halfedge which begins at this vertex. This means vertex X can either point to halfedge b or j and conversely, vertex Y can reference f or h. Unlike the winged-edge structure, which only has one entry per edge, each edge effectively has two entries in the halfedge table, one for each oppositely orientated halfedge. Examples of these halfedge table entries are given in Table 3.3 for the two halfedges a and b in Figure 3.5. Each halfedge provides a reference to [3.16]:

- the vertex it emanates from (Y),
- the face it belongs to (*face 1*),
- the next halfedge inside the face (ordered counter-clockwise) (c),
- optionally, the previous halfedge in the face (d),

Halfedge	**Vertex**	**Face**	**Halfedge Traversal**		
name	start	incident face	Next	Previous	Opposite
a	Y	*face 1*	c	d	b
b	X	*face 2*	f	e	a

Table 3.3: Halfedge table entries relating to halfedges a and b in Figure 3.5. Each halfedge references its starting vertex, its incident face, the next halfedge, anticlockwise in the incident face, and (optionally) the previous halfedge in the incident face.

- and the opposite halfedge, known as its *flip* halfedge (b).

This connectivity is generalised in Figure 3.6. It can be seen that there are no arrays of references in the Halfedge encapsulation which are present in the edge and face representations. Single references are made to vertices and faces that constitute the mesh, with an explicit navigation from halfedge to halfedge.

Vector		
x	y	z

Vertex	
Vector	position
HalfedgeRef	halfedge

Face	
HalfedgeRef	halfedge

Halfedge		
(1)	VertexRef	vertex
(2)	FaceRef	face
(3)	HalfedgeRef	next
(4)	HalfedgeRef	previous
(5)	HalfedgeRef	opposite

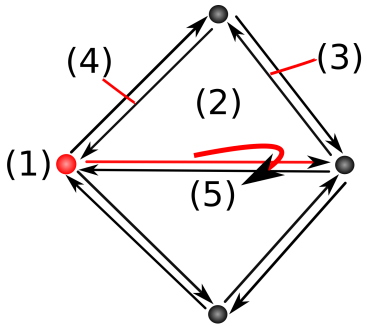

Figure 3.6: connectivity information stored in a halfedge based structure: each halfedge stores a reference to the vertex it starts from (1), its face (2), the next halfedge (3) and the previous halfedge (4) and the opposite halfedge (5)

Defining these links between the mesh items means it is now possible to circulate around a face in order to enumerate all of its vertices, half edges and neighbouring faces, which can now be achieved without expensive $if - then$ searches. Rather the connectivity and access of the various mesh elements can be implemented through the use of pointers or indices. Figure 3.7 demonstrates an example of how a one-ring neighbourhood is traversed about a centre vertex. An outgoing halfedge from the centre vertex in a) is flipped in order to traverse to its opposite halfedge. The vertex that the halfedge was pointing to in a) can now be enumerated through the opposite halfedge. The next halfedge can then be accessed as shown in c) and this process

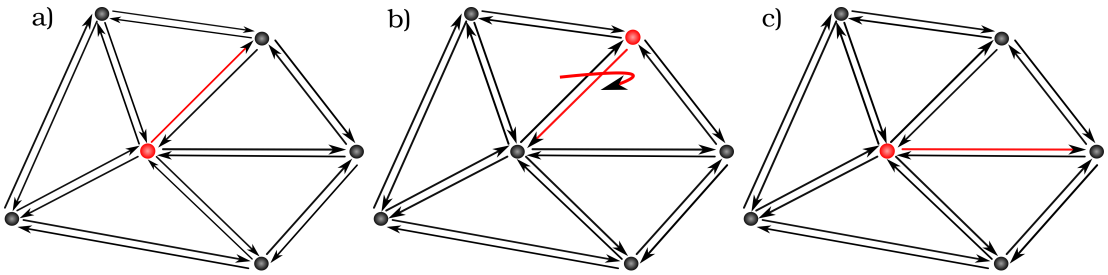

Figure 3.7: Traversing the halfedge data structure. a: outgoing halfedge from centre vertex, b: flip of initial halfedge moves to neighbouring face, c: next halfedge in sequence. This is repeated such in order to traverse the full one ring neighbourhood

is repeated until the starting halfedge is reached again, and the entire one-ring is traversed.

It follows that each boundary of the mesh can be seen as an empty face of potentially high degree, which can be traversed as any other face constituting the mesh [3.16]. This is depicted in Figure 3.8. The orientation of traversal is consistent with the interior mesh halfedges. The ability to enumerate the boundary independently allows algorithms pertaining to the boundary to be applied without the need of traversing the entire mesh to find the boundary elements, or resorting to case distinctions to determine if the edge is a boundary edge or not.

An explicit representation of edges is also obtained as a pair of halfedges; this is important for associating data with an edge as opposed to halfedges. It should also be noted that the previous halfedge does not need to be stored explicitly, as it can be derived from the links to the next halfedges; traversing to the next halfedge twice will result in the access of the previous halfedge, however this can be included for the sake of performance, allowing the enumeration of halfedges in a clockwise orientation, not only anticlockwise.

Since the number of halfedges is about 6 times the number of vertices, the total theoretical memory consumption is 16 bytes per vertex, 20 bytes per halfedge, and

4 bytes per face. The summation of these yields an average memory consumption of 144 bytes per vertex, making this representation the most memory intensive of those studied.

3.6 Data Structures for UTLM

With a definition of how the topology of the problem space can be represented, attention is now turned to how the algorithms for UTLM can be applied to the topological data structures.

In Chapter 2 a method for representing the propagation of electric and magnetic fields using the analogy of transmission lines was derived. Here we will take a more detailed look at how a TLM simulation appears algorithmically and how this is applied to the data structures.

There are three main steps to the runtime evolution of a UTLM simulation. The excitation, scatter and connect. But beforehand, the link line and stub parameters

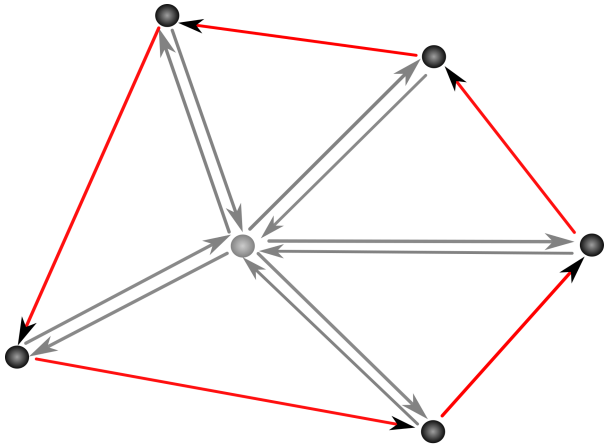

Figure 3.8: Traversing the boundary with a half-edge data structure. Each boundary can be seen as an empty face, allowing the traversal of the boundary in a counter clockwise direction

need to be defined and stored. How these are represented will depend on the chosen data structure. If a face-based structure is chosen, representations of the link lines will need to be stored in the faces. If an edge-based or halfedge data structure is used then the link line representation can be stored at the edges.

Figure 3.9 shows two adjacent mesh triangles with a link length configuration between the two circumcentres, *node n* and *node m*. The circumcentre of face n is a distance $\Delta_i(n) + \Delta_i(m)$ from the circumcentre of face m, where $\Delta_i(n)$ is the link length of $port_i(n)$, and $\Delta_i(m)$ is the link length of $port_i(m)$.

A voltage signal travels between two adjacent nodes via the link line with a time step Δt. It therefore takes $\Delta t/2$ to travel from a node of a face to the representative port. The shortest link length $\Delta_{i_{min}}$ across the mesh governs the maximum allowable time step of the simulation. The time step obeys the following restraint to minimise dispersion [3.17]:

$$\Delta t_{max} = \Delta_{i_{min}} \sqrt{2\varepsilon\mu}, \tag{3.2}$$

where ε is the permittivity of the material, and μ is the permeability.

Figure 3.10 shows the link impedances and stub admittances for each link line. The link and stub admittances for each port are given by:

$$Y_{link_i} = \frac{1}{Z_{link_i}} = \frac{l_i \Delta t}{\mu_0 \Delta_i}, \tag{3.3}$$

$$Y_{stub_i} = \frac{\varepsilon_0 \varepsilon_r l_i \Delta_i}{2\Delta t} - Y_{link_i} \tag{3.4}$$

The excitation involves applying a voltage to the node or nodes that coincide with the source location. In this case, the voltage value is applied to the circumcentre of the mesh element, however the excitation could be applied directly to the link lines of the cell.

Transmission line theory states that the total voltage at any point on a transmission

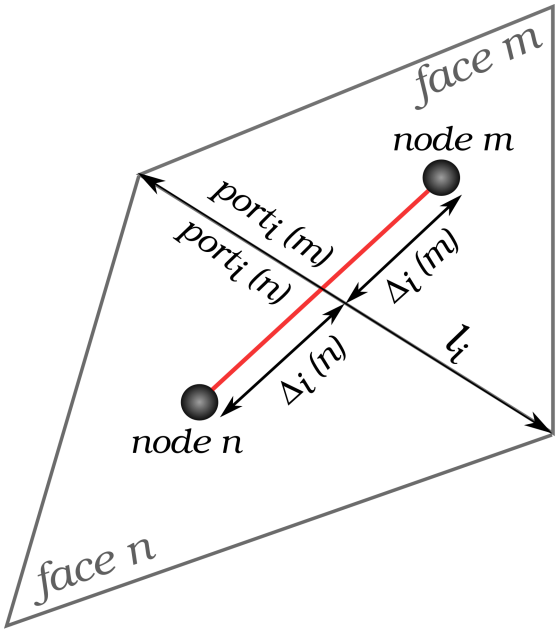

Figure 3.9: UTLM link lengths: Δ_i are the link lengths of the link lines, l_i is the intersecting triangle edge length corresponding to $port_i(n)$ and $port_i(m)$

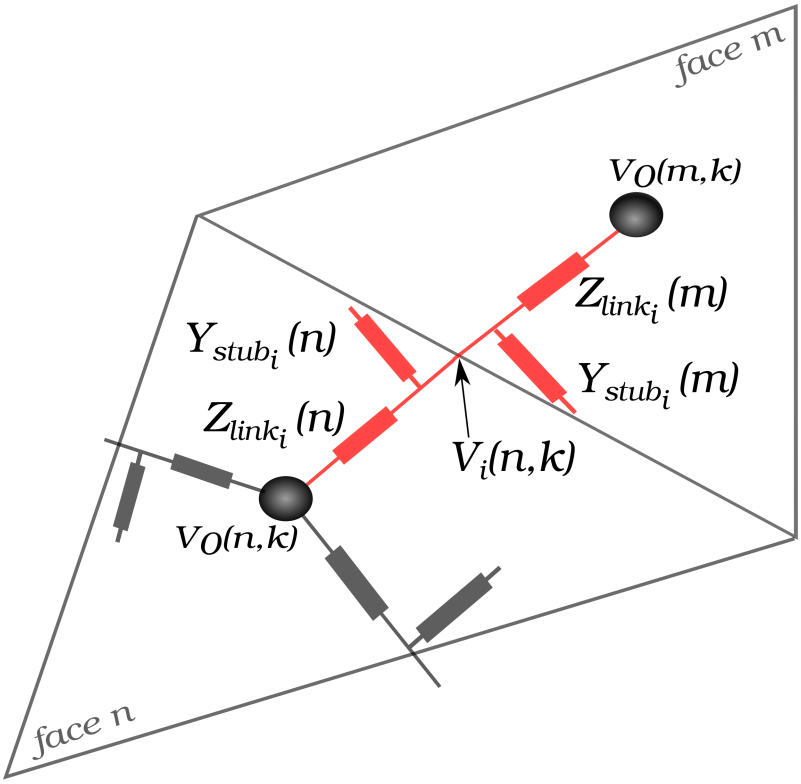

Figure 3.10: UTLM admittances: Y_{stub_i} are the stub admittances, and Z_{link_i} are the link line impedances.

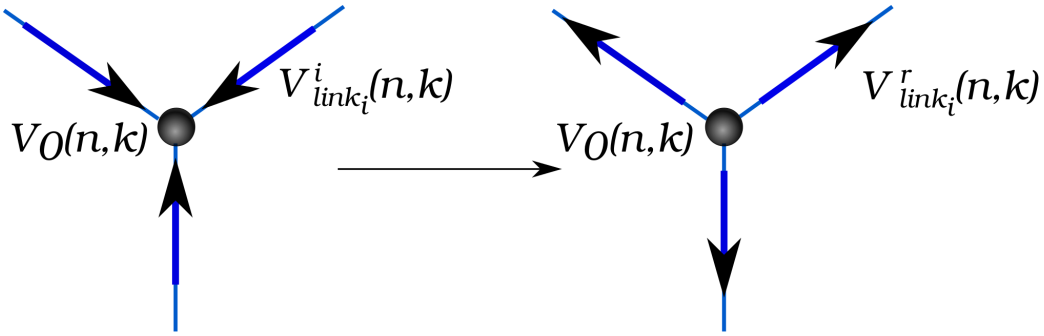

Figure 3.11: The scatter process of UTLM, occurring at the node n at time step k.

Figure 3.12: The connect process of UTLM, occurring at the port interface between node n and node m at time step k

line is derived from the sum of the incident and reflected voltages. At any given time step k, the voltages are incident to the nodes, from which they are scattered. Figure 3.11 shows the incident voltages $V^i_{link_i}$ to node n, which initiate the calculation of the reflected voltages $V^r_{link_i}$, which are, in turn, reflected back towards to the ports. These reflected voltages subsequently connect with neighbouring nodes at the ports (Figure 3.12), which defines the new incident voltages at the next time step $k+1$.

The scattering of voltages at the nodes can be calculated using the Thevenin equivalent circuit of the node which is shown in Figure 3.13. First the nodal voltage is calculated using Kirchhoff's current law,

$$ {}_kV_0(n) = \left[\frac{2_kV^i_{link_1}}{Z_{link_1}} + \frac{2_kV^i_{link_2}}{Z_{link_2}} + \frac{2_kV^i_{link_3}}{Z_{link_3}}\right] Z_{eq} \tag{3.5}$$

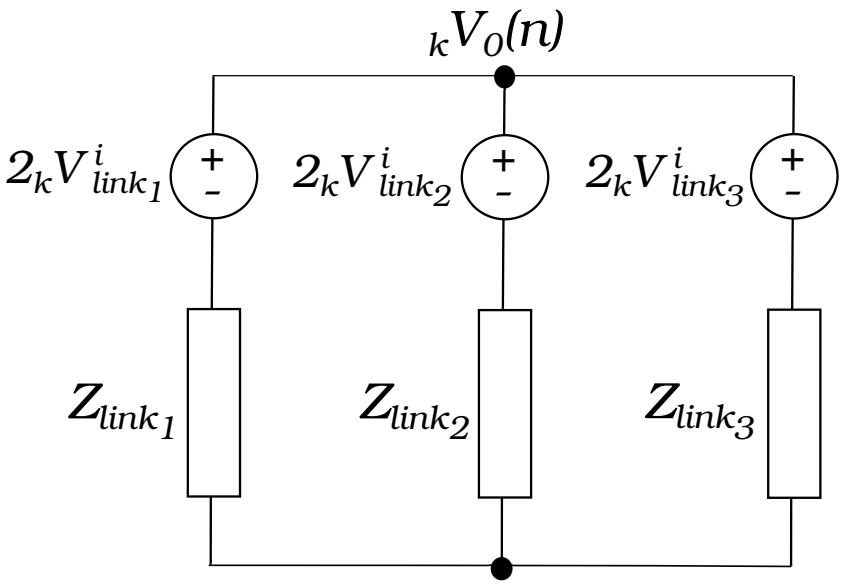

Figure 3.13: Thevenin equivalent circuit for unstructured TLM node n, used for deriving the scatter algorithms.

where:

$$\frac{1}{Z_{eq}} = \frac{1}{Z_{link_1}} + \frac{1}{Z_{link_2}} + \frac{1}{Z_{link_3}} \tag{3.6}$$

The reflected voltages in the scattering process, at each link line in face n for time k, are then calculated as:

$$_kV^r_{link_1}(n) = {}_kV_o(n) - {}_kV^i_{link_1}(n) \tag{3.7}$$

$$_kV^r_{link_2}(n) = {}_kV_o(n) - {}_kV^i_{link_2}(n) \tag{3.8}$$

$$_kV^r_{link_3}(n) = {}_kV_o(n) - {}_kV^i_{link_3}(n) \tag{3.9}$$

The reflected voltages derived in the scatter process now propagate to neighbouring nodes and reach the ports at the interface between $faces$ n and m ready for the connection process. From the Thevenin equivalent circuit for the connection process shown in Figure 3.14 the voltage at the port is calculated as follows:

$$_kV_1 = \left[\frac{2_kV^i_{link_1}(n)}{Z_{link_1}(n)} + \frac{2_kV^i_{stub_1}(n)}{Z_{stub_1}(n)} + \frac{2_kV^i_{link_1}(m)}{Z_{link_1}(m)} + \frac{2_kV^i_{link_3}(m)}{Z_{link_3}(m)}\right] Z_{eq} \tag{3.10}$$

where

$$\frac{1}{Z_{eq}} = \frac{1}{Z_{link_1}(n)} + \frac{1}{Z_{stub_1}(n)} + \frac{1}{Z_{link_1}(m)} + \frac{1}{Z_{stub_1}(m)} \tag{3.11}$$

The reflected voltages on the transmission lines on either side of *port* 1 are updated as follows:

$$_kV^r_{link_1}(n) = kV_1 - {}_kV^i_{link_1}(n) \tag{3.12}$$

$$_kV^r_{stub_1}(n) = kV_1 - {}_kV^i_{stub_1}(n) \tag{3.13}$$

$$_kV^r_{link_1}(m) = kV_1 - {}_kV^i_{link_1}(m) \tag{3.14}$$

$$_kV^r_{stub_1}(m) = kV_1 - {}_kV^i_{stub_1}(m) \tag{3.15}$$

These reflected voltages at the port subsequently become the incident voltages for the next time step $k+1$:

$$_{k+1}V^i_{link_1}(n) = {}_kV^r_{link_1}(n) \tag{3.16}$$

$$_{k+1}V^i_{stub_1}(n) = {}_kV^r_{stub_1}(n) \tag{3.17}$$

$$_{k+1}V^i_{link_1}(m) = {}_kV^r_{link_1}(m) \tag{3.18}$$

$$_{k+1}V^i_{stub_1}(m) = {}_kV^r_{stub_1}(m) \tag{3.19}$$

This connection process is then repeated for each link line contained in $face\ n$.

A link line for a port facing a boundary has a special connection procedure. Figure 3.15 shows the Thevenin equivalent representations for different boundary types, such that $face\ m$ is replaced by a boundary in Figure 3.10. These are the short-circuit, open-circuit and matched boundary.

For a short circuit implementation shown in Figure 3.15 a), the reflected voltage is simply:

$$_kV^r_{link_1}(n) = -{}_kV^i_{link_1}(n) \tag{3.20}$$

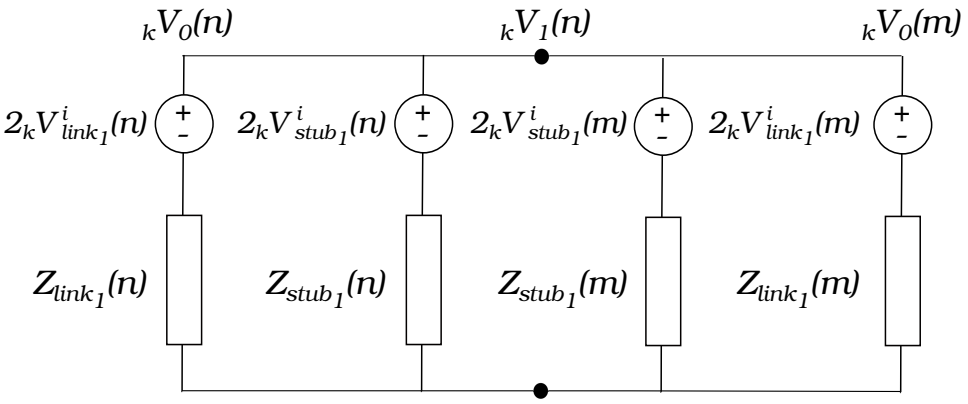

Figure 3.14: Thevenin equivalent circuit for the connection process

If the boundary presents itself as an open circuit, as shown in Figure 3.15 b), the voltage at $kV_1(n)$ reduces to:

$$kV_1(n) = \left[\frac{2_kV^i_{link_1}(n)}{Z_{link_1}(n)} + \frac{2_kV^i_{stub_1}(n)}{Z_{stub_1}(n)}\right] Z_{eq} \tag{3.21}$$

where

$$\frac{1}{Z_{eq}} = \frac{1}{Z_{link_1}(n)} + \frac{1}{Z_{stub_1}(n)}. \tag{3.22}$$

In the case of a match boundary condition, with a Thevenin equivalence presented in the circuit of Figure 3.15 c), the port voltage $kV_1(n)$ becomes:

$$_kV_1(n) = \left[\frac{2_kV^i_{link_1}(n)}{Z_{link_1}(n)} + \frac{2_kV^i_{stub_1}(n)}{Z_{stub_1}(n)}\right] Z_{eq} \tag{3.23}$$

where, for the matched boundary case,

$$\frac{1}{Z_{eq}} = \frac{1}{Z_{link_1}(n)} + \frac{1}{Z_{stub_1}(n)} + Z_{BND}. \tag{3.24}$$

For the open and matched cases, the voltage reflected from the boundary is calculated from:

$$_kV^r_{link_1} = {}_kV_1(n) - {}_kV^i_{link_1}, \tag{3.25}$$

$$_kV^r_{stub_1} = {}_kV_1(n) - {}_kV^i_{stub_1} \tag{3.26}$$

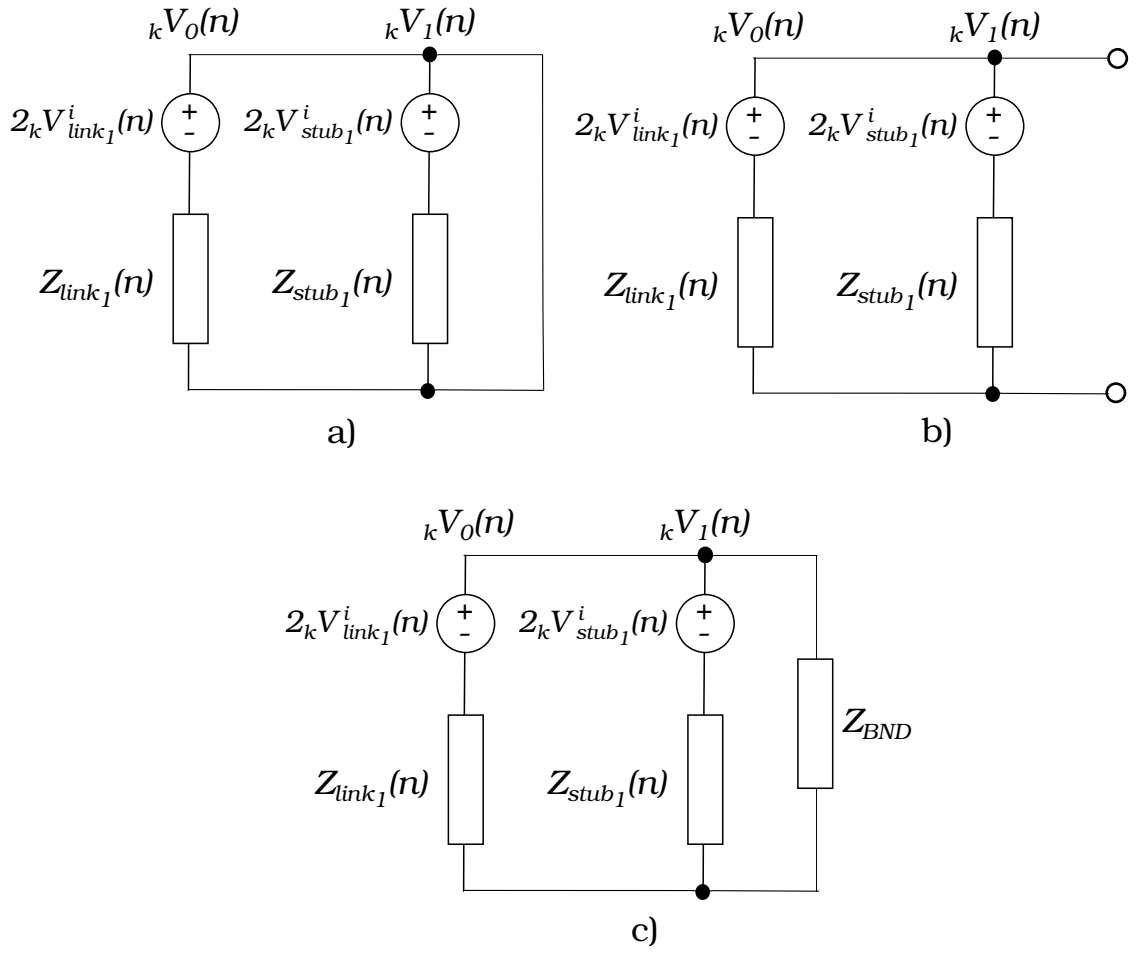

Figure 3.15: Thevenin equivalent circuit for a face adjacent to a short circuit boundary (top left), open circuit boundary (top right) and matched boundary (bottom).

The data structure must facilitate the representation of link lines and their parameters at the edges of each element, in addition to providing access to, and updating the voltages at the node centres and ports.

If a face-based data structure is used to host the TLM simulation, one can hypothesise that this would be the most memory efficient method to use. It was theoretically shown in section 3.3 that the memory foot print for representing the topology of the mesh using this structure is lower than the edge based structures discussed. However no information can be attributed to the edges and case distinctions in the form of $if - then$ searches must be performed at every time step in order to derive the ad-

jacency information and ensure the connection phase is carried out with the correct neighbour face at the ports. This, of course, will adversely impact the runtime of the simulation. The greater the number of elements that constitute the mesh, the greater the impact on the runtime this structure can have.

Using an edge-based data structure facilitates the storage of link line parameters, at the edges, which coincide with the adjacency information of neighbouring mesh elements. This removes the runtime overhead of searching for the correct neighbouring port. However case distinctions are still necessary for deducing the correct subsequent edge about a face.

The halfedge data structure solves the issue of case distinctions as the entire connectivity of the topology is stored. An explicit representation of the link lines can be stored with each halfedge storing a reference to its corresponding link line. Each link line then stores its attributes such as the link and stub admittance, along with voltages local to the transmission line, to be used and updated in the evolution of the TLM algorithms. The connect phase for the boundary and the mesh interior can be independently carried out, as traversal of the mesh boundary is possible without the requirement of checking if the edge is a boundary edge every time step. This method allows the TLM network to be explicitly and efficiently traversed, however, it understandably comes with the consequence of requiring the largest memory footprint.

In the following section, an experimental study into the runtime and memory use of these data structures is presented, while providing validation for the Unstructured TLM algorithms through a comparison of simulation results for a rectangular resonator and the corresponding analytical solutions.

3.7 A Comparison of Data Structures for UTLM

In order to provide a comparison of data structures for the purposes of UTLM, frameworks have been built to compare the speed and memory requirements of each data structure, using the C++ programming language. Each individual framework is capable of loading and representing a mesh using one of three data structures discussed in this chapter; the face-based, edge-based and halfedge-based data structures.

By providing a range of mesh densities, the impact on memory consumption a data structure imposes can be compared; subsequently running a UTLM simulation on these meshes will provide an insight into the efficiency of the data structure in terms of the runtime of the simulation. These tests also serve to provide validation of the UTLM two dimensional algorithms. A rectangular resonator of height of 0.1m and width of 0.2249m is modelled and meshed using varying degrees of mesh refinement. As analytical solutions for the resonant frequencies are easily calculated for such a structure, a direct comparison with the UTLM simulation results can be readily made in order to provide validation for the algorithms operating on the mesh. Figure 3.16 shows examples of the meshed geometry using 303 faces (a), 1050 faces (b) and 6860 faces (c). All tests were run on a desktop computer with an Intel quad core i7, 2.67GHz processor, 16GB of RAM, and using a Linux operating system (Ubuntu 14.10).

3.7.1 Memory Consumption Results

Meshes describing a rectangular resonator are used to investigate the impact on memory consumption. The geometry is meshed using a Delaunay mesh criteria [3.18], and subsequently reducing the maximum element size in the triangulation

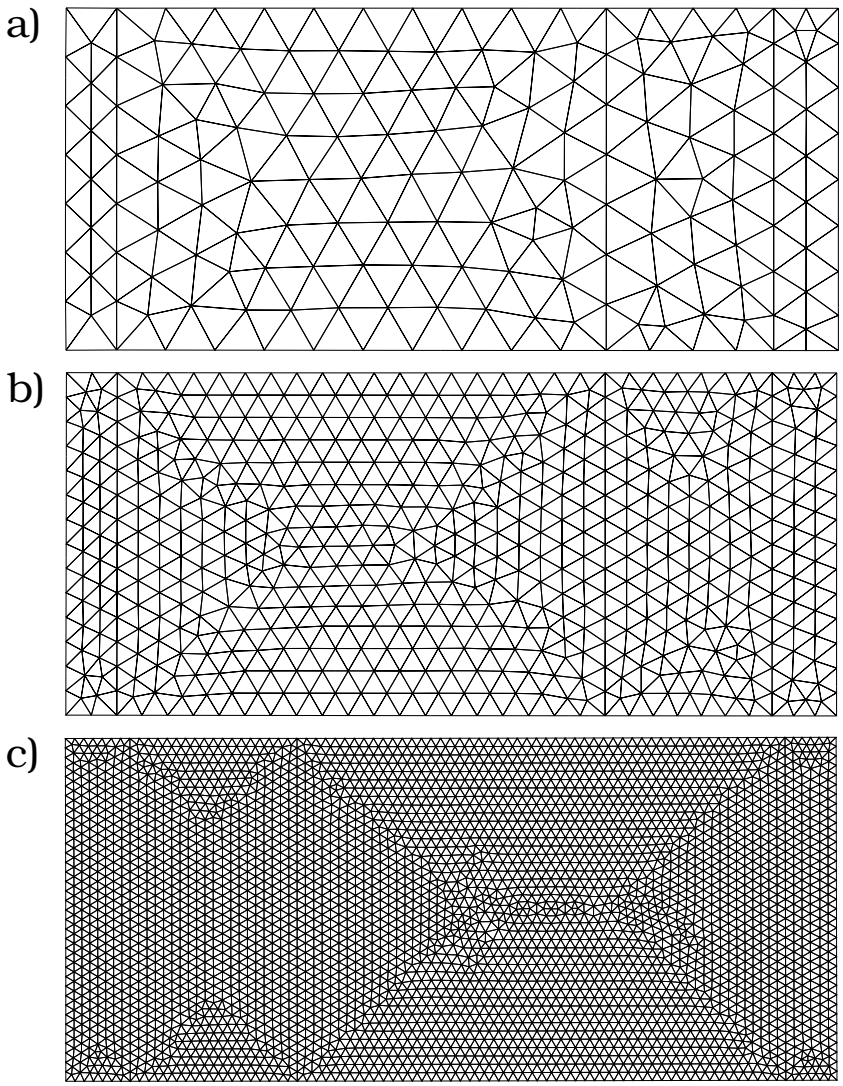

Figure 3.16: A rectangular resonator of width 0.1m and length 0.2249m meshed using a: 303 faces, b: 1050 faces and c: 6860 faces

algorithm to produce a set of meshes with increasing element density. The number of faces constituting the meshes range from 303 to 296356. The UTLM parameters, pertaining to the link line and stub admittances are calculated according to Equation (3.4) and stored to provide a total memory consumption prior to running the TLM simulation. The total memory to load the mesh and simulation parameters are provided in Table 3.4.

Number of	**Number of**	**Memory Consumption (MB)**		
Faces	**Vertices**	**Face-Based**	**Edge-Based**	**Halfedge-Based**
303	**177**	0.026	0.041	0.044
628	**350**	0.053	0.082	0.091
1050	**571**	0.088	0.136	0.149
3410	**1788**	0.283	0.433	0.476
6860	**3549**	0.567	0.865	0.950
17882	**9130**	1.473	2.240	2.459
55938	**28301**	4.598	6.976	7.655
296356	**148843**	24.334	36.827	40.400

Table 3.4: Memory consumption in MB of the face-based, edge-based and halfedge-based data structures for varying degrees of mesh density.

The results demonstrate that the face-based data structure is the most memory efficient, while the halfedge data structure requires the largest amount of memory to load and represent the mesh connectivity. This is expected, due to the additional number of references to mesh elements required to represent the mesh in a halfedge fashion. The edge-based data structure requires more memory than the face-based data structure, but less than the halfedge representation. This again is due to requiring fewer references than the halfedge structure to represent the connectivity, providing a median option. It should be noted that in order to represent a 2D mesh in memory, using any of these structures provides a memory efficient representation

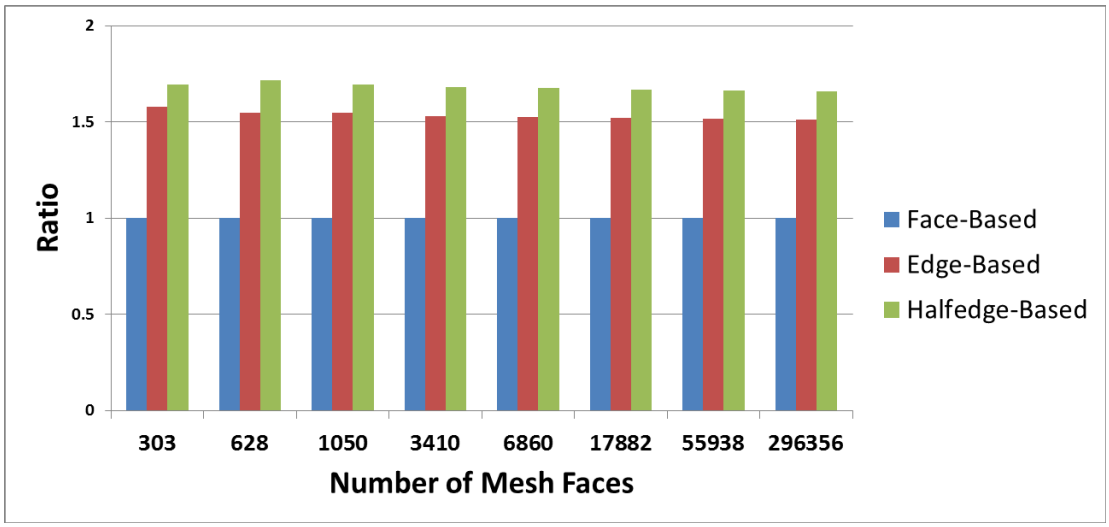

Figure 3.17: Ratio of memory consumption between the edge-based and halfedge-based data structures, compared with the face-based strcuture as reference. The memory consumption consists of the memory needed to represent the mesh and the stored TLM parameters

which will not tax the RAM of any modern desktop computer. If meshes with a significantly higher mesh density are required than, for example, meshes with millions of faces, then a consideration for using a data structure which has less of an impact of memory becomes apt. Table 3.4 demonstrates that the memory requirement for a face-based structure is at least 1.5 times less than that required to represent a mesh in an edge and halfedge manner. This is visually represented in Figure 3.17, which shows the ratio of memory consumption of the data structures against the face-based representation.

3.7.2 Runtime Results

The runtime tests involve measuring the amount of time to pre-process each mesh and the time necessary to simulate the electromagnetic behaviour on the geometry. This is divided into the amount of time to derive the connectivity of the mesh based

on the criteria dictated by each data structure, followed by the calculation and storage of the simulation parameters for each face. The amount of time to complete the running of the simulation for a given simulated time across meshes with varying degrees of refinement is investigated, and finally, the amount of time necessary to complete the simulation using a varying number of time steps, while keeping the mesh density constant.

Number of	**Mesh Preprocessing Time (ms)**		
Faces	**Face-Based**	**Edge-Based**	**Halfedge-Based**
303	7.567	14.142	18.163
628	16.110	23.783	38.680
1050	44.172	65.038	106.013
3410	203.105	299.049	487.451
6860	658.770	969.963	1581.049
17882	4047.451	5959.436	9713.882
55938	37774.515	55618.896	90658.836
296356	1020900.065	1435551.810	2497860.156

Table 3.5: Mesh processing time in milliseconds for the face-based, edge-based and halfedge-based data structures for varying degrees of mesh density. The timings relates to the amount of time to load the meshes and represent the connectivity using the individual data structures.

Table 3.5 shows the amount of time taken to pre-process the meshes in milliseconds. The halfedge data structure is shown to be the slowest at preprocessing the connectivity of a mesh. This is to be expected, with the additional requirements of deriving and storing the full connectivity of the mesh compared to the face-based and edge-based structures. Considerations for using the face-based structure begins to make sense with higher density meshes. From Table 3.5, the mesh constituting

55938 faces requires 37 seconds to store the mesh connectivity using the face-based paradigm, while the halfedge data structure requires 90 seconds. This point is certainly reinforced when dealing with meshes constituting nearly three hundred thousand faces. The time requirement to fulfil the preprocessing of the halfedge data structure, using a mesh consisting of 296,356 faces, is 27 minutes, compared to a modest 11 minutes for the face-based approach. The edge based structure in each case provides a compromise between the two. The extended preprocessing times can also be attributed to the size of the files initially used to load the meshes. As the number of mesh elements increases, the larger they become in memory and a longer amount of time is required to open and buffer these files by the software. To ensure a fair experiment, such that the results are not skewed, the same file type was used to represent each mesh as an input to each framework.

Number of	**UTLM Parameter Processing Time (ms)**		
Faces	**Face-Based**	**Edge-Based**	**Halfedge-Based**
303	4.389	2.979	1.828
628	10.583	6.388	3.993
1050	18.247	10.841	6.776
3410	43.134	26.406	16.504
6860	71.655	45.7568	28.598
17882	207.124	131.401	82.126
55938	698.406	462.930	289.334
296356	4234.6	2826.14	1681.280

Table 3.6: Time to process the UTLM parameters in milliseconds for the face-based, edge-based and halfedge-based data structures for varying degrees of mesh refinement. The time relates to the amount of time to traverse the entire mesh, calculate the parameters for each link line and store the parameters using the individual data structures.

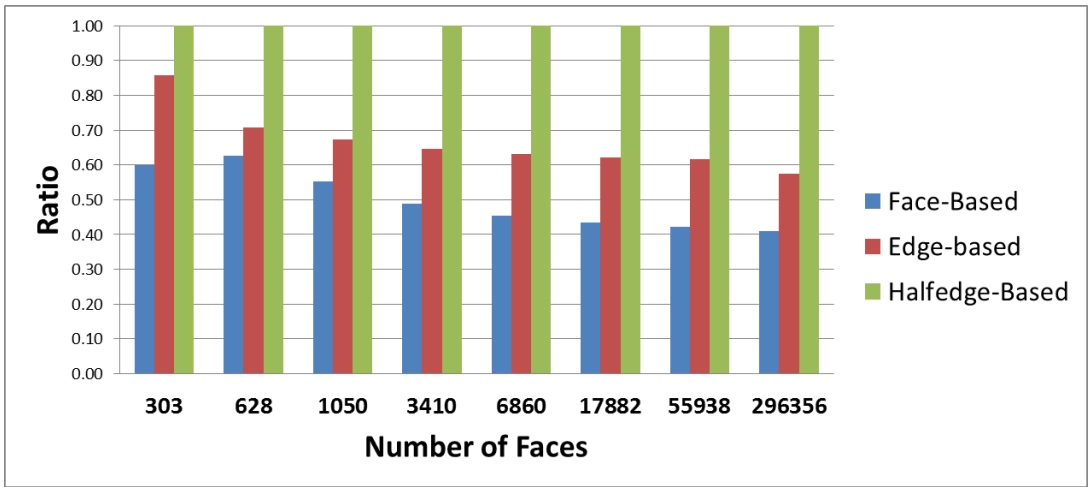

Figure 3.18: Ratio of the total preprocessing times between the face-based and edge-based data structures, compared with the halfedge strcuture as reference. The total preprocessing time is assembled from the loading and deriving of the mesh connectivity and subsequently deriving and storing the Untructured TLM parameters.

When it comes down to preprocessing the Unstructured TLM parameters, the mesh is traversed and the link line and stub admittances are stored at each face or edge depending on the data structure. The full knowledge of the mesh connectivity of the halfedge data structure allows the mesh to be now traversed much faster than the other data structures, which still require case distinctions in order to enumerate vertices. This is reflected in the experimental timings displayed in Table 3.6, which relate the amount of time to enumerate the entire mesh and compute the simulation parameters for each framework, using meshes ranging from 303 to 296356 faces. The halfedge data structure is able to make up some of the preprocessing time during this portion of the process for this reason, taking only 1.7 seconds to complete, compared with 4.2 seconds using the face-based structure and 2.8 seconds with the edge-based approach. Comparing these times with the times necessary to preprocess the mesh connectivity, from Table 3.5, it can be seen that the biggest impact on the preprocessing times is the loading of the mesh into each structure.

Figure 3.18 shows the times taken for the full preprocessing portion of the simulation as a ratio of the recorded time for the halfedge data structure against the other data structures for each mesh. This allows a visual overview of the impact that the mesh density has on the total preprocessing time for each framework. The reduced amount of connectivity information held by the face-based structure allows for total preprocessing times that are around 40% faster for the coarse meshes, and 60% faster for the very fine mesh, when compared to the halfedge representation. Deriving the connectivity information using the edge-based construction, provides an improvement of 15% for the coarsest mesh, and this improvement increases to 42% for the mesh with the highest element count, compared with the halfedge structure. It is apparent that preprocessing the connectivity of the halfedge structure is a more computationally intensive activity, not only in memory but also in runtime, when compared to the other implemented representations. The complexity of the implementation requires a greater effort for a computer to gain an understanding of the mesh connectivity. Even once this connectivity is understood, the framework is only required to traverse the mesh once in order to calculate the UTLM simulation parameters. The requirement of searching for adjacency information only once in the other frameworks does not provide the halfedge data structure any advantage as far as preprocessing the mesh is concerned.

Once the preprocessing of the mesh is complete, with all of the simulation parameters calculated, a simulation of the rectangular resonator is initiated on meshes ranging from a very coarse 303 faces to a fine mesh constituting 55938 faces. The time step for each mesh is calculated based on the resonator being air filled ($\varepsilon_r = 1$). Table 3.7 shows the time step, Δt, for each mesh with a different number of faces, N_F, and the impact this has on the number of time steps, N_T, necessary to complete the simulation for a total simulated time of 2×10^{-7} seconds. It can be seen that as the number of mesh elements increases, the time step for the simulation does not always become smaller; a result due to the mesh quality. A single link length

N_F	Δt (s)	N_T
303	1.286×10^{-12}	62100
628	1.813×10^{-12}	44100
1050	2.290×10^{-12}	34900
3410	2.495×10^{-12}	32058
6860	1.447×10^{-12}	55288
17882	9.519×10^{-13}	84045
55938	2.897×10^{-13}	276054

Table 3.7: Number of time steps, N_T required to complete the simulation for meshes of differing number of faces, N_F, with the maximum allowable time step for each mesh, Δt.

significantly shorter than the average link length across the mesh will dictate the time step, calculated from Equation (3.2), and will negatively impact the number of time steps necessary to run the simulation. As an example, it can noticed from Table 3.7 that although the mesh with 628 faces has more than twice the number of elements than the mesh constituting 303 faces, the maximum allowable time step is slightly larger for the mesh with a finer descritisation.

The runtime for each mesh is shown in Table 3.8. The times represent the amount of time, in seconds, to run the simulation for the predetermined simulated time. It can be seen that the halfedge data structure provides the fastest time to run the simulation, with full knowledge of the mesh connectivity clearly playing an advantage; a runtime of 323 minutes to complete the simulation using the mesh with 55938 faces resulted in a saving of 213.18 minutes compared to running the simulation using the edge-based structure, and reducing the simulation runtime by 439.28 minutes compared to the face-based structure.

Although this experiment demonstrates the superior runtime efficiency of the halfedge

Number of Faces	UTLM Simulation Runtime (s)		
	Face-Based	**Edge-Based**	**Halfedge-Based**
303	52.64	34.78	23.93
628	67.45	44.67	32.07
1050	97.62	64.37	42.63
3410	272.20	167.59	119.54
6860	886.09	645.24	420.99
17882	4083.96	2889.09	1783.39
55938	45737.04	32170.97	19380.10

Table 3.8: Runtimes for a 2D UTLM simulation of an air filled rectangular cavity, 0.1m in height and 0.2249m in width, for a range of mesh densities. The simulated time is 2×10^{-7} seconds

Number of Time Steps	UTLM Simulation RunTime (s)		
	Face-Based	**Edge-based**	**Halfedge-Based**
1,000	16.43	10.95	7.82
10,000	151.51	101.01	72.15
100,000	1721.82	1150.38	751.89
1,000,000	17994.02	12123.09	7624.58
10,000,000	186852.42	123049.16	75956.27
100,000,000	1769629.05	1161100.48	699458.12

Table 3.9: UTLM simulation runtimes for a 2D UTLM simulation of an air filled rectangular cavity, 0.1m in height and 0.2249m in width, descritised using 6860 faces, altering the number of iterations of the simulation by an order of magnitude from 1000 to 100000000.

paradigm when recursive algorithms are implemented on varying degrees of mesh density, this does not provide a clear comparison of how the runtime is affected if only the number of iterations are changed. A further investigation into how the number of time steps affects the runtime of the simulation is presented in Table 3.9.

The simulation is run on a single mesh, constituting 6860 faces, shown in Figure 3.16 c). Instead of simulating a fixed simulation time, as with the previous study, the number of iterations is altered by an order of magnitude for each simulation. Running the simulation in such a way removes the variability of the time step, which is now constant, and only the *number* of time steps used to complete the simulation is varied.

The results again show the halfedge data structure offering the fastest run times. For simulations requiring 100,000 time steps, the halfedge structure facilitates a saving in time of 74 minutes against the edge-based structure, and 172 minutes compared to the face-based representation. While this is still a substantial saving in time, the impact the data structure has on the runtime can truly be realised when considering a UTLM simulation requiring 100,000,000 iterations. Compared to the edge-based framework, a time saving of 128 hours, or five and a half days is achieved, while choosing the face-based representation over the halfedge paradigm can have a computational engineer waiting an additional 12 days for results using the same computational resources.

Figure 3.19 shows the factor of the runtime increases as a ratio against the halfedge data structure for each simulation. As the number of times the mesh is traversed increases, the greater the impact the data structure has on the runtime. It was shown in the runtime tests for the preprocessing of the UTLM parameters that traversing the mesh just once did not provide a significant advantage for the halfedge representation, despite the requirement of the face-based and edge based structures to search for the correct adjacency relationships. However, when the overhead of

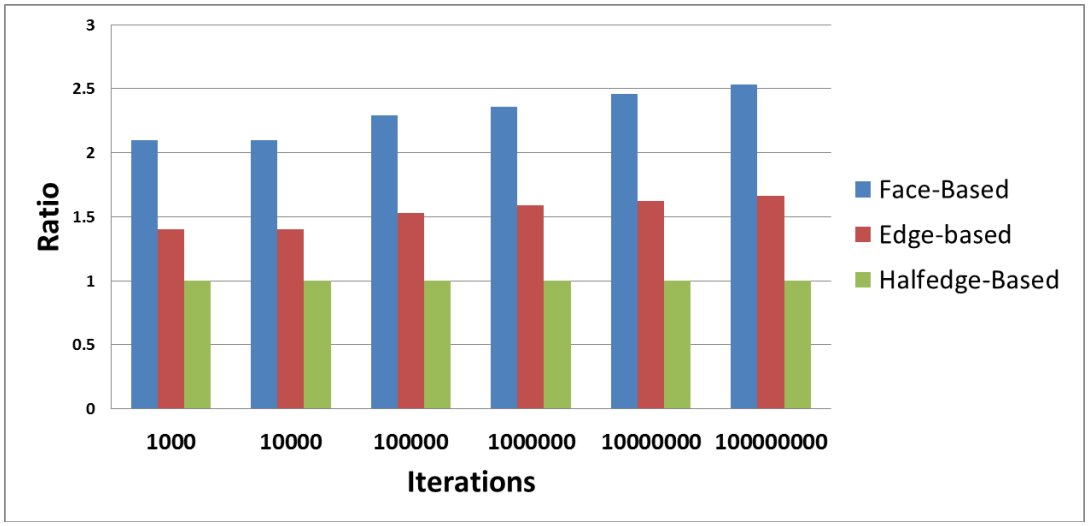

Figure 3.19: Ratio of the runtime between face-based and edge-based data structures compared with the halfedge data structure

case distinctions is compounded over many traversals, the halfedge implementation resulted in a significant increase in runtime efficiency. For 1000 traversals of the mesh, the halfedge approach is more than twice as fast as the face-based structure, and nearly 1.5 times faster than the edge-based implementation. For 100,000,000 iterations, full connectivity knowledge provided a speed up by a factor greater than 2.5 against the face-based implementation and 1.65 when compared to the edge-based. Despite taking significantly longer to preprocess the mesh and connectivity as shown in Figure 3.18, this is more than compensated for by the simulation runtime, making the halfedge data structure the most suitable for when runtime is given priority over memory consumption.

3.7.3 UTLM results

In order verify the UTLM algorithms presented in Section 3.6 and provide validation of the UTLM implementation, the resonant frequencies are extracted from the time domain sampled data using a Fast Fourier Transform (FFT). The sample point of

the time domain data is defined to be a single UTLM node which coincides with the centre point of the geometry, where the electric field component, E_z is measured at each time step. The accuracy is then compared with analytical solutions derived for the PEC rectangular resonator with dimensions 0.1m in height and 0.2249m in width, modelled for the simulation.

The analytical resonant modes TM $_{mn}$ are calculating using [3.19]:

$$f_c^{mn} = \frac{1}{2\sqrt{\mu\varepsilon}}\sqrt{\left(\frac{m}{a}\right)^2 + \left(\frac{n}{b}\right)^2} \tag{3.27}$$

where a and b are the resonator width and height respectively, ε is the material permittivity and μ is the material permeability. It should also be noted that $m \neq 0$ and $n \neq 0$ for TM modes [3.20].

Table 3.10 shows simulated resonant frequencies and the percentage relative error against the analytic result for the first 6 modes of resonance for each descritisation of the geometry. It is observed that as the mesh refinement increases, so too does convergence with the analytical solution f_{exact}. The relative error induced for the coarsest mesh, with 303 TLM nodes, was 0.71% for TM_{11} mode and 0.89% for the higher frequency mode TM_{32}. The simulation utilizing the finest mesh, consisting of 55938 TLM nodes, reduced this error to 0.11% for the TM_{11} mode and 0.2% for the TM_{32} mode, demonstrating good agreement with the analytically derived solutions.

Figure 3.20 summarises this trend in the plot of the relative error for the first three modes, TM_{11}, TM_{21} and TM_{31}, as the meshing of the problem space is refined. As the density of sample points increases, so too do the accuracies of the sampled fields at the observation point.

Figure 3.21 and Figure 3.22 show the plots of the FFT of the sampled time domain data, for each mesh. Figure 3.21 a), b), c) and d) show the peaks of the excited modes for meshes with 303, 628, and 1050 respectively, with Figure 3.22 displaying

Number of Faces		Resonant Frequencies f (GHz)					
		TM_{11}	TM_{21}	TM_{31}	TM_{12}	TM_{22}	TM_{32}
	$\boldsymbol{f_{exact}}$	1.64291	2.01391	2.51337	3.07243	3.28581	3.61355
303	f_{TLM}	1.63125	1.99838	2.48625	3.04475	3.24550	3.58123
	Relative Error %	**0.71002**	**0.77114**	**1.07903**	**0.90092**	**1.22679**	**0.89455**
628	f_{TLM}	1.63875	2.00375	2.49375	3.04750	3.26500	3.63746
	Relative Error %	**0.25327**	**0.50454**	**0.78070**	**0.81147**	**0.63339**	**0.66179**
1050	f_{TLM}	1.63957	2.00432	2.49595	3.05967	3.27195	3.60000
	Relative Error %	**0.20324**	**0.47609**	**0.69309**	**0.41521**	**0.42169**	**0.37505**
3410	f_{TLM}	1.64001	2.00501	2.49751	3.06251	3.27876	3.59997
	Relative Error %	**0.17682**	**0.44208**	**0.63110**	**0.32300**	**0.21453**	**0.37575**
6860	f_{TLM}	1.64050	2.00600	2.49885	3.06876	3.27801	3.60247
	Relative Error %	**0.14651**	**0.39257**	**0.57767**	**0.11945**	**0.23751**	**0.30665**
17882	f_{TLM}	1.64040	2.00757	2.49974	3.06874	3.27874	3.59974
	Relative Error %	**0.15278**	**0.31481**	**0.54218**	**0.12007**	**0.21514**	**0.38217**
55938	f_{TLM}	1.64097	2.00902	2.50000	3.06333	3.27974	3.60630
	Relative Error %	**0.11796**	**0.24281**	**0.53194**	**0.29625**	**0.18467**	**0.20072**

Table 3.10: The relative error in resonant frequencies between those obtained via UTLM simulation, f_{TLM} and the analytical solution, f_{exact} for the first 6 TM modes of a rectangular resonator 0.1m in height and 0.2249m in width.

the frequency spectra for 6860, 17882 and 55938 in e) f) and g) respectively. The arrows on each plot are positioned at the analytically obtained frequencies for the first 6 modes. The graphs demonstrate how the peaks in frequency better align with the exact solutions when greater mesh densities are used. A fast convergence to the first 3 modes as the mesh density increases to 1050 elements in d) is observed. The higher frequency modes converge better when slightly higher density meshes are used, shown in Figure 3.22 e) f). However no significant improvement in accuracy is observed when the mesh density is increased from Figure 3.22 f) to g), which consists of 3 times the number of faces; this demonstrates a rate of diminishing returns in accuracy as the mesh is further refined.

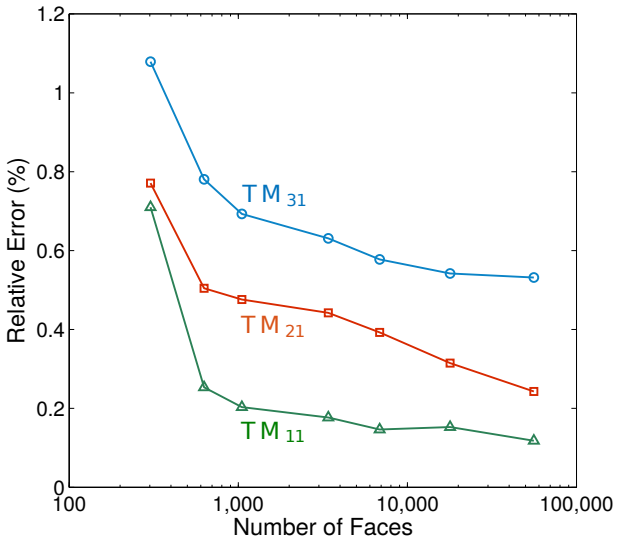

Figure 3.20: Relative error for the different resonant TM modes of an air filled PEC rectangular resonator against the number of faces constituting a mesh

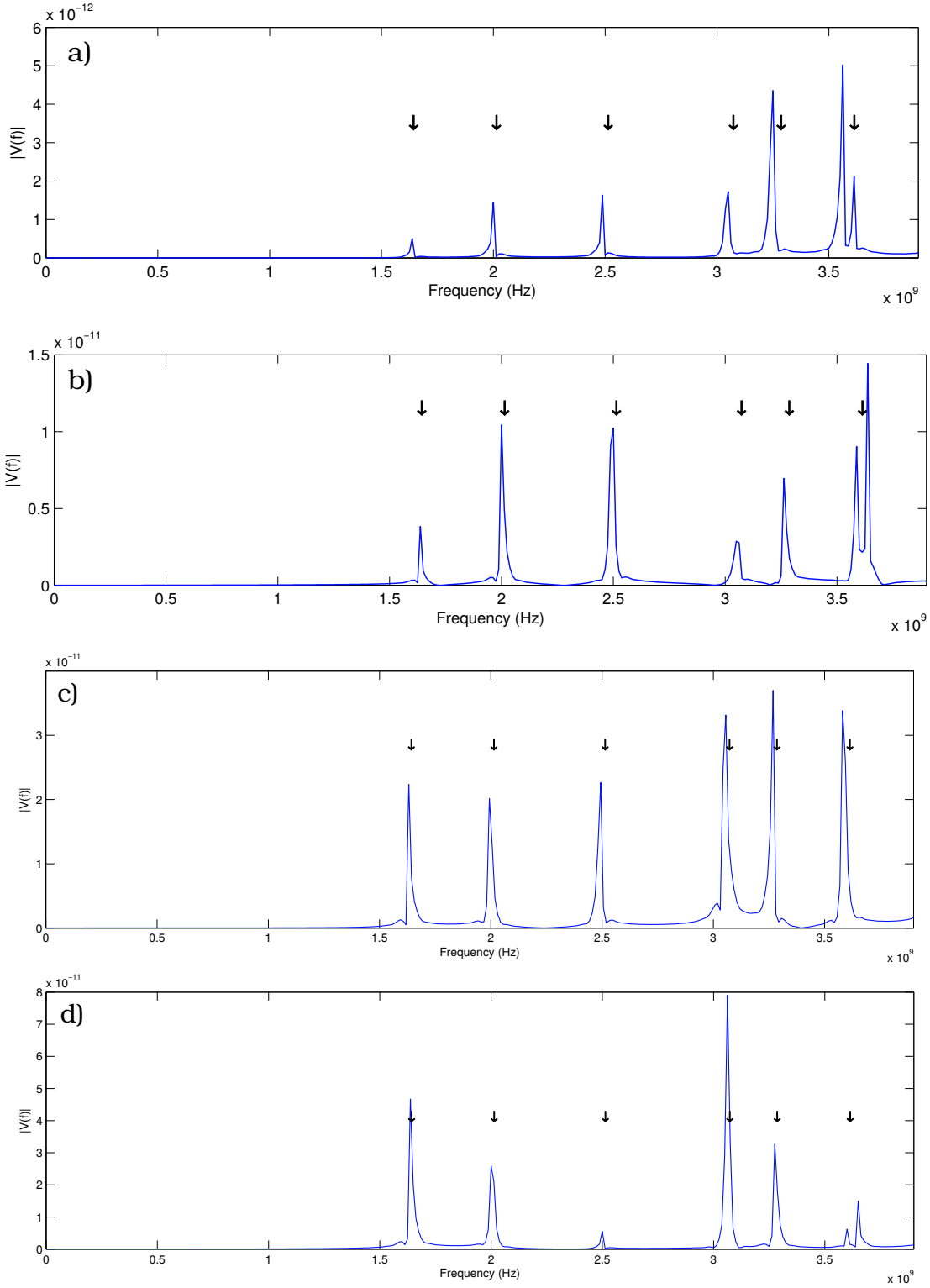

Figure 3.21: Frequency spectra obtained through UTLM simulations of the rectangular resonator. Arrows on each plot represent the analytic solutions for the first frequencies TM_{11}, TM_{21}, TM_{31}, TM_{12}, TM_{22}, TM_{32} respectively. Plots are for simulations using a) 303, b) 628, c) 1050 and d) 3410 mesh faces.

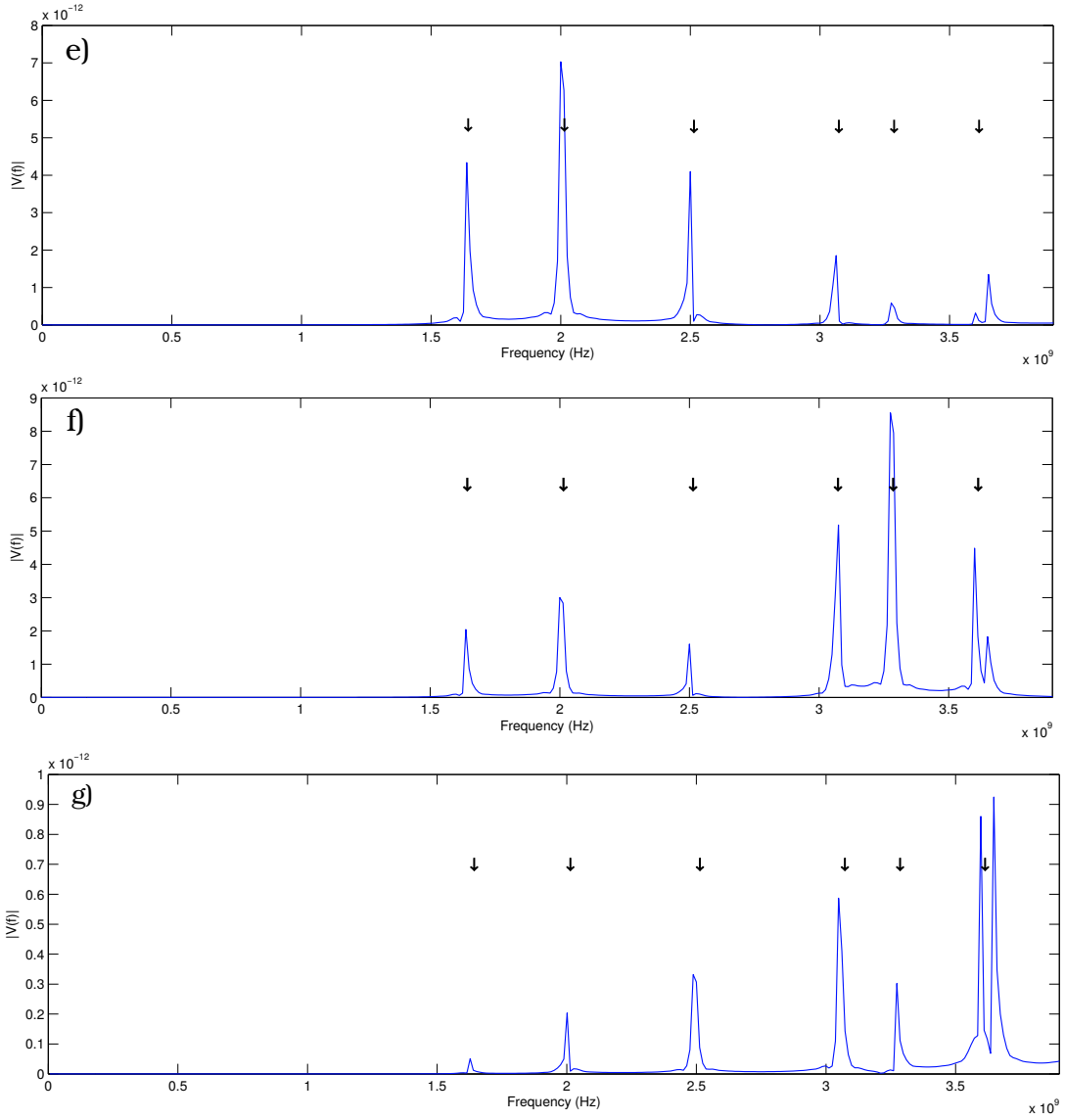

Figure 3.22: FFT plots of the resonant frequencies obtained through UTLM simulations of the rectangular resonator. The arrows on each plot represent the analytic solutions for the resonant frequencies TM_{11}, TM_{21}, TM_{31}, TM_{12}, TM_{22}, TM_{32} respectively. Plots are for simulations for e) 6860, f) 17882 and g) 55938 mesh faces.

3.8 Summary

This chapter introduced methods for representing a descritised geometry and its connectivity in computer memory, a process fundamental to TLM simulations. General considerations for choosing a specific data structure were provided, pertaining to the topological requirements of the mesh, as well as algorithmic considerations of the simulation, before defining the specific considerations the UTLM method. A method for estimating the memory requirements for a mesh was subsequently provided through the use of the Euler-Poincaré characteristic, which provides a relationship between the number of vertices, edges and faces of a mesh.

Three popular data structures were then introduced, namely, the face-based, edge-based and halfedge-based data structures. These methods facilitate the description of the connectivity of a mesh, each having differing impacts on the memory consumption necessary to represent the topology, and the impact on the runtime when executing algorithms on the mesh. The more a data structure understands the connectivity of a mesh, in terms of adjacency information, the more memory is required to store the mesh. Consequently, the greater the understanding of adjacency information, the faster the mesh can be navigated leading to more efficient execution times of algorithms.

How UTLM parameters and algorithms are represented on a mesh were discussed, applying the theory presented in Chapter 2 to a simulation. This was then used to provide a case study of a rectangular resonator, allowing for a comparison of memory and runtime consumption of the various data structures introduced. It was shown that the face-based data structure consumed the least amount of memory to represent and store a mesh, compared to the edge-based and halfedge data structures, with the halfedge based data structure exhibiting the highest memory foot print. This higher memory consumption of the halfedge data structure facilitated a faster

runtime of the UTLM simulations due to the requirement of fewer case distinctions in order to traverse the mesh as with the other data structures. It became evident that each data structure allows a Computational Engineer to tailor the use of the available resources based on the computer system being used. With an abundance of memory, the topology of very dense meshes can be represented with full connectivity information in a halfedge structure which in turn provides a faster simulation runtime. Counter to this scenario, the runtime of the simulation can be sacrificed when memory is more scarce, through the use of a face-based or edge based representation. However, even the 2D meshes with a very high face count, when represented with any of the structures, did not tax the memory resources available using a modern day desktop computer.

A comparison of the UTLM simulation results to analytical solutions of resonant modes of the rectangular resonator provided validation of the UTLM algorithms. It was shown that increasing the mesh density demonstrated good convergence with analytical solutions, with minimal relative error between the simulation results and the analytical ones.

In the next chapter, an investigation into the feasibility of further reducing the necessary computational effort of UTLM simulations, for low frequency applications, is presented in the form of surface parameterisation.

References

[3.1] B. L. M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, *Polygon Mesh Processing*. CRC Press, 2010.

[3.2] M. W. Beall and M. S. Shephard, "A General Topology-Based Mesh Data Structure," *Int. J. Numer. Methods Eng.*, vol. 40, no. 9, pp. 1573–1596, 1997.

[3.3] J. Bloomenthal and K. Ferguson, "Polygonization of non-manifold implicit surfaces," *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '95*, pp. 309–316, 1995.

[3.4] L. Kettner, "Using Generic Programming for Designing a Data Structure for Polyhedral Surfaces," *Comput. Geom.*, vol. 13, no. 1, pp. 65–90, 1999.

[3.5] L. Kobbelt and M. Botsch, "Freeform Shape Representations for Efficient Geometry processing," *Comput. Graph. Appl. 2003. Proceedings. 11th Pacific Conf.*, pp. 111–115, 2003.

[3.6] H. Coxeter, *Introduction to Geometry*, 2nd ed. Wiley, 1989.

[3.7] R. E. Tarjan, *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, 1983.

[3.8] 3D-Systems, "Stereolithography Interface Specification," Tech. Rep., 1989.

[3.9] "web3D Consotium Open Standards for Real Time 3D Communication." [Online]. Available: http://www.web3d.org/standards

[3.10] W. R. Franklin, "Polygon properties Calculated From The Vertex Neighborhoods," *Annu. Symp. Comput. Geom.*, pp. 110—-118, 1987.

[3.11] B. G. Baumgart, *Winged Edge Polyhedron Representation.* Stanford : Stanford University. Computer Science Department, 1972.

[3.12] L. Guibas and J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi," pp. 74–123, 1985.

[3.13] J. O'Rourke and S. L. Devadoss, *Discrete and Computational Geometry.* Princeton University Press, 2011.

[3.14] D. Muller and F. Preparata, "Finding The Intersection of Two Convex Polyhedra," *Theor. Comput. Sci.*, vol. 7, no. 2, pp. 217–236, 1978.

[3.15] L. Kettner, "Using Generic Programming for Designing a Data Structure for Polyhedral Surfaces," *Comput. Geom.*, vol. 13, no. 1, pp. 65–90, 1999.

[3.16] H. Brönnimann, "Designing and Implementing a General Purpose Halfedge Data Structure," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2141 LNCS, pp. 51–66, 2001.

[3.17] C. Christopoulos, *The Transmission-line Modeling (TLM) Method in Electromagnetics.* Morgan & Claypool Publishers, 2006.

[3.18] J. R. Shewchuk, "Reprint of: Delaunay Refinement Algorithms for Triangular Mesh Generation," *Comput. Geom. Theory Appl.*, vol. 22, pp. 21–74, 2014.

[3.19] F. Iskander, *Electromagnetic Fields and Waves.* Waveland Press. Inc., 2000.

[3.20] C. A. Balanis, *Advanced Engineering Electromagnetics*, 2nd ed. Wiley, 2012.

Chapter 4

Mesh Parameterisation

4.1 Introduction

The previous chapter demonstrated how the choice of the underlying data structures used to represent problem spaces impacts on the runtime or memory consumption of a simulation. When considering thin, flexible curved geometries in the 3D domain, reductions in memory and runtime may be possible when simulating low frequency electromagnetic behaviour on these surfaces. By providing a one-to-one mapping of the surface in the 3D domain to a 2D flat plane, the necessity of meshing the full 3D problem space may be negated. The goal of this chapter is to provide a background to these one-to-one mappings, known as mesh parameterisation. In this chapter the various parameterisation methods are reviewed, summarising the main ideas of each technique which have been implemented as part of this investigation.

Parameterising a 3D mesh amounts to computing a correspondence between a discrete surface patch in 3D space and a planar region in the 2D domain. In practice, this piecewise linear mapping entails assigning each mesh vertex a pair of new coordinates *(u,v)* referring to its position on the planar region. The one-to-one mapping provides a flat parametric surface, facilitating the performance of any complex op-

Figure 4.1: Parameterisation of the Stanford bunny for the purposes of texture mapping. Image taken from [4.2]

eration directly in the flat domain as opposed to the curved surface [4.1].

Parametrising surfaces from the 3D domain to a flat 2D plane has been used for centuries; it is a practice used by cartographers in order to represent our spherical world as a flat map [4.3]. However, the driving force for the development of the first parameterisation methods for discrete surfaces has been the computer graphics industry. Parameterising a 3D model to a flat plane allows textures to be easily applied in 2D before being mapped back to the original surface, enhancing the visual richness of polygonal models [4.4] [4.5] [4.2]. Figure 4.1 provides an example of a texture mapping application. This Stanford bunny is first segmented, and then parameterised to the 2D domain. The art work is applied on this flat surface, before being mapped back to the original geometry. For over a decade, mesh parametrisation has become a ubiquitous tool, extending its application to aid processes such as remeshing [4.6] [4.1], mesh subdivision [4.7] and repairing CAD and 3D scan data [4.8] [4.9].

A mapping is piecewise linear, associating each triangle of the original mesh with a triangle in the flat domain. An important goal of parameterisation is to obtain *bijective* (invertible) maps, where each vertex constituting the surface in the pa-

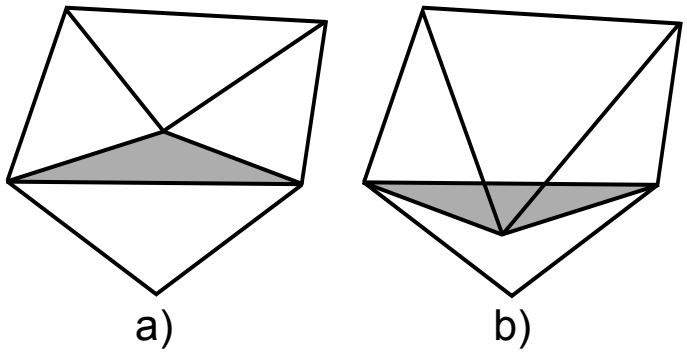

Figure 4.2: Bijectivity of a parameterisation. a) all normal directions of the surface are in the same direction. b) The grey triangle in a) has been flipped, such that its normal direction now points in the opposite direction

rameter domain corresponds to exactly one point of the mesh in the 3D domain. Enforcing this condition amounts to ensuring that there are no overlaps of elements in the parameter domain as a result of the parameterisation. Globally bijective parameterisations require that the problem boundary does not self intersect. Local bijectivity requires a map of any sufficiently small region of the mesh to be bijective. This bijectivity condition is violated if the mappings of adjacent triangles intersect. To illustrate this, Figure 4.2 shows two parameterisations. In part a) the triangulation is valid, such that all of the triangle normals are in the same direction and is bijective. In b) the parameterisation is locally non-bijective, where the normal of the highlighted triangle in a) is inverted or *flipped* with respect to the other triangle normals. Some mesh parameterisation techniques, under certain conditions may result in self intersecting boundaries, and triangle flips, which prevent the parameterisation from being bijective. This will also be explored in this chapter.

The ideal mapping is one that is isometric, preserving all surface metrics, that is, area and angle, through the parameterisation. However, this is not possible apart from in special cases. Only developable surfaces, that is, surfaces with a zero Gaussian curvature, can be parameterised isometrically. Examples are cylindrical or conical surfaces [4.10].

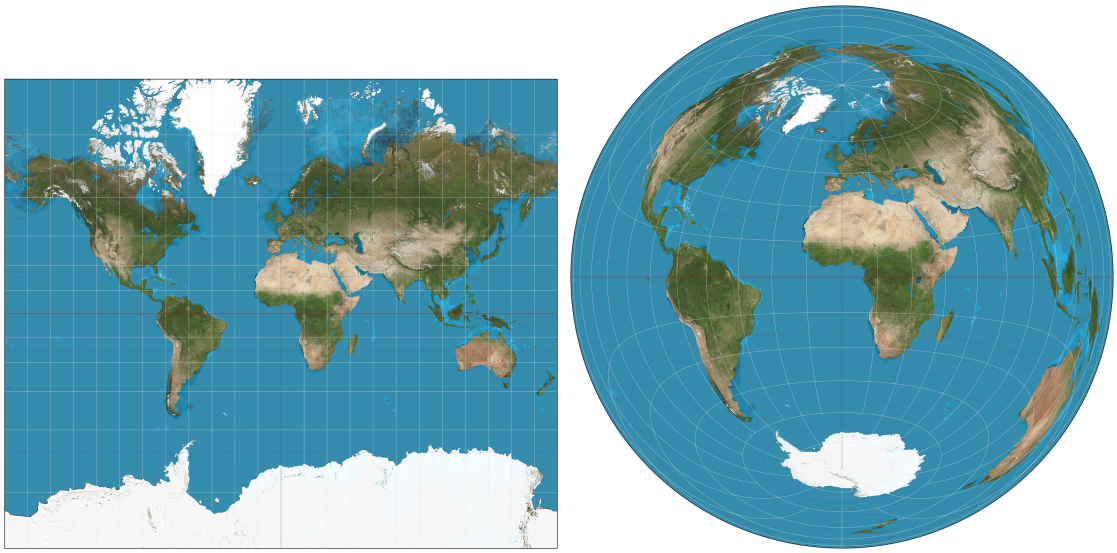

Figure 4.3: Parameterising the globe: The Mercator projection (left) preserves angles, but not area. The Lambert azimuthal projection (right) accurately represents area but does not preserve angles.[1]

Like when attempting to flatten the peel of an orange onto a table, we cannot project a sphere onto the plane without distorting the surface, and therefore certain compromises must be made [4.11]. This explains why cartographers have so many different projections of the globe. Figure 4.3 shows two well known parameterisations of our world. The Mercator projection, presented on the left, was first introduced by the Flemish geographer and cartographer Gerardus Mercator in 1569. While the linear scale is equal in all directions around any given point, thus preserving angles and the shapes of the countries, this mapping distorts the size of objects as the latitude increases from the equator to the poles. This results in Greenland and Antarctica (both coloured white in both images) appearing much larger relative to land masses near the equator than they actually are. The mapping shown to the right of Figure 4.3 shows the Lambert azimuthal projection, attributed to the Swiss mathematician, Johann Heinrich Lambert in 1772. This particular mapping from a sphere to a disk (that is, a region bounded by a circle) accurately preserves area in all regions of the sphere, however it does not accurately represent angles [4.12].

Parametrisations will almost always introduce distortion in the form of a change in either angles (conformal mappings) or area (authalic mappings), and a good parametrisation for an application is one that minimizes the effect on these metric properties in some sense. Many different ways have been proposed to achieve this, all varying only by the metric of distortion considered and the minimisation processes used [4.13].

The majority of the parameterisation techniques developed in the computer graphics industry are dedicated to preserving angles. Though authalic parameterisations are achievable, they are not very useful by themselves, as they allow extreme angular and linear distortion [4.14]. An abstract demonstration of what area preservation really implies is shown in Figure 4.4. Figure 4.4 a) shows a spherical surface descritised using a curvilinear equilateral triangulation. Area can be preserved on the sphere by swirling its surface around in a fluid like motion as shown conceptually in Figure 4.4 b). From the point of view of general geometry processing, and more specifically for our purposes of electromagnetic simulation, where the stability of a simulation is dependent on the quality of the triangulation, this type of motion is undesirable and leads to arbitrarily bad degeneration of the shape of mesh elements. This, in turn, can lead to a distortion of local attributes and data associated with the surface, such as the link lines which define a UTLM network, and as such, the material parameters associated with each link length. In contrast, angle-preserving motions (Figure 4.4 c)) are more *rigid* and hence better preserve the features we are interested in; the quality of the triangulation, with regards to the shape of individual triangles, and consequently the simulation parameters of the surface. In order to find a well-behaved authalic mapping it is often necessary to combine area-preservation with some minimization of angular distortion [4.1].

It should also be noted that in the discrete setting, conformal maps are sometimes

[1]Images By Strebe, via Wikimedia Commons (http://creativecommons.org/licenses/by-sa/3.0)]

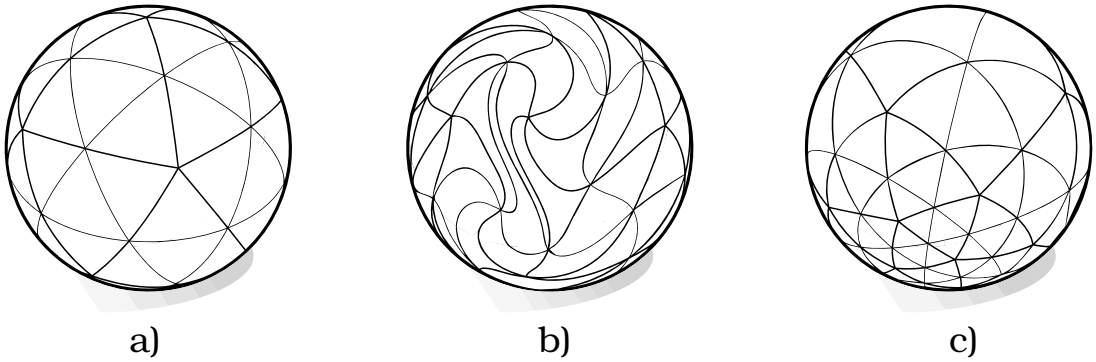

Figure 4.4: An impression of what an authalic mapping can imply. a) Original triangulation of a 3D spherical surface using curvilinear equilateral triangles, b) area preserving and c) ab angle preserving parameterisation

referred to as harmonic maps; the term is borrowed from the differential geometry of smooth surfaces, though the two expressions are not tantamount. It has been shown that Bernhard Riemanns mapping theorem, derived in 1851 in his PhD thesis [4.15] [4.16], which guarantees that conformal maps (with zero angular distortion) exist for any planar parametrisation, does not persist for discrete surfaces. Meshes are an approximation of smooth surfaces, so although it can be argued that it is possible to parametrise the mesh with little angular distortion, the sum of the angles about an interior mesh vertex in the 3D domain can deviate from 2π, while the angles about an interior vertex in the 2D plane will always sum to 2π. Some methods, applied to meshes with a progressive increase in element density, will converge in the limit to a smooth conformal map [4.17]. It is therefore necessary to investigate the impact of the planar parameterisations across varying degrees of mesh refinement.

What follows in the remainder of this chapter is an overview of the existing parameterisation techniques. These techniques are divided into linear and non-linear methods. First a basic parameterisation concept is introduced in Section 4.2.1, which is based on the minimisation of the energy of a spring. This initial methodology forms the foundation for the early subsequent parameterisation frameworks, which attempt to minimise angular distortion through the choice of barycentric co-

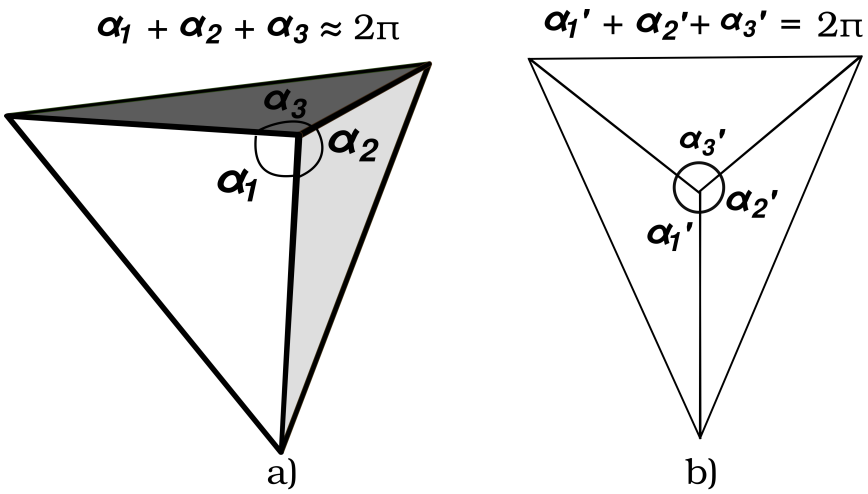

Figure 4.5: Angles about a vertex in the 3D do not always sum to 2π (a), where as angles about a vertex in the 2D plane always do (b).

ordinates that are used to effectively define spring constants. Different methods of defining barycentric coordinates exist [4.18] [4.19] [4.20]. Their definitions are discussed, in addition to presenting their influence on the mapping and how they impact the bijectivity of the parameterisation, in Section 4.3. These techniques offer a linear solution, however they require that the boundary is predefined in the parameter domain, and only later were free boundary techniques derived. This leads the discussion in Section 4.5 to free boundary alternatives, such as the Least Squares Conformal Map (LSCM) [4.2], in addition to non-linear techniques which aim to provide as isometric as possible parameterisations, including the Most Isometric ParameterisationS (MIPS) method [4.21] and Angle Based Flattening (ABF) [4.22]. An overview is provided, before moving to an experimental comparison of the different methods, which follows in Chapter 5.

4.2 Linear Methods

In this section parameterisations that rely on the solution of a linear set of equations are presented. In order to avoid confusion when referring to points in the 3D and 2D domain, a brief description of the notation is provided here. Points in 3D space will be denoted using $\boldsymbol{p} = (x, y, z)$ and points in 2D parameter space will be referred to using $\boldsymbol{u} = (u, v)$. A triangular mesh will be denoted by the capital letter T, which describes the union of triangles t such that $T = t_1, ..., t_n$, and a set of vertices $V = \boldsymbol{p_1}, ..., \boldsymbol{p_{n+b}}$ in the 3D domain and $V = \boldsymbol{u_1}, ..., \boldsymbol{u_{n+b}}$ in 2D space. More specifically, the set of vertices consists of n interior vertices $V_I = \boldsymbol{p_1}, ..., \boldsymbol{p_n}$ and b boundary vertices $V_B = \boldsymbol{p_{n+1}, ..., p_{n+b}}$. Two distinct vertices $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ are neighbours if they are connected by an edge and N_i will refer to set of indices of all neighbours of $\boldsymbol{p_i}$

4.2.1 Spring Model For Mesh Parameterisation

A simple idea for constructing a parameterisation of a triangular mesh is based on a physical spring model [4.11]. Consider a sequence of n vertices, connected by edges, where the edges are represented by springs as depicted in Figure 4.6. If the two end points of this sequence of vertices are pulled the springs will relax to a stable equilibrium of minimal energy, where we assume each spring to be ideal in the sense that the rest length is zero and the new position of the vertices can be taken as the parameter points.

Conceptually, The potential energy of a spring is $E = \frac{1}{2}DS^2$, where D is the spring constant and S is the length of the spring. The parameter points $\boldsymbol{u_i} = (u_i, v_i)), i = n+1, ..., n+b$ for the boundary vertices $\boldsymbol{p_i} \in V_B$, of the mesh are specified in some way (Section 4.4), where n is the number of interior vertices, b is the number of

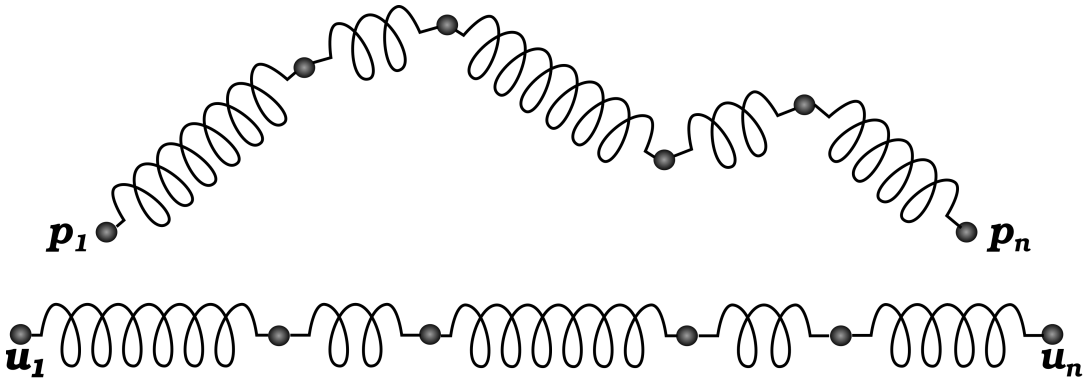

Figure 4.6: Conceptual parameterisation of a sequence of n vertices with the spring model

boundary vertices, and V_B is the set of boundary vertices in the 3D domain. The overall spring energy F is then minimised [4.23]:

$$F = \frac{1}{2}\sum_{i=1}^{n+b}\sum_{j\in N_i}\frac{1}{2}D_{ij}\|\boldsymbol{u_i} - \boldsymbol{u_j}\|^2 \tag{4.1}$$

where $D_{ij} = D_{ji}$ is the spring constant between 3D vertices $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ with respect to the unknown 2D parameter positions $\boldsymbol{u_i} = (u_i, v_i)$ for interior points. $j \in N_i$ is the index of a neighbour vertex to p_i in the set of all neighbouring vertices N_i connected by an edge. The additional factor of $\frac{1}{2}$ appears in equation (4.1) because summing the edges in this way counts every edge twice. Differentiating this energy F with respect to $\boldsymbol{u}_i$ yields [4.23]:

$$\frac{\partial F}{\partial \boldsymbol{u_i}} = \sum_{j\in N_i} D_{ij}(\boldsymbol{u_i} - \boldsymbol{u_j}) \tag{4.2}$$

The minimum of F is obtained if

$$\sum_{j\in N_i} D_{ij}\boldsymbol{u_i} = \sum_{j\in N_i} D_{ij}\boldsymbol{u_j} \tag{4.3}$$

holds for all $i = 1, ..., n$ [4.23]. This is equivalent to saying that each interior parameter point $\boldsymbol{u_i}$ is an affine combination of its neighbours,

$$\boldsymbol{u_i} = \sum_{j\in Ni} \lambda_{ij}\boldsymbol{u_j}, \tag{4.4}$$

where the normalised coefficients

$$\lambda_{ij} = \frac{D_{ij}}{\sum\limits_{j \in N_i} D_{ik}}, \tag{4.5}$$

sum to 1.

Separating the parameter points for the interior and the boundary vertices in the sum on the right hand side of equation (4.4) gives [4.23]

$$\boldsymbol{u_i} - \sum_{j \in N_i, j \leq n} \lambda_{ij} \boldsymbol{u_j} = \sum_{j \in N_i, j > n} \lambda_{ij} \boldsymbol{u_j} \tag{4.6}$$

From here, computing the coordinates u_i and v_i of the interior parameter points $\boldsymbol{u_i}$ requires solving the linear systems

$$AU = \bar{U} \quad \text{and} \quad AV = \bar{V}, \tag{4.7}$$

where $U = (u_1, ..., u_n)$ and $V = (v_1, ..., v_n)$ are the column vectors of unknown coordinates, $\bar{U} = (\bar{u}_1, ..., \bar{u}_n$ and $\bar{V} = (\bar{v}_1, ..., \bar{v}_n)$ are the column vectors with coefficients

$$\bar{u}_i = \sum_{j \in N_i, j > n} \lambda_{ij} u_j, \tag{4.8}$$

and

$$\bar{v}_i = \sum_{j \in N_i, j > n} \lambda_{ij} v_j \tag{4.9}$$

and $A = (a_{ij})_{i,j=1,...,n}$ is the $n \times n$ matrix with elements

$$a_{ij} = \begin{cases} 1 & \text{if } i = j, \\ \lambda_{ij} & \text{if } j \in N_i, \\ 0 & \textit{otherwise.} \end{cases} \tag{4.10}$$

The sparse linear systems in equation 4.7 can be efficiently solved using iterative methods, with state-of-the-art, open source libraries such as TAUCS [4.24], OpenNL [4.25], and SuiteSparse [4.26].

4.3 Fixed Boundary Parameterisation

The spring model discussed in the previous section provides a base for the development of discrete surface parameterisations. The linear approaches that follow all have the following strategy in common:

1) find a parameterisation for the boundary points and
2) minimize an edge-based energy functional (equation (4.1)) to determine the parameterisation for the internal points such that the minimum energy is found.

The question remains how to choose the spring constants D_{ij} in the spring model, or more generally the normalised coefficients λ_{ij} in equation 4.6, for each edge. It has been shown that bijectivity of the parameterisation can be achieved if the parameter points for the boundary vertices are set correctly and the coefficients are chosen such that [4.27]:

$$\boldsymbol{p_i} = \sum_{j \in N_i} \lambda_{ij} \boldsymbol{p_j} \quad \text{and} \quad \sum_{j \in N_i} \lambda_{ij} = 1, \tag{4.11}$$

for all interior vertices. The values of λ_{ij} with both of these properties are known as the barycentric coordinates of $\boldsymbol{p_i}$ with respect to its neighbours $\boldsymbol{p_j}$, $j \in N_i$, and can be computed by the normalisation:

$$\lambda_{ij} = \frac{\omega_{ij}}{\sum\limits_{k \in N_i} \omega_{ik}}, \tag{4.12}$$

which is synonymous with equation (4.5). ω_{ij} is a weight applied to the edge connection $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ and ω_{ik} are weights for edges providing the connectivity between $\boldsymbol{p_i}$ and $\boldsymbol{p_k}$; k represents the index of vertices in the set of connected vertices N_i. There are many ways of defining barycentric coordinates, based on the choice of defining ω_{ij}. The most popular choices pertaining to mesh parameterisation are reviewed in this section, which are Uniform weights, Wachspress coordinates, Discrete Harmonic Coordinates and Mean Value coordinates.

One of the oldest methods referenced in the context of mesh parameterisation is the graph embedding method of Tutte [4.28], who used uniform unit edge weights, setting $\omega_{ij} = 1$ if (i, j) is an edge in the mesh. This means for each interior vertex $\boldsymbol{p_i}$, λ_{ij} is equal to $\frac{1}{d_i}$ where d is the valency of $\boldsymbol{p_i}$, i.e the number of edges emanating from $\boldsymbol{p_i}$. This parameterisation is provably bijective [4.29]. Figure 4.7 shows an example of a Uniform weight parameterisation. An open hemispherical surface in the 3D domain is approximated using 658 triangles, and subsequently parameterised to a circular and square domain, both predefined in 2D space, before the parameterisation of the interior vertices.

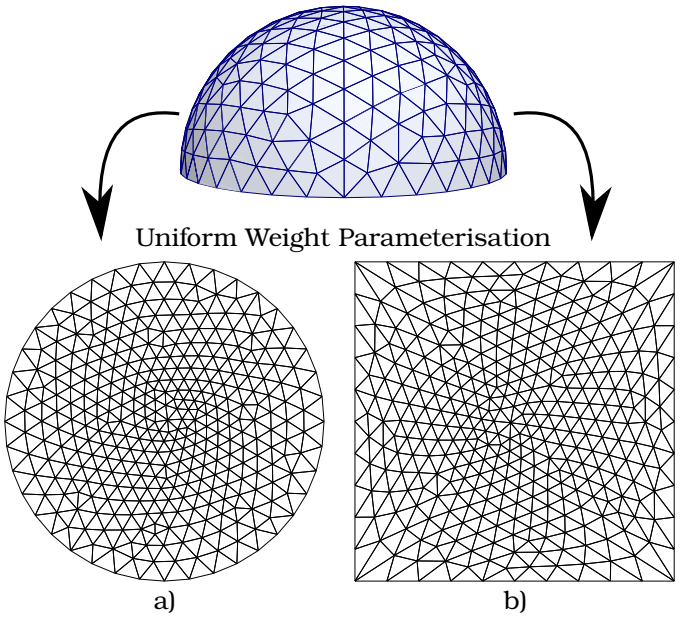

Figure 4.7: Example of a parameterised hemisphere, meshed using 658 faces, using Uniform weight [4.28]. a) this the parameterisation using a circular boundary, and b) uses a square boundary.

Many of the mesh parameterisation techniques are focused on minimising the angular distortion. This is due to many of the current applications of surface parametrisation requiring the maintenance of good quality triangle shapes. Remeshing, for example, replaces one triangulation with another of better quality, typically by mapping a regular triangulation of the domain onto the mesh [4.6]. More importantly for

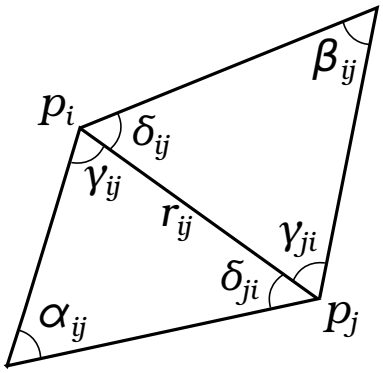

Figure 4.8: Notation for the formulation of barycentric coordinates. The triangles are in the 3D domain, with r_{ij} representing the length of the edge $\|\boldsymbol{p_i} - \boldsymbol{p_j}\|$

our application of TLM and numerical simulations, the accuracy and stability are adversely affected by small angles, which induce sliver elements in the mesh [4.30], as discussed in Chapter 2.

Angle-preserving parametrisation is efficient to compute and, providing that the stretch, of the resultant mesh is minimal, which is often the case when the original geometry is not too far from developable [4.31]. Tutte's method formed the foundation for the approach of several parameterisation techniques [4.19] [4.32] [4.20]. The differences between them amount to the choice of ω_{ij}.

Wachspress coordinates are the earliest generalisation of barycentric coordinates and go back to the 1970's, where Wachspress suggested to set [4.33]

$$\omega_{ij} = \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{r_{ij}^2}, \tag{4.13}$$

where r_{ij} defines the length of the edge $\|\boldsymbol{p_i} - \boldsymbol{p_j}\|$ and the angles defined in Figure 4.8. While Wachspress' primary application was the Finite Element method, these coordinates were first used by Desbrun *et al.* in [4.1] for the purposes of mesh parameterisation.

Discrete Harmonic coordinates, also known as cotangent weights [4.34] [4.19], are perhaps the most widely known in the graphics community, and define another type

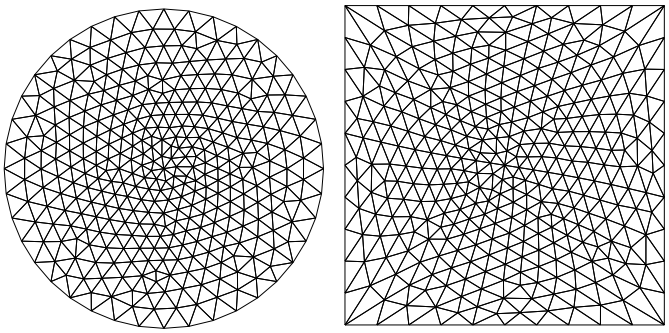

Figure 4.9: Example of a Discrete Harmonic Parameterisation of a hemisphere meshed using 658 faces (shown in Figure 4.7), meshed using 658 faces, parameterised using cotangent weights, into a circular and square parameter domain.

of barycentric coordinates, defined using:

$$\omega_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}, \tag{4.14}$$

where α_{ij} and β_{ij} pertain to the opposing angles to the common edge r_{ij} in Figure 4.8. These were first used to compute discrete minimal surfaces [4.34]. However, in the context of mesh parameterisation, these coordinates were first used by Eck *et al.* in [4.19], which serve to reduce angular distortion of the parameterisation.

A major drawback of Discrete Harmonic Parameterisations, however, is that if the original mesh contains obtuse angles, the weights can be negative. This can be seen by applying the trigonometric identity:

$$\cot \alpha + \cot \beta = \frac{\sin(\alpha + \beta)}{\sin \alpha \sin \beta}. \tag{4.15}$$

It can be deduced that $\omega_{ij} \geq 0$ if and only if $\alpha + \beta \leq \pi$.

Negative weights can result in the parameterisation being non-bijective. Hence, if choosing weights based on cotangent weights, a good quality preliminary mesh needs to be obtained. It has been shown that if the mesh satisfies the Delaunay criterion [4.35], the parameterisation obtained using cotangent weights in equation (4.14) will always be bijective [4.36]. Providing such a triangulation as an input to this

parameterisation technique should be no issue as the meshing criterion used for Unstructured TLM is the Delaunay criterion as discussed in Chapter 2. Figure 4.9 presents an example of a Discrete Harmonic Parameterisation of the hemisphere first presented in Figure 4.7. The mesh in 3D space constitutes 658 faces, and the surface is parameterised into a circular and square parameter domain.

Mean Value Coordinates are an additional set of barycentric coordinates, derived by Floater [4.37]. By descritising the mean value theorem, the following weights were derived, facilitating another conformal parameterisation.

$$w_{ij} = \frac{\tan(\frac{\gamma_{ij}}{2}) + tan(\frac{\delta_{ij}}{2})}{r_{ij}} \tag{4.16}$$

where $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ are the 3D positions of vertices and γ_{ij} are the angles in the two triangles shared by the edge (i, j), as shown in Figure 4.8. The mean-value weights are always positive and it has been proven that, provided that the boundary points in the parameter domain form a convex shape, bijectivity is guaranteed.

The beauty of using any of these choices is that the coefficients based on ω_{ij} only depend on the angles and distances. For any interior vertex $\boldsymbol{p_i} \in V_I$, the angles and distances are simply taken from triangles about $\boldsymbol{p_i}$. Each offers a method which is efficient and simple to implement, as they only require solving a single linear system. However, the distortion induced by the parameterisation additionally depends on how closely the boundary of the geometry in 3D space matches the boundary in the parameter domain. If the 3D meshes have non-convex boundaries, or boundaries that differ significantly from the specified boundary of the planar domain, these fixed boundary techniques can induce significant distortion at the perimeter of the 2D domain. Therefore, these techniques work best when the original 3D mesh have well-shaped, nearly-convex, boundaries. This can be seen in Figure 4.10, which presents a parameterisation of the hemispherical mesh using Mean Value weights, into a circular and square parameter domain. As with all of the fixed boundary parameterisations presented in this section, the impact on the parameter-

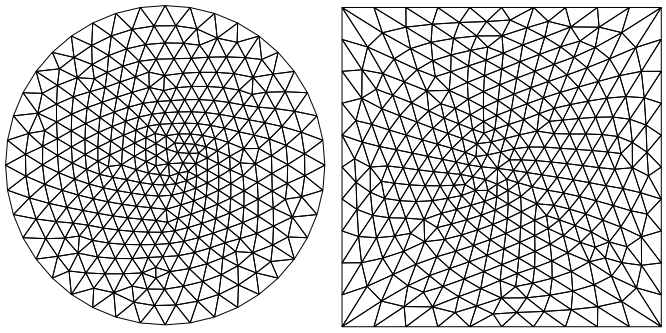

Figure 4.10: Example of a Mean Value Parameterisation [4.20] of a hemisphere (shown in Figure 4.7), meshed using 658 faces, using cotangent weights and circular and square boundary in 2D space

isation the choice of the boundary can have can be seen, where the square domain exhibits significant stretch on triangles at the perimeter of the boundary in the 2D domain.

4.4 The Boundary Conditions For Mappings

Parameterisation techniques that involve the use of barycentric coordinates require the boundary to be predefined in parameter space. Intuitively, the minimization problem can be imagined as an attempt to stretch out and flatten the original mesh in the 3D domain, as if it was a tiny sheet of elastic material (like a small piece of a rubber balloon), over a larger region in the plane. If the boundary of this elastic sheet is not fixed in enough places, it will simply collapse to point. This is conceptually demonstrated in Figure 4.11.

For geometries with regular shaped boundaries, the easiest approach to achieve this minimisation is to map the boundary vertices to the flat plane in a least squares sense. For more complex, irregular shaped boundaries, this simple procedure may lead to undesirable fold-overs in the boundary polygon [4.14]. There are two con-

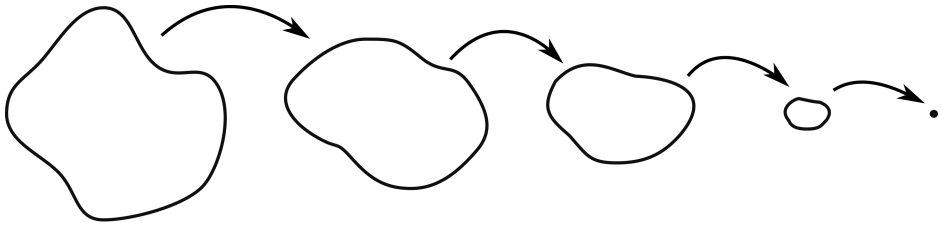

Figure 4.11: If the boundary is not constrained in parameter space, the solution will converge to a single point

siderations when defining the boundary of the parameter space. These are choosing the shape of the parameter domain, and choosing the distribution of the parameter vertices around the boundary.

4.4.1 Choosing the shape

Choosing a well defined, regular, convex shape guarantees the bijectivity of the parameterisation, as long as positive barycentric coordinates are used. For this reason the choices are limited to square, or circular boundaries in the parameter domain. Choosing such convex shapes, when the boundary of the 3D geometry does not represent simple, or regular shapes, can result in large amounts of distortion at the boundary once mapped to the plane. This can be seen in the experimental results in Chapter 5.

4.4.2 Choosing the distribution

The accepted methods in the literature for choosing the distribution of the boundary points is to use a univariate parameterisation method, such as chord length [4.38] or centripetal parameterisation [4.39] for placing the parameter boundary points around the chosen convex shape. Additionally, methods which attempt to fix the boundary vertices in a uniform distribution around the parameter domain are ac-

cepted. Having to fix the boundary vertices may be a severe limitation for some applications. When performing numerical analysis using UTLM for example, the minimum link length of the mesh dictates the largest allowable time step of the simulation. If the boundary mapping results in large amounts of distortion and reduction of the link lengths as a result of distortion to angle or area of the original mesh elements, the runtime of the simulation may be detrimentally affected.

Parameterisations which do not require a fixed boundary, and where the parameter boundary vertices are defined as part of the solution, do exist and are discussed in the following section.

4.5 Setting the Boundary Free

In a bid to reduce the distortion at the boundary of the parameter domain, techniques have been derived which integrate the computation of the boundary points as part of the solution. Lee *et al.* [4.40] achieve this by introducing additional triangles at the boundary of the original 3D mesh in order to create a virtual boundary. An example of this is shown in Figure 4.12. Part a) shows the original 3D geometry. The new virtual boundary is then mapped to a fixed convex boundary in the parameter domain, with the interior vertices parameterised using Floater's method [4.37] [4.20]. This creates a buffer at the boundary, where the vertices of the real boundary are allowed to move freely, with triangles in the virtual boundary absorbing the resultant distortion, obtaining a parameterisation with less distortion at the real boundary. This is demonstrated in Figure 4.12 parts b) and c) showing the resultant mapping to a circular and square domain respectively.

This method, though successful in its efforts to reduce the boundary distortion, is not a simple solution. Modifications to a simulation framework would have to be

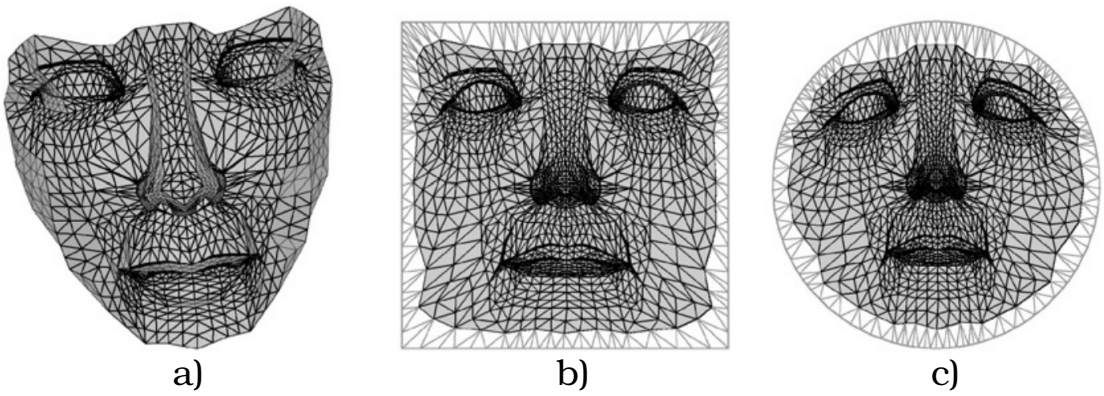

Figure 4.12: Creating a virtual boundary around a mesh in the 3D domain can reduce distortion at the boundary. a) is the original 3D mesh, b) is a parameterisation to a 2D square domain, c) is the same parameterisation to a circular 2D domain. Images taken from [4.40]

made to accommodate for the addition of the virtual triangles, rather than being able to use the 3D geometry natively.

Two explicit formulations of free-boundary, linear parameterisation techniques were derived which remove the need to adapt the original geometry. These are a Least Squares Conformal Map (LSCM) and a Discrete Conformal Parameterisation (DCP), which were independently proposed by Levy *et al.* [4.2] and Desbrun *et al.* [4.1] respectively. Both independently derived equivalent formulations for free-boundary parameterisation that aim to minimize angular distortion.

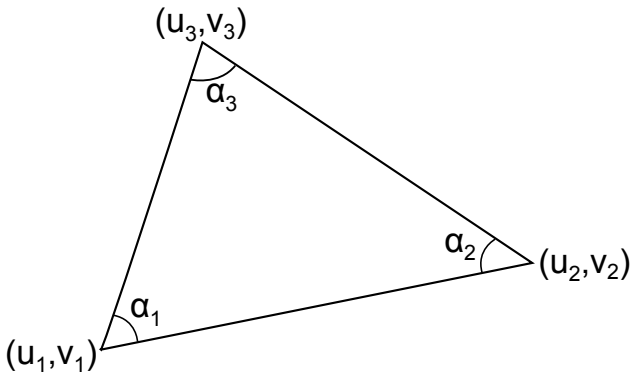

Figure 4.13: Notation for the LSCM parameterisation method

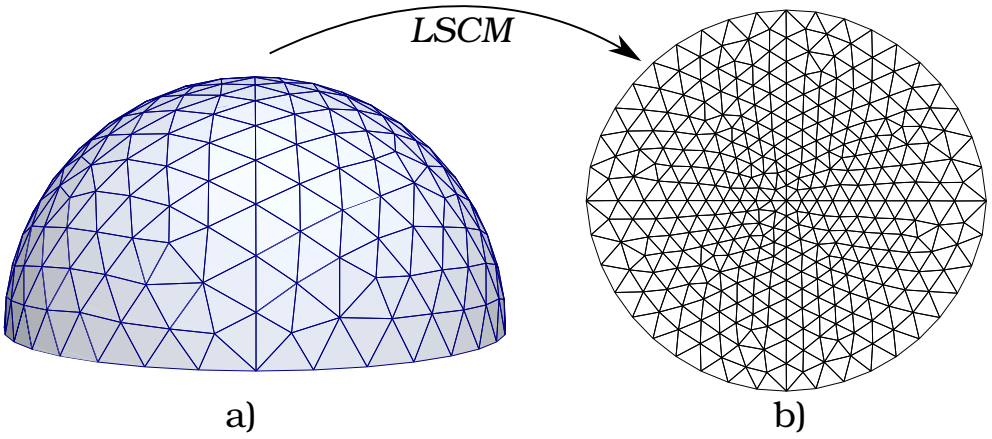

Figure 4.14: A hemispherical surface in the 3D domain (a) is meshed using 658 faces, and parameterised using the free boundary, and linear LSCM method (b)

The formulation of the LSCM method is based on the observation that given the angles α_1 α_2 and α_3 of a planar triangle, as shown in Figure 4.13, the following holds:

$$(u_3, v_3)(u_1, v_1) = \frac{\sin(\alpha_2)}{\sin(\alpha_3)} R^{\alpha_1} \left[(u_2, v_2) - (u_1, v_1)\right], \tag{4.17}$$

where u_i, v_i are the planar coordinates of the triangle vertices and R^{α_1} is rotation matrix with angle α_1. To minimize the angular distortion, the angles from the original 3D mesh are provided as input into equation (4.17) and the square of the distances between the left and right sides are minimized in the least squares sense. In order to avoid the degenerate solution, where all the vertices are at one point, two vertices are fixed, allowing the remaining boundary points to freely move. An example of the resultant parameterisation from using the LSCM method is shown in Figure 4.14, where the hemispherical surface in the 3D domain is approximated using 658 faces, and subsequently parameterised.

The LSCM method do not guarantee local or global bijectivity however, and can theoretically result in flipped triangles, as well as global boundary overlaps. There is no formal investigation in the literature with regards to preventing non-bijective behaviour, however in the experimental investigations presented in Chapter 5, where

the original triangulation is Delaunay, parameterisations resulting from this methodology have been consistently well behaved. While the linear, fixed boundary methods discussed in Section 4.3 can induce distortion, due to the constraints of the boundary condition, the LSCM method introduces significantly less stretch with its free boundary approach, while still providing a linear solution. However, meshes with high surface curvature may result in a significant stretching of mesh elements [4.31].

4.5.1 Non-linear Solutions

Aside from the linear solutions detailed in the previous section, non-linear solutions exist, which attempt to further minimise distortion. The parameterisation in Figure 4.15 is a result of the MIPS (Most Isometric ParameterisationS) method [4.21] [4.41], which provides a non-linear conformal solution pre-dating the LSCM linear method. The MIPS method optimises a non-linear functional that measures mesh conformality. Facilitating a solution amounts to beginning with a fixed-boundary, barycentric parameterisation, used as a best initial guess [4.32]. This is followed by the systematic movement of vertices, each one being moved one at a time, in order to reduce the metric distortion. In an effort to prevent triangular flips, vertices are constrained to move inside the kernel of neighbouring vertices. The procedure guarantees that the solution remains globally bijective throughout, by checking for boundary overlaps when moving boundary vertices. The non-linearity of the solution, however, requires a substantially greater computational effort than the simpler linear solutions [4.23].

Instead of defining a planar parameterisation in terms of vertex coordinates, the ABF (Angle-Based-Flattening) method [4.42] [4.22] [4.43] define it in terms of the angles of the planar triangles. This is based on the observation that a planar triangulation is uniquely defined by the corner angles of its triangles. This simple remark leads

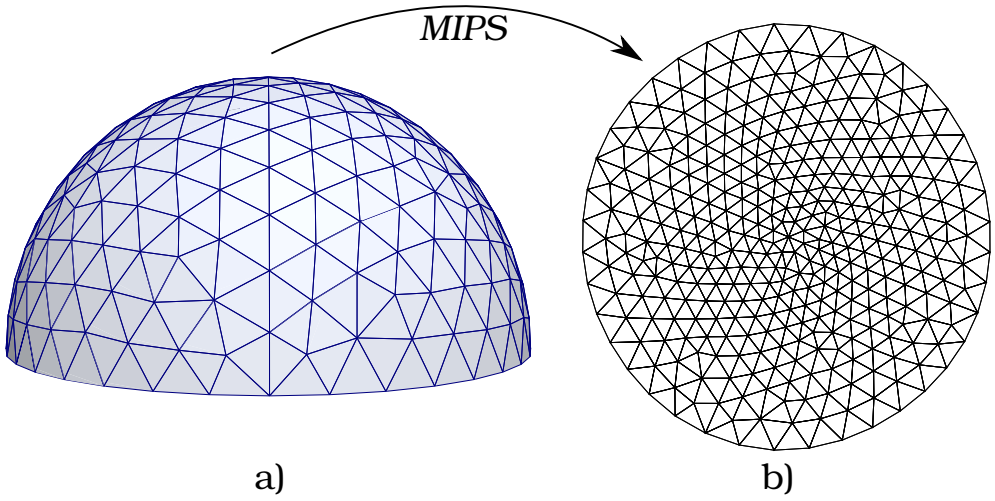

Figure 4.15: Example of a parameterisation of a hemisphere (a) meshed using 658 faces, using the Most Isometric Parameterisation (MIPS) method (b) [4.21]

to the reformulation of the parameterisation problem; finding (u_i, v_i) coordinates in terms of angles. That is, finding α_k^t, where α_k^t denotes the angle in parameter space at the corner of triangle t incident to vertex index k, as shown in Figure 4.16.

For any interior vertex $\boldsymbol{v} \in V_I$, the planar angles α_k sum to 2π, but the corresponding mesh angles in the 3D domain usually do not. This angular deformation is inevitable for piecewise linear mappings, as witnessed in Figure 4.5, and the best one can hope for is that the deformation is distributed evenly around the vertex. The authors of [4.22] therefore define for each vertex $\boldsymbol{v} \in V$ the optimal angle $\beta_i = \theta_i s(\boldsymbol{v})$, where θ_i is the corresponding angle about vertex $\boldsymbol{v}$ and $s(\boldsymbol{v})$ is a uniform scaling factor:

$$s(\boldsymbol{v}) = \begin{cases} \frac{2\pi}{\theta(\boldsymbol{v})} & \text{if } \boldsymbol{v} \in V_I, \\ 1 & \text{if } \boldsymbol{v} \in V_B, \end{cases} \tag{4.18}$$

where $\theta(\boldsymbol{v})$ is the sum of all angles about vertex $\boldsymbol{v}$ in the 3D domain. To determine an optimal set of planar angles, the following energy is minimized [4.22]:

$$E_{ABF}(\boldsymbol{\alpha}) = \sum_{t \in T} \sum_{k=1}^{3} \left(\frac{\alpha_k^t - \beta_k^t}{\beta_k^t} \right)^2, \tag{4.19}$$

where the sum is over all triangles t in the triangulation T, and the energy measures the relative deviation of the unknown 2D angles α_k^t from the *optimal* angles β_k^t, measured on the 3D mesh.

To enforce that the 2D angles define a valid triangulation, a set of constraints needs to be satisfied:
(i) Triangle validity for each triangle t: The three triangle angles must sum to π such that,

$$\forall t \in T: \quad \alpha_1^t + \alpha_2^t + \alpha_3^t - \pi = 0. \tag{4.20}$$

(ii) For each internal vertex the incident angles have to sum to 2π, ensuring and planarity for each internal vertex v such that:

$$\forall v \in V_n: \quad \sum_{(t,k)\in v*} \alpha_k^t - 2\pi = 0, \tag{4.21}$$

where V_{int} denotes the set of internal vertices, and where v^* denotes the set of angles incident to vertex v.

(iii) This constraint ensures that the relations of edge lengths and angles around a vertex are consistent. By fixing the length of one (arbitrary) edge about an interior vertex (edge l_1 in Figure 4.16) moving anti-clockwise over the other edges, the length of the last edge coincides with the length of the first edge if:

$$\forall v \in V_n: \quad \frac{\prod_{(t,k)\in v*} \sin \alpha_{k\oplus 1}^t}{\prod_{(t,k)\in v*} \sin \alpha_{k\ominus 1}^t} - 1 = 0, \tag{4.22}$$

where $k\oplus 1$ and $k\ominus 1$ denote the next and previous angle in the triangle, respectively and the symbol $\prod$ indicates the product of its arguments.

Figure 4.16 demonstrates the necessity of equation (4.22). Considering the centre vertex, and assuming the edge length l_1 is fixed, angles α_1^1 and α_1^2 in face f_1 can be used to calculate l_2 using the relation:

$$\frac{l_1}{1_2} = \frac{\sin \alpha_1^2}{\sin \alpha_1^1}. \tag{4.23}$$

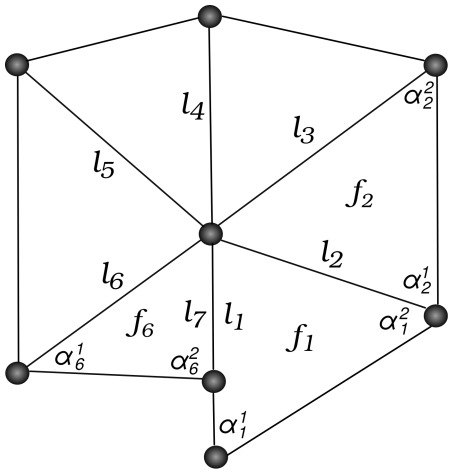

Figure 4.16: ABF parameterisation: Incompatibility of edge length in the one-ring neighbourhood gives rise to the constraint in equation (4.22). Angles α_k^t relate to the indexed k angles of triangle t, f is the face index, and l_i is the edge index.

Using l_2 and the angles α_2^1 and α_2^2 in f_2, l_3 can be computed to obtain an equation analogous to equation (4.23),

$$\frac{l_2}{l_3} = \frac{\sin \alpha_2^2}{\sin \alpha_2^1} \tag{4.24}$$

Combining (4.23) and (4.24), l_3 can be related to l_1 as:

$$\frac{l_1}{l_3} = \frac{l_1 l_2}{l_2 l_3} = \frac{\sin \alpha_1^2}{\sin \alpha_1^1} . \frac{\sin \alpha_2^2}{\sin \alpha_2^1} \tag{4.25}$$

Applying equation (4.23) for each face, traversing anticlockwise around the one-ring neighbourhood, l_4, l_5 ... l_7 can be computed. However for the mesh to be valid, it is necessary for l_7 to equal l_1. Using an equation analogous to equation (4.25) in order to relate l_7 and l_1 we get

$$\frac{l_1}{l_7} = \frac{l_1}{l_2} . \frac{l_2}{l_3} \ldots \frac{l_6}{l_7} \tag{4.26}$$

$$= \frac{\sin \alpha_1^2}{\sin \alpha_1^1} . \frac{\sin \alpha_2^2}{\sin \alpha_2^1} \ldots \frac{\sin \alpha_6^2}{\sin \alpha_6^1} = 1, \tag{4.27}$$

demonstrating the necessity of the constraint in equation (4.22)

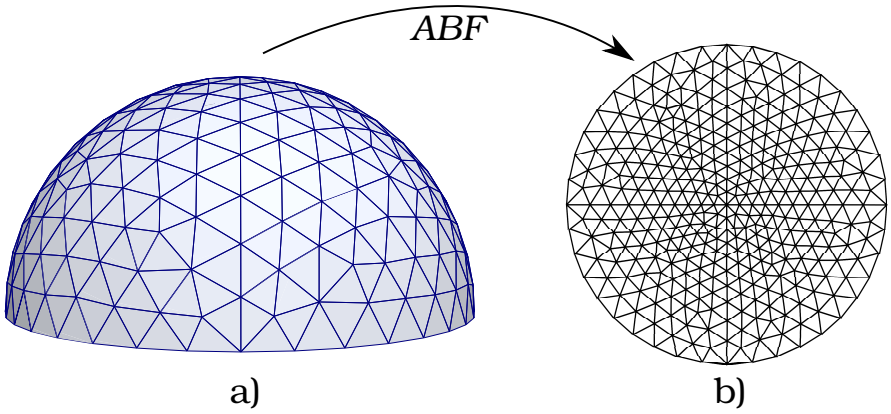

Figure 4.17: Example of a parameterisation of a hemisphere surface (a) meshed using 658 faces, using the Angle Based Flattening (ABF) method (b).

A search for angles that are as close as possible to the original 3D mesh angles ensues, which satisfy these three constraints. The resulting parameterisations are guaranteed to contain no flipped triangles, i.e it is locally bijective, however no guarantees are made about global bijectivity. An example of Figure 4.17, which shows the parameterisation of a hemispherical surface, meshed using 658 faces, resulting from the ABF method. Superficially analysing this parameterisation, it appears that this parameterisation preserves the shape qualities of the individual triangles better than the MIPS method, and only slightly better than the LSCM method.

4.6 Summary

This chapter presented an overview of the mesh parameterisation techniques implemented for this project. The concepts of mesh parameterisation have been introduced, categorising the various methods into linear and non-linear formulations. Linear methods, are simple to implement, but rely on the definition of regular, convex shaped boundary in the 2D domain, which can result in distortion at the boundary in 2D space. Free boundary methods exist, however the parameterisation is a solution to a non-linear system. Mesh parameterisations will introduce

distortion in the form of angular or area distortion, and the mesh parameterisation techniques aim to minimise this distortion in some sense. The resultant distortion will affect the length of the link lines constituting a UTLM network, so choosing a parameterisation technique that minimises this distortion becomes paramount. The next chapter aims at providing an experimental comparison of these techniques, to determine their suitability for UTLM simulations.

References

[4.1] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parametrization of Surface Meshes," *Eurographics*, vol. 21, no. 2, 2002.

[4.2] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Trans. Graph.*, vol. 21, no. 3, 2002.

[4.3] C. Ptolemy, *Geographia*. Editore Alfredo Firmin Didot, 1883.

[4.4] C. Bennis, J.-M. Vézien, and G. Iglésias, "Piecewise Surface Flattening For Non-Distorted Texture Mapping," *ACM SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 237–246, 1991.

[4.5] J. Maillot, H. Yahia, and A. Verroust, "Interactive Texture Mapping," *Proc. 20th Annu. Conf. Comput. Graph. Interact. Tech.*, pp. 27–34, 1993.

[4.6] T. Lee, M. Leok, and N. H. McClamroch, "Geometric Numerical Integration for Complex Dynamics of Tethered Spacecraft," *Proc. 2011 Am. Control Conf.*, no. February, pp. 1885–1891, 2011.

[4.7] A. Lee, H. Moreton, and H. Hoppe, "Displaced Subdivision Surfaces," *Proc.*

27th Annu. Conf. Comput. Graph. Interact. Tech., pp. 85–94, 2000.

[4.8] B. Allen, B. Curless, and Z. Popović, "The Space of Human Body Shapes," *ACM Trans. Graph.*, vol. 22, no. 3, p. 587, 2003.

[4.9] V. Kraevoy and A. Sheffer, "Template-Based Mesh Completion," *Eurographics Symp. Geom. Process.*, pp. 13–22, 2005.

[4.10] D. Hilbert and C.-V. Stephan, *Geometry and The Imagination*, 2nd ed. New york: Chelsea, 1952.

[4.11] K. Hormann, "Parameterizing Triangulations," phD, University of Erlangen, 2001.

[4.12] D. H. Maling, *Coordinate Systems and Map Projections*. Littlehampton Book Services, 1972.

[4.13] K. Hormann, K. Polthier, and A. Sheffer, "Mesh Parameterization," *ACM SIGGRAPH ASIA 2008 courses - SIGGRAPH Asia '08*, vol. 2, no. December, pp. 1–87, 2008.

[4.14] M. S. Floater and K. Hormann, "Parameterization of Triangulations and Unorganized Points," *Tutorials Multiresolution Geom. Model.*, pp. 287–316, 2002.

[4.15] S. Bell, *The Cauchy Transform, Potential Theory and Conformal Mapping*. CRC Press, 1992.

[4.16] J. Walsh, "History of the Riemann Mapping Theorem," *Am. Math. Mon.*, no. 34, pp. 47–94, 1973.

[4.17] K. Hormann and M. Floater, "Surface Parameterisation: A Tutorial and Survey," *Adv. Multiresolution Geom. Model.*, pp. 405–417, 2005.

[4.18] W. T. Tutte, "Convex Representations of Graphs," *Proc. London Math. Soc.*, vol. 10, no. February 1959, 1960.

[4.19] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '95*, vol. 6, pp. 173–182, 1995.

[4.20] K. Hormann and M. S. Floater, "Mean Value Coordinates for Arbitrary Planar Polygons," *ACM Trans. Graph.*, vol. 25, no. 4, pp. 1424–1441, 2006.

[4.21] K. Hormann and G. Greiner, "MIPS : An Efficient Global Parametrization Method," *Curve Surf. Des.*, pp. 153–162, 2000.

[4.22] A. Sheffer and E. de Sturler, "Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening," *Eng. Comput.*, vol. 17, no. 3, pp. 326–337, 2001.

[4.23] K. Hormann, K. Polthier, and A. Sheffer, "Mesh Parameterization," *ACM SIGGRAPH ASIA 2008 courses - SIGGRAPH Asia '08*, no. August, pp. 1–87, 2008.

[4.24] S. Toledo, D. Chen, and V. Rotkin, "TAUCS - A Library of Sparse Linear Solvers," 2003. [Online]. Available: http://www.tau.ac.il/~stoledo/taucs/

[4.25] B. Lévy, "OpenNL - Open Numerical Library," 2010. [Online]. Available: http://alice.loria.fr/index.php/software/4-library/23-opennl.html

[4.26] T. Davis, "SuiteSparse." [Online]. Available: http://faculty.cse.tamu.edu/davis/suitesparse.html

[4.27] M. S. Floater, K. Hormann, and G. Kós, "A General Construction of Barycentric Coordinates Over Convex Polygons," *Adv. Comput. Math.*, vol. 24, no. 1-4,

pp. 311–331, 2006.

[4.28] W. Tutte, "How To Draw a Graph," *Proc. London Math. Soc*, vol. 8, no. May 1962, pp. 743–767, 1963.

[4.29] M. S. Floater, "Parametric Tilings and Scattered Data Approximation," *Int. J. Shape Model.*, vol. 04, no. 03, pp. 165–182, Sep. 1998.

[4.30] P. Sewell, J. Wykes, T. Benson, C. Christopoulos, D. Thomas, and A. Vukovic, "Transmission-Line Modeling using Unstructured Triangular Meshes," *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 5, pp. 1490–1497, 2004.

[4.31] A. Sheffer, E. Praun, and K. Rose, "Mesh Parameterization Methods and Their Applications," *Found. Trends Comput. Graph. Vis.*, vol. 2, no. 2, pp. 105–171, 2006.

[4.32] M. S. Floater, "Parametrization and Smooth Approximation of Surface Triangulations," *Comput. Aided Geom. Des.*, vol. 14, no. 3, pp. 231–250, 1997.

[4.33] E. L. Wachpress, *A Rational Finite Element Basis*. Acadenic Press, New York, 1975.

[4.34] U. Pinkall, K. Polthier, and S. D. Juni, "Computing Discrete Minimal Surfaces and Their Conjugates," *Exp. Math.*, vol. 2, pp. 15–36, 1993.

[4.35] J. R. Shewchuk, "Reprint of: Delaunay Refinement Algorithms for Triangular Mesh Generation," *Comput. Geom. Theory Appl.*, vol. 22, pp. 21–74, 2014.

[4.36] L. Kharevych, B. Springborn, and P. Schröder, "Discrete Conformal Mappings via Circle Patterns," *ACM Trans. Graph.*, vol. 25, no. 2, pp. 412–438, 2006.

[4.37] M. S. Floater and M. S. Floater, "Convex Combination Maps," *Algorithms Approx. IV*, vol. 0, no. 0, pp. 18–23, 2002.

[4.38] Ahlberg, *The Theory Of Splines and Their Applications.* New York: Academic Press, 1967.

[4.39] E. T. Y. Lee, "Choosing Nodes in Parametric Curve Interpolation," *Comput. Des.*, vol. 21, no. 6, pp. 363–370, 1989.

[4.40] Y. Lee, H. S. Kim, and S. Lee, "Mesh Parameterization With a Virtual Boundary," *Comput. Graph.*, vol. 26, no. 5, pp. 677–686, 2002.

[4.41] K. Hormann, G. Greiner, and S. Campagna, "Hierarchical Parametrization of Triangulated Surfaces," *Proc. Vision, Model. Vis. 1999*, pp. 219–226, 1999.

[4.42] A. Sheffer and E. De Sturler, "Surface Parameterization for Meshing by Triangulation Flattening," *Proc. 9th Int. Meshing Roundtable*, pp. 161–172, 2000.

[4.43] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov, "ABF++: Fast and Robust Angle Based Flattening," *ACM Trans. Graph.*, vol. 24, no. 2, pp. 311–330, 2005.

Chapter 5

Comparison of Planar Methods

This chapter is dedicated to a comparison of the parameterisation techniques that have been discussed in Chapter 4. The purpose of this experimentation is to realise the effects of the distortion each parameterisation technique imposes, in addition to investigating the length of time each takes to converge to a mapping solution. Even if the mesh in 3D space is well conditioned, there are no guarentees that the parameterisation will result in a well conditioned mesh in the 2D domain. A parameterisation could exhibit large amounts of angular distortion, stretch, or a combination of the two. This can result in a negative impact on the runtime of an Unstructured Transmission Line Modelling Simulation if the parameterisations induce small UTLM link lengths. Additionally the accuracy of the simulation can be affected, which is a consequence of the link line parameters differing greatly between the 2D parameter domain and the 3D domain.

The investigation is restricted to geometries that are open surfaces, topologically homeomorphic to a disk. This means no cuts in the mesh are necessary to flatten the surface to the two-dimension plane. Closed surfaces, such as spheres, require a seam to be cut into the surface such that they can be parameterised. This would introduce boundaries in the geometry where continuity is expected. UTLM simulations can be carried out on parameterisations of closed meshes by ensuring continuity of fields

across the seam with which the mesh was cut, however, the best practices for mesh cutting are reserved for a future investigation.

In addition to distortion, other factors should be considered when choosing a parameterisation method for the application at hand:

- **Free versus fixed boundary**. Many methods assume the boundary of the planar domain to be pre-defined and convex. It has been shown in the previous chapter that fixed boundary methods typically use simple formulations. This should result in parameterisations that are very fast to compute. Such methods are well suited for some applications, for instance, parameterising geometries where the boundary of the original three dimensional surface resembles a simple convex shape. However if the problem geometry possesses arbritary shaped boundaries, free boundary solutions, which determine the boundary as part of the solution, are often slower but typically introduce significantly less distortion. The effects of the boundary choice will be experimentally compared. For techniques that demand a predefined boundary in the parameter domain, the parameterisations will be presented for both, a circle domain and a square domain, such that the effect of fixing a boundary for a parameterisation can be evaluated, and compared with the free boundary techniques.

- **Robustness**. Most applications of parameterisation, including our application for the purpose of UTLM simulations, require it to be bijective. For some applications local bijectivity (no triangle flips) is sufficient while others require global bijectivity (where by the boundary does not self-intersect). Only a subset of the parameterisation techniques guarantee local or global bijectivity. Some can guarantee this condition provided that the initial mesh in 3D space is well conditioned, conforming to specific criteria, such as the Delaunay criterion [5.1]. Ensuring the bijectivity condition is met will become part of the investigation. For the purposes of Unstructured Transmission Line Mod-

elling, the parameterisation is required to be bijective. In the event that the direction of the normal of a triangle is reversed, resulting in triangle flips and consequently negating the bijectivity of the mapping, the connectivity of the mesh moving from the 3D domain to the plane is altered. This inconsistency of triangle orientation will result in the TLM ports, the interface between transmission link lines at the triangle edges, coinciding with different edges when the TLM network is constructed in the 2D domain.

- **Numerical Complexity**. The existing methods discussed in Chapter 4 were roughly classified into linear and non-linear methods. Linear methods are typically significantly faster and simpler to implement. However, the simplicity usually comes at a greater cost of increased distortion. If acceptable levels of distortion are attainable using a linear methodology, not adopting the simpler implementation would be irrational, given that the non-linear approaches are significantly more difficult to implement and require a greater computational effort [5.2]. This is investigated in Section 5.4.

Table 5.1 summarises the methods reviewed thus far, each defined by the distortion minimised, the boundary conditions, whether bijectivity is guaranteed, and the complexity of the implementation. The first three table entries are the parameterisation methods discussed in Section 4.3, which rely on barycentric coordinates. These are Uniform weight parameterisation [5.3], Discrete Harmonic Parameterisation, based on cotangent weights [5.4] and Mean Value Coordinates [5.5]. The remaining linear methods are the Discrete Authalic Parameterisation [5.6], the only method which prioritises the preservation of area over angle, and the Least Squares Conformal Map method [5.7] (Section 4.5), offering the only linear, free boundary solution. The non-linear methods under investigation are the Most Isometric Parameterisation method [5.8] and the Angle Based Flattening method [5.9], introduced in Section 4.5.1 in Chapter 4.

Parameterisation Method	**Distortion Minimised**	**Boundary**	**Bijectivity**	**Complexity**
Uniform [5.3]	None	Fixed, Convex	Yes	Linear
DHP [5.4]	Angles	Fixed, Convex	No	Linear
Mean Value [5.5]	Angles	Fixed, Convex	Yes	Linear
DAP [5.6]	Area	Fixed, Convex	No	Linear
LSCM [5.7]	Angles	Free	No	Linear
MIPS [5.8]	Angles	Free	Yes	Non-Linear
ABF/ABF++ [5.9]	Angles	Free	Yes (local)	Non-Linear

Table 5.1: A summary of the parameterisation methods discussed, categorised by the primary distortion minimised by each technique, whether local and global bijectivity is guaranteed, and the complexity of each method in terms of the difficulty of the computation

These methods have been implemented as part of an experimentation framework, such that each can be compared to gain an insight into the suitability for Unstructured TLM simulations. Each method has been tested by parameterising a selection of geometries, each meshed with increasing degrees of mesh refinement. This facilitates an investigation into the effect of mesh density on each technique, not only in terms of the induced distortion, but also the amount of time necessary to map the geometry from the 3D domain to 2D parameter space.

The geometries under test are shown in Figure 5.1. Figure 5.1 a) is a hollow hemispherical surface with a radius of 1m. This provides a simple initial geometry, with a regular convex boundary, with which to begin the investigation.

A slightly more complicated geometry provides the next test case. Figure 5.1 b) i) shows an arbitrary flexible sheet of uniform material in the 3D domain. This flexible sheet has a width of 0.1m and an unflexed length of 0.2249m, resulting

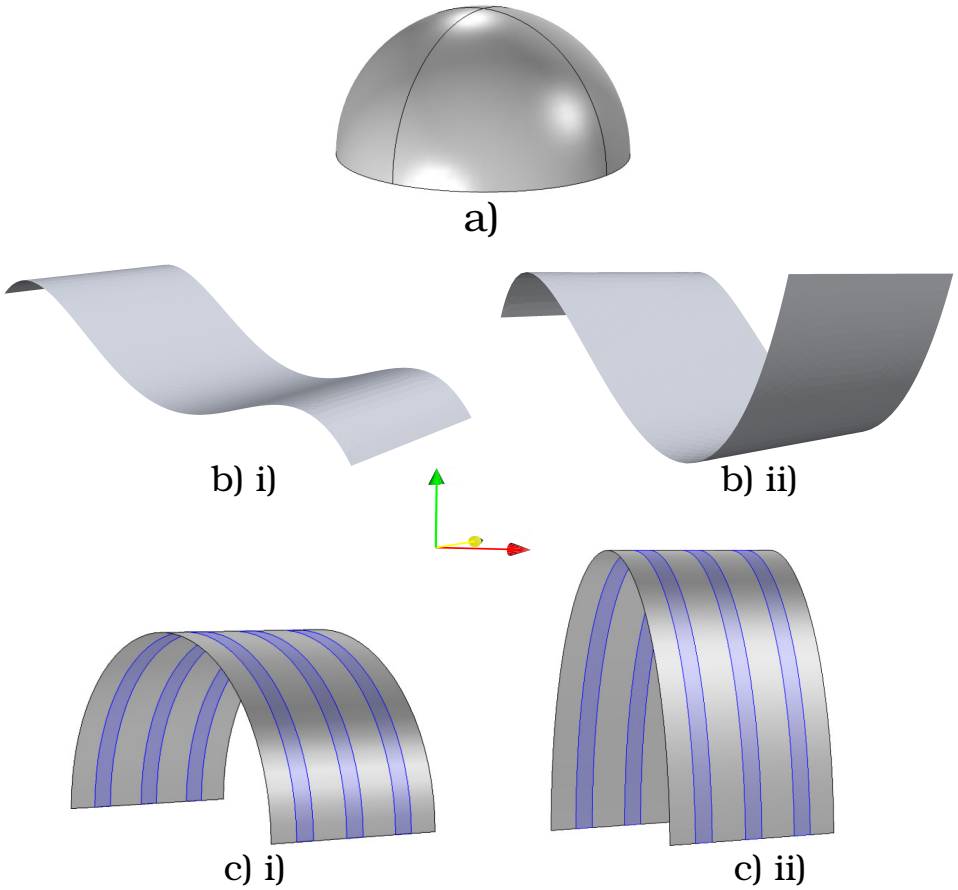

Figure 5.1: The test cases used to investigate each parameterisation method. a) is an open hemispherical surface, 1m in diameter, b)i) is a flexible metalic surface, with width 0.1m and unflexed length of 0.2249m, flexed into an arbritary shape. b)ii) is the same flexible metallic sheet, with a change in curvature. c)i) is a curved flexible PCB with three metal wire tracks seperated by the substrate. c)ii) is the same flexible PCB flexed into a new curvature.

in a surface area of 0.02249m^2. This surface will allow an investigation into how a change in curvature affects the result of a parameterisation. The flexible sheet is then deformed into a new shape, as shown in Figure 5.1 b) ii). The physical dimensions, in terms of length and width, of the newly deformed surface is kept consistent with the original one.

Figure 5.1 c i) is a surface model of a flexible PCB. The purpose of this test case is to enable an investigation into how attributes attached to the original surface are preserved moving from 3D space to parameter space. The material parameters of this surface are no longer uniform. Three metal tracks are modelled, separated by the substrate. A parameterisation should preserve the material properties through the mapping process for a valid solution, allowing a simulation to be carried out in the 2D domain for non-uniform materials. The curvature of this surface is uniform, conforming to that of a cylinder. Therefore, if the literature is to be believed [5.10] [5.2], the parameterisations should result in a mapping that is very close to an isometric one for free boundary parameterisations. Isometry, however, may not be achieved if the parameterisation method requires the boundary to be fixed and pre-defined in parameter space, especially in the event that the parameter domain does not match the original geometry shape. Figure 5.1 c ii) shows this geometry flexed, resulting in a different, now non-uniform, curvature.

Chapter 3 demonstrated how the underlying data structure for representing these meshes impacts the runtime of applying algorithms to a mesh. For this reason each mesh has been represented by the halfedge data structure (Section 3.5) to ensure consistency in the execution of each parameterisation methodology. All tests are carried out using the same computational resources. These are an Intel quad core i7, 2.67GHz processor, 16GB of RAM, and using a Linux operating system (Ubuntu 14.10).

5.1 Case 1: Hemispherical Open Surface

The hemisphere in Figure 5.1 a) has been meshed using 420, 658, 1424, 3300, 14792, 45648 and 152912 faces. Figure 5.2 shows three of these test meshes, a) shows the mesh with 658 elements, whilst b) has 3300, and c) has 14792 faces. Each one of these meshes has subsequently been parameterised using each methodology listed in Table 5.1, and the induced distortions to the metrics, in terms of angle and area to each mesh face, are measured such that they can be compared. The length of time each parameterisation technique takes to arrive at a solution is also recorded across the varying degrees of mesh refinement.

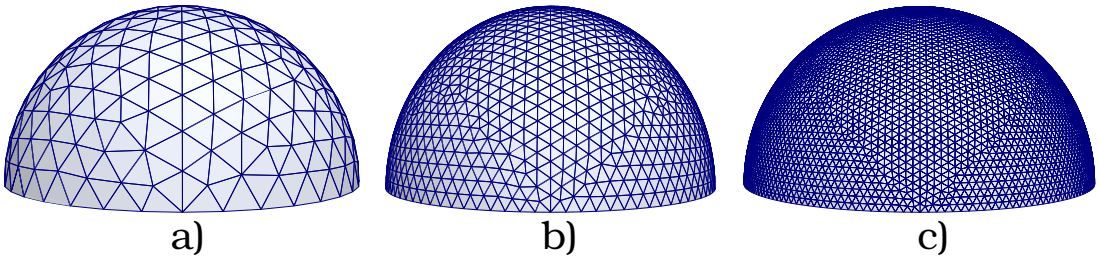

Figure 5.2: Open surface hemisphere with 1m radius, meshes using a) 658 faces, b) 3300 faces and c) 14792 faces

The first metric under investigation is area. The error in area experienced by a parameterised face constituting a mesh, relative to its corresponding face in the 3D domain is calculated using:

$$\Delta^{area} = \frac{\|area^t_{2D} - area^t_{3D}\|}{area^t_{3D}}, \tag{5.1}$$

where Δ^{area} is the relative error in area and t is the index of the triangle in 3D space and its corresponding triangle in the 2D domain.

Focusing the results for the fixed boundary parameterisations, Figure 5.3 shows the maximum relative error in area experienced by a single triangle, as the mesh density is increased. Figure 5.3 a) shows the maximum error induced by a triangle when fixing the parameter domain to be a square, and b) is based on a pre-defined

circular boundary. The 2D domain is predefined to be the same surface area as the original geometry in the 3D domain. The effect the definition of the boundary has on authalic deformation is demonstrated through the comparison of these two graphs. The square domain results in a high maximum error in area, with an increasing trend as more faces are incorporated into the mesh. The circular boundary, which better conforms to the circumference of the hemisphere, reduces the maximum relative error in area significantly compared to the square boundary. Counter to the square parameter domain, the maximal amount of distortion decreases as more faces are introduced to the 3D mesh and parameterised to the circular parameter domain. The use of a circular boundary, which better describes the boundary of the 3D geometry is shown to induce a maximum error of 10% for the coarsest mesh, which reduces to 0.5% for the finest mesh when using all of the fixed boundary techniques, with the exception of the Uniform weighted barycentric mapping. This method performs badly compared to the other methods, inducing a maximum relative error of at least 30% for the circular boundary. It should be pointed out that the DAP method, which primarily focuses on minimising distortion to area, was outperformed by the other fixed boundary techniques which concentrate on minimising distortion to angles (with the exception of the Uniform mapping), though the error in area between them is minimal.

Comparing the maximum relative error in area, Δ^{area}, for the free boundary techniques, Figure 5.4 shows that the ABF method generally exhibited the least amount of maximal error, converging to an error of 0.5% as the number of faces constituting the mesh tends towards 150,000. As the mesh refinement increases, the other free boundary methods converge to a maximum relative distortion of 1%. Interestingly, the only linear free boundary technique compares favourably to the non-linear MIPS parameterisation in terms of preventing stretch through the mapping. LSCM performed better than MIPS on meshes above 1000 faces, and converged to a final error of 1%, while MIPS yielded a final maximal error of 0.9%.

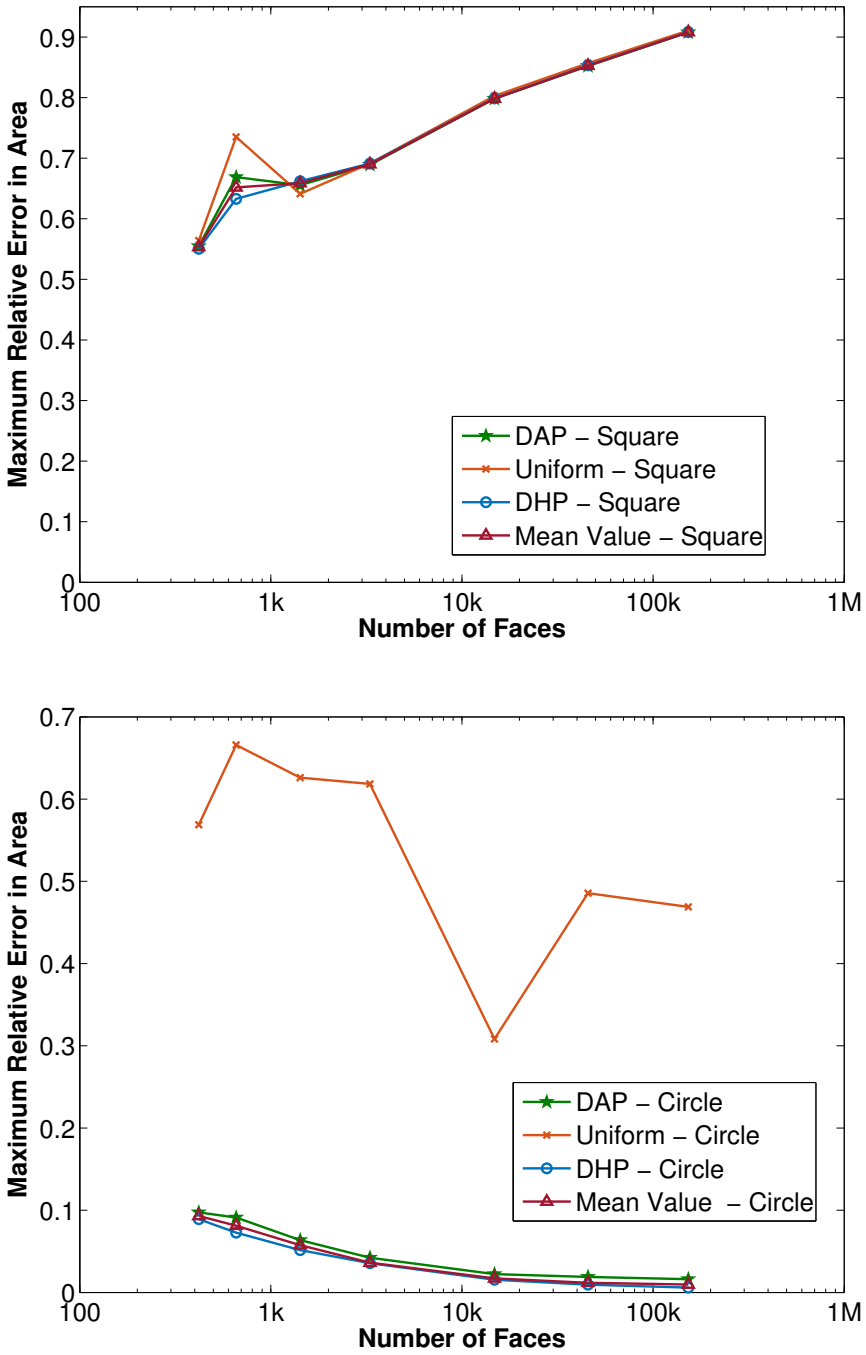

Figure 5.3: Maximum relative error in area experienced by a triangle as the mesh density is increased for the fixed boundary parameterisations for the hemisphere : a) shows the error using a square parameter domain, and b) using a circular domain.

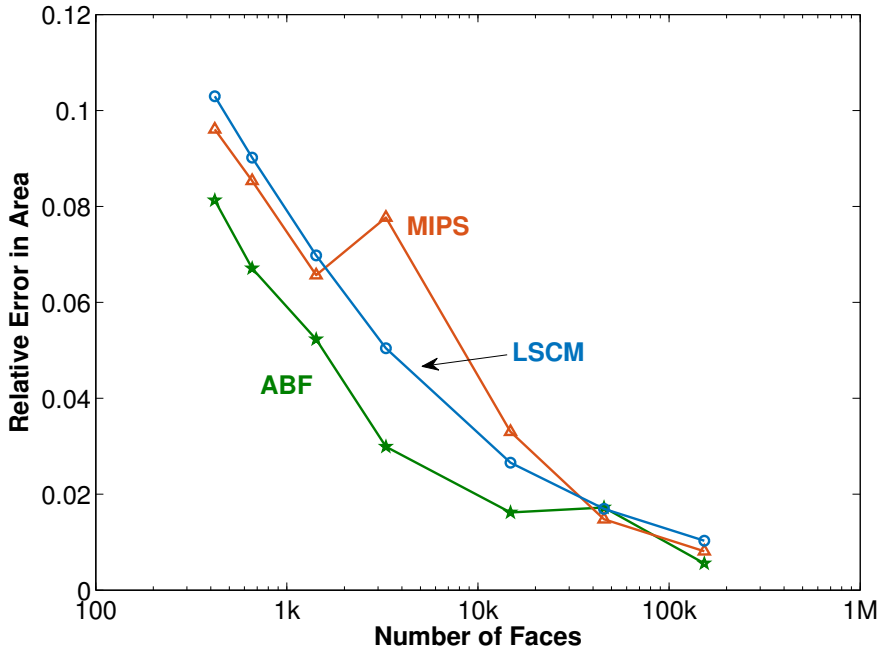

Figure 5.4: Maximum relative error in area experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the hemisphere

Attention is now turned to the results for angular deformation, and how well the shape of individual triangles is preserved through the parameterisation. The relative error in angle is calculated as:

$$\Delta^{\alpha} = \frac{\|\alpha^i_{2D} - \alpha^i_{3D}\|}{\alpha^i_{3D}}, \tag{5.2}$$

where Δ^{α} is the relative error in angle and α^i_{3D} is the index of an angle inside a triangle in 3D space and its corresponding angle in the 2D domain, α^i_{2D}. Figure 5.5 shows graphs of the maximal angular distortion experienced by an individual angle through a fixed boundary parameterisation as the mesh density is increased. Figure 5.5 a) shows the results obtained using a square domain. A similar trend is seen in the angular distortion as with the relative area distortion from Figure 5.3 a); an increasing trend in the maximum metric distortion as the mesh density is increased. This again is in contrast to the distortion exhibited through the use of the circular domain (Figure 5.5 b)), which better matches the shape of the boundary of the

problem domain. As the mesh density is increased, the relative error in angular distortion decreases. The Uniform weight parameterisation demonstrates the only exception to this; it is characterised by high levels of distortion and a more erratic and inconsistent trend. Ignoring this exception, the use of the circular domain for parameterising this test case demonstrates close similarities in results with the free boundary techniques. In fact, the fixed boundary techniques outperform the free boundary techniques for the mesh with the highest face count. The trend in maximum relative error for the free boundary techniques is shown in Figure 5.6. Each free boundary technique, like the fixed boundary techniques, induces a maximum error of 10% for the extremely coarse mesh, which tends towards 1% as the mesh becomes more refined. ABF consistently induces the least maximal distortion to angles, resulting in angles that deviate from their original magnitude by 0.59%. A parameterisation using DHP, and a circular domain matches ABF, resulting in a maximum relative error of 0.57%.

The analysis of this first test case highlights the effect that the boundary can have on the resulting parameterisation. The fixed boundary parameterisations offer a much simpler implementation and, on the provision that a parameter domain which matches the boundary of the original geometry can be used, the performance of the linear techniques are comparable to the non-linear methods; the exception to this being the Uniform weight fixed boundary method.

The maximum error in angle and area provides an insight into the maximum amount of error a single parameterised triangle experiences, relative to its corresponding triangle in the 3D domain. However, it does not provide a clear picture of how angular and authalic distortion is distributed across the 2D surface. Figure 5.7 shows the distribution of area distortion across a relatively fine mesh, consisting of 14792 faces, resulting from a parameterisation using the fixed boundary methods. It can be seen here that a high degree of error resides at the boundary of the square domain,

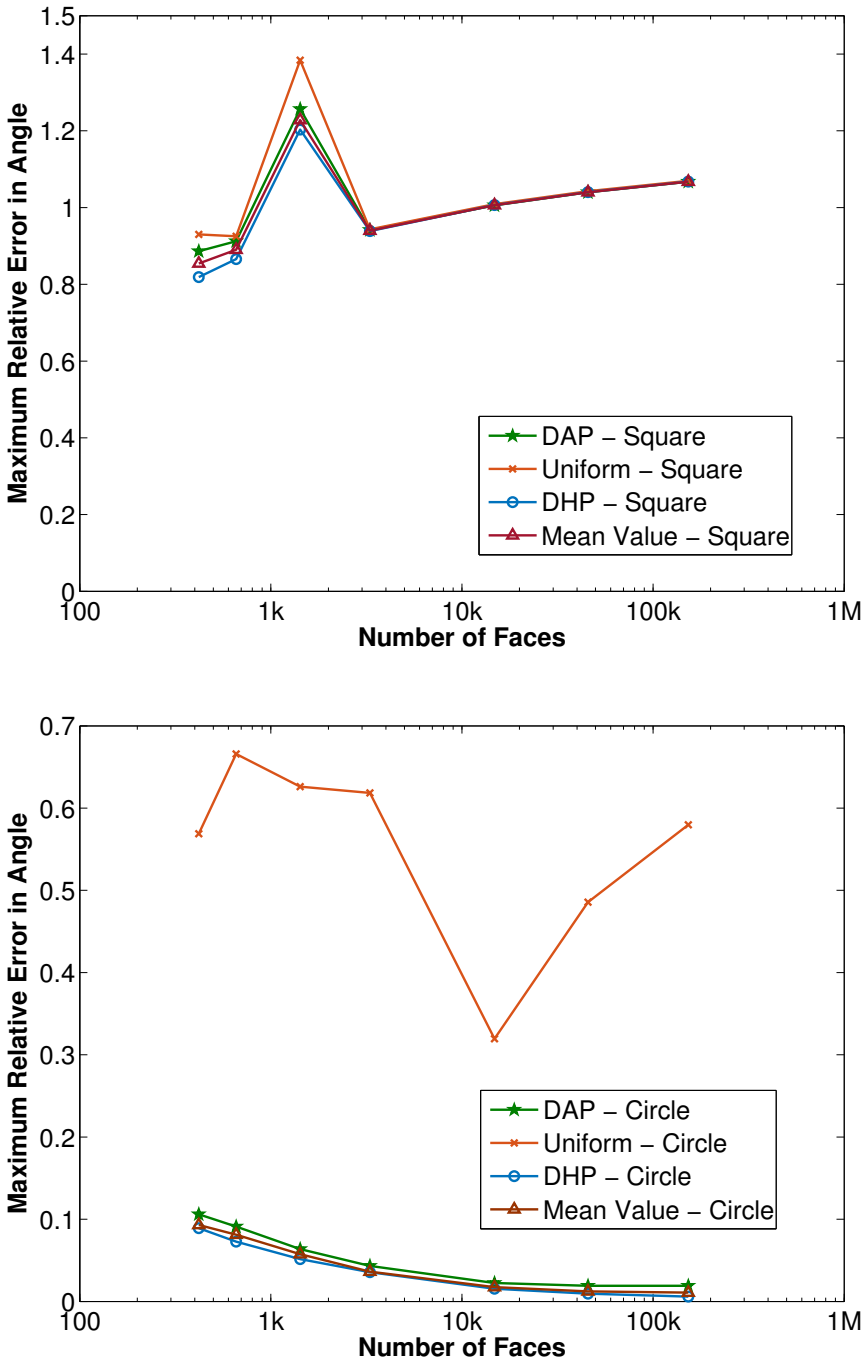

Figure 5.5: Maximum relative error in angle experienced by a triangle as the mesh density is increased for the fixed boundary parameterisations of the hemisphere: a) shows the error using a square parameter domain, and b) uses a circular domain.

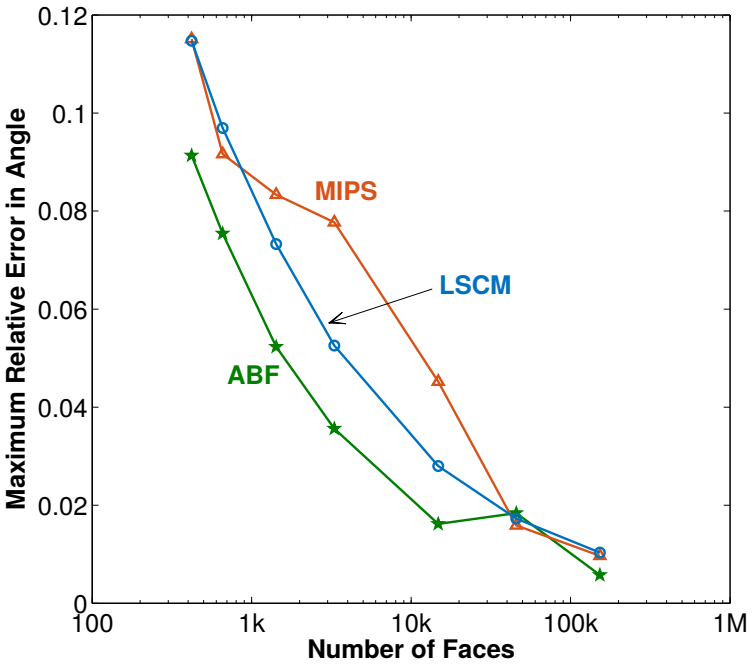

Figure 5.6: Maximum relative error in angle experienced by a triangle as the mesh density is increased for the free boundary parameterisations for the hemisphere

particularly at the corners. No detail is seen in parts b), c) and d) for the circular domain parameterisations, which exhibit very little distortion. This distribution of error is also matched with regards to angles, shown in Figure 5.8 a), for the fixed boundary, square and circular domain. The Uniform weight barycentric mapping demonstrates a greater amount of distortion distributed towards the centre of the mesh for both area and angle in Figure 5.7 a) and Figure 5.8 a) respectively.

Figure 5.9 and Figure 5.10 provide a visual comparison of the free boundary parameterisations for distortion in area and angle respectively. Pat a) of each figure shows the 2D domains for the ABF method, b) is the MIPS method, and c) depicts the distribution of error for the LSCM parameterisation in each case. Parts a)i), b)i) and c)i) depict the distribution using a scale of 0 to 1, allowing a direct comparison between these free boundary parameterisations and the fixed boundary techniques shown in Figure 5.7 and Figure 5.8, which were presented using the same

scale. This comparison suggests that the average magnitude in conformal and authalic distortion is very low, but this scale does not provide much insight into to the actual distribution of the little distortion induced. For this reason, Figure 5.9 and Figure 5.10 a)ii), b)ii) and c)ii) show the distribution of the relative error in area and angle induced by the ABF, MIPS and LSCM methods respectively, on a scale from 0 to 0.03. The magnitude of the relative error each parameterisation method induces is largely below 0.7%, with the maximum error seen only at two points when using LSCM. This coincides with the two vertices chosen to fix the boundary in parameter space. The free boundary techniques parameterise the boundary of the original geometry as part of the solution, which for this test case results in a parameter domain that is subtly elliptical.

This hemispherical test case has demonstrated that parameterisation methods based on linear formulations can provide mappings that induce similar levels of distortion to the non-linear solutions. This is particularly the case with the parameterisation methods which require a predefined parameter domain. The defined boundary is the limitation with these techniques, however in the case of the circular parameter domain, the boundary closely matches the boundary of the original surface in 3D space, resulting in parameterisations which perform similarly to the free boundary techniques. The linear solution provided by LSCM also performs well compared to the other free boundary, but non-linear, methods. The next test case provides a geometry with a different and varying curvature in 3D space, with a boundary that is no longer regular. This is investigated in Section 5.2.

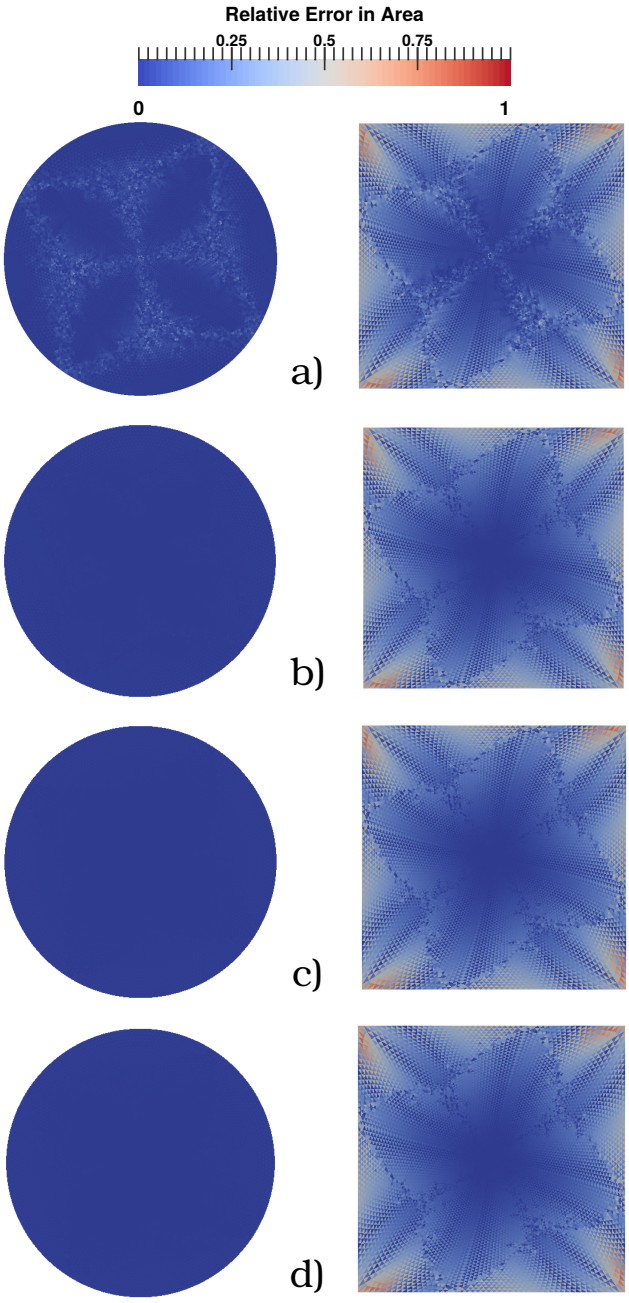

Figure 5.7: Distribution of relative error in area for fixed boundary parameterisations using a) Uniform b) DAP c) DHP d) Mean Value, for a hemisphere meshed using 14792 faces, using circular (left) and square (right) parameter domains.

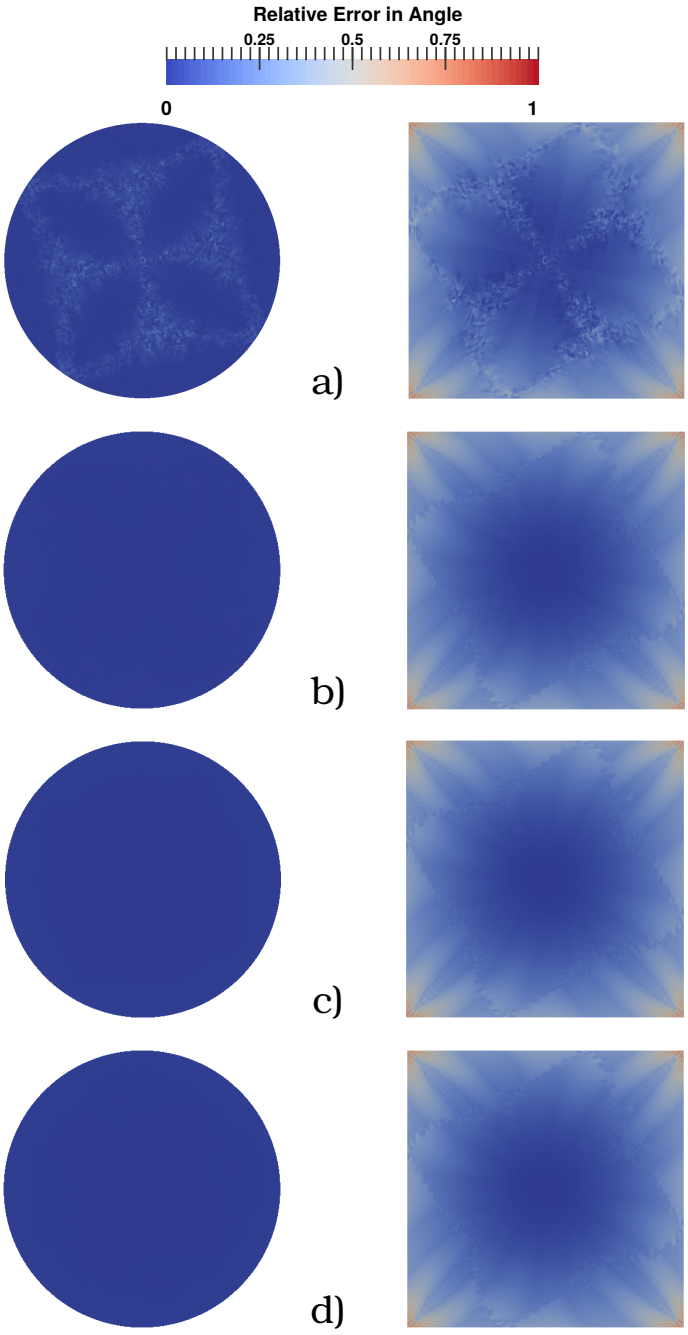

Figure 5.8: Distribution of relative error in angle for fixed boundary parameterisations a) Uniform b) DAP c) DHP d) Mean Value, for a hemisphere meshed using 14792 faces, using circular (right) and square (left) parameter domains.

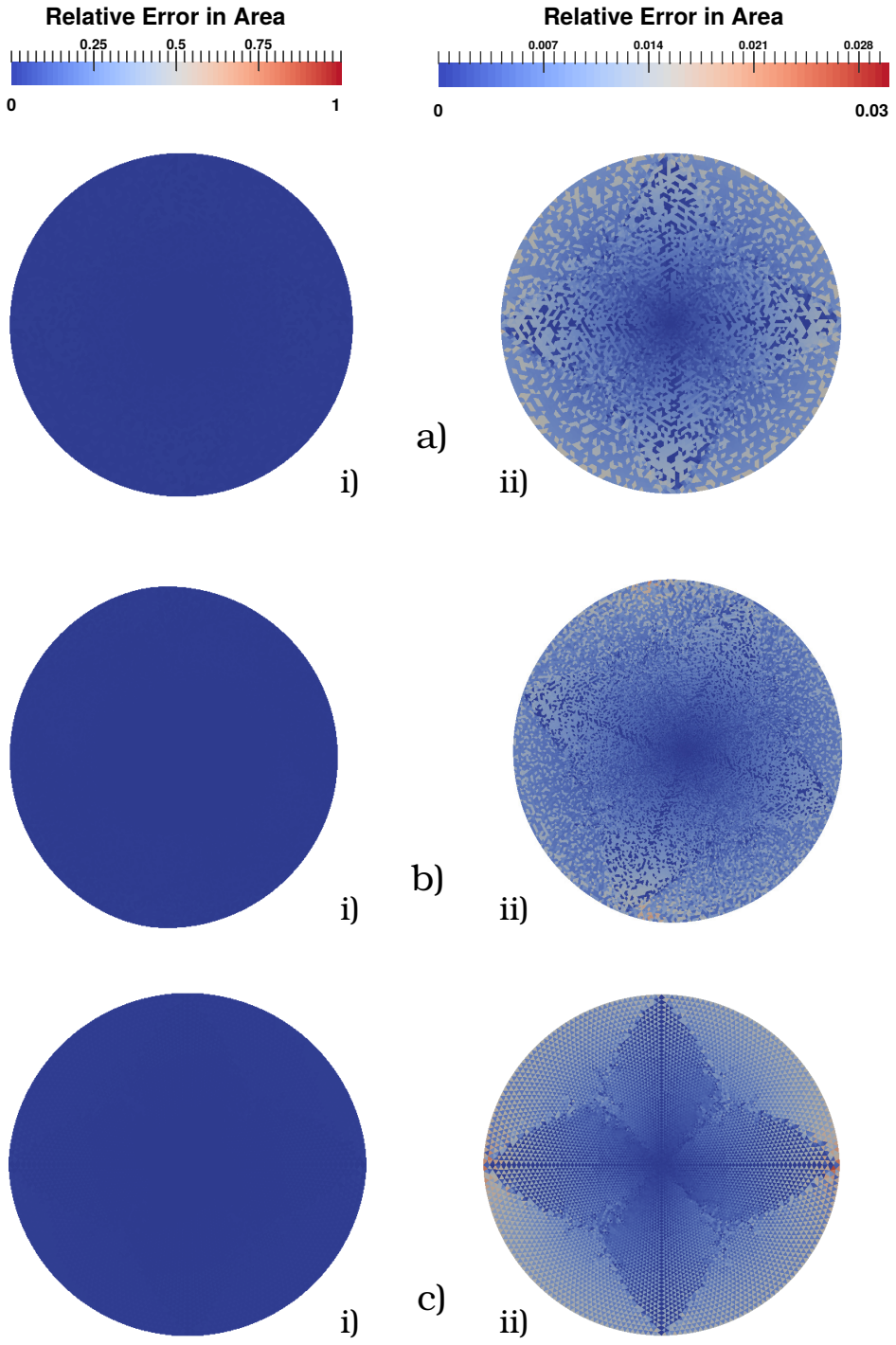

Figure 5.9: Distribution of relative error in area for free boundary parameterisations a) ABF b) MIPS c) LSCM for a hemisphere meshed using 14792 faces. To the left, the scale shows the error between 0 and 1. The right scale has been zoomed in from 0 to 0.03

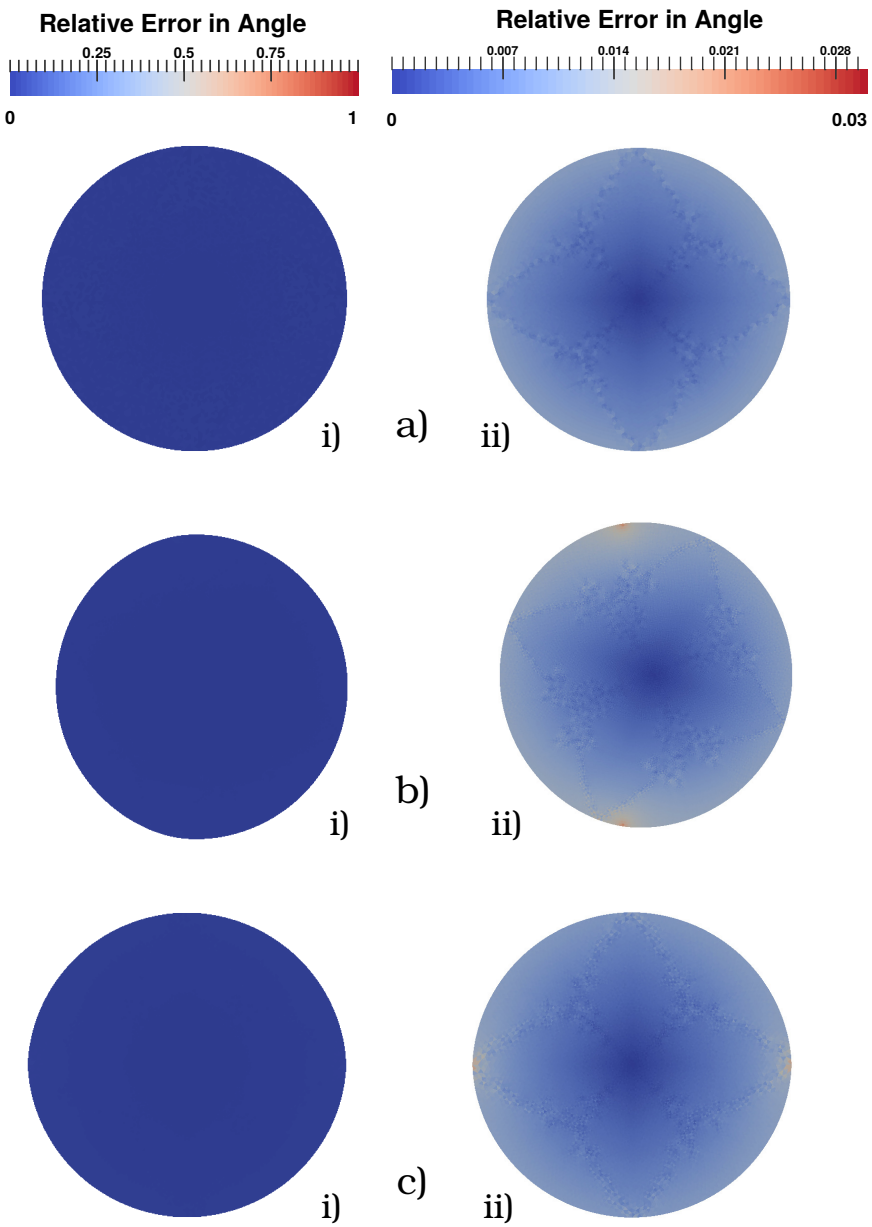

Figure 5.10: Distribution of relative error in area for free boundary parameterisations a) ABF b) MIPS c) LSCM for a hemisphere meshed using 14792 faces. To the left, the scale shows the error between 0 and 1. The right scale demonstrates the relative error on a scale from 0 to 0.03

5.2 Case 2: Flexible Sheet

The next test case is a flexible metallic sheet, shown in Figure 5.1 b)i) which has been bent into an arbitrary shape. The dimensions of this geometry are 0.1m in width and 0.2249m in length. The surface in the 3D domain has been meshed using 303, 628, 1050, 2170, 3140, 6860, 17882 and 55938 faces. Three of these meshes are shown in Figure 5.11. Part a) shows the mesh with 628 faces, b) 3140 faces and c) 6860 faces.

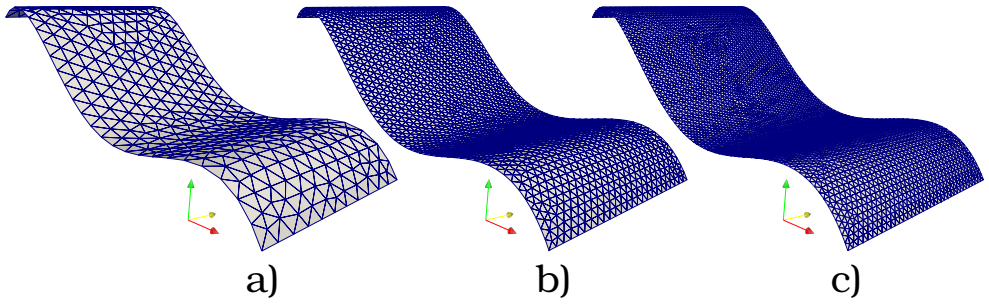

Figure 5.11: Flexible sheet, meshed using a) 628 faces, b) 3140 faces and c) 6860

Neither the circular or square parameter domains conform to the boundary of this flexible sheet in 3D space. From the experience of parameterising the hemisphere to a square domain in the previous subsection, it is expected that fixed boundary parameterisations will induce a significant amount of distortion at the boundary of the 2D parameter domain. The maximum relative error in area exhibited by an individual triangle using the fixed boundary parameterisation techniques is calculated using Equation (5.1) and is presented in Figure 5.12. Figure 5.12 a) shows the maximum relative error in area when using the square parameter domain. Here, the maximum error experienced using this boundary is at least 160% for coarse meshes, which increases to nearly 280% for the mesh constituting 55938 faces, regardless of the fixed boundary parameterisation method used. A similar trend is seen for the circular parameter domain in Figure 5.12 b) which shows a maximum realtive error in area between 100% and 150% for coarse meshes, and rising to nearly 250% at the

upper limit of mesh density.

Figure 5.13 demonstrates the distribution of the relative error in area induced by the fixed boundary parameterisations for the flexible sheet, meshed using 17882 faces, using square and circular boundaries, where a) is the Uniform weight parameterisation, b) is the Discrete Authalic Parameterisation, c) pertains to the Discrete Harmonic Parameterisation and d) depicts the Mean Value method. For each method, i) shows the parameterisation for the circular domain, and ii) the square domain. The distribution of relative error clearly shows the magnitude of the maximum relative error in area is not simply experienced by one or two faces. A high relative error is exhibited not only at the boundary of each parameterisation, but also within the domain.

The same can be seen with angle distortion for the fixed boundary parameterisations. Figure 5.14 shows the graphs comparing the two pre-defined boundaries for each of the methods. Parts a) shows the maximum relative error between angles moving from the 3D domain to 2D using a square boundary, and part b) represents the maximum relative error induced using the circular domain, for an increasing mesh density. The relative error is calculated using Equation (5.2) in each case.

These graphs, again demonstrate the excessive angular distortion experienced by the mesh when flattened to the 2D plane by the fixed boundary methods. Using a square boundary, one can expect at least one angle in the mesh to be distorted by 200 % for a coarse mesh, increasing to 280% for a fine mesh consisting of 55938 faces. This is not quite as severe for the circular boundary. However, maximum angular distortion ranging from 130% and 250%, depending on the degree of mesh refinement, can be expected.

The distribution of angular distortion for the fixed boundary methods is shown in Figure 5.15. It can be seen that the distribution of this metric distortion is similar

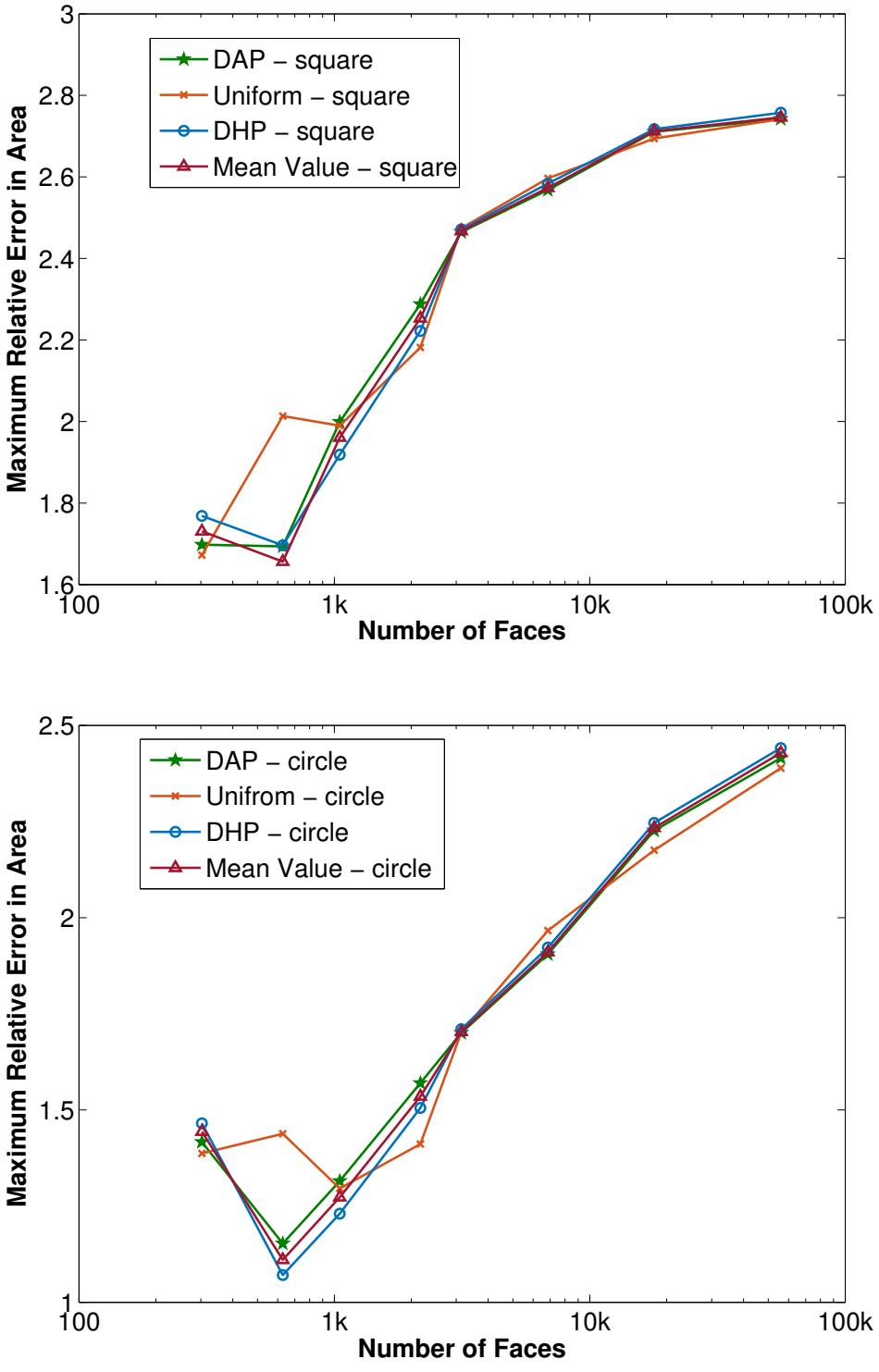

Figure 5.12: Maximum relative error in area experienced by a triangle as the mesh density is increased for the fixed boundary parameterisations of the flexible sheet : a) shows the error using a square parameter domain, and b) uses a circular domain.

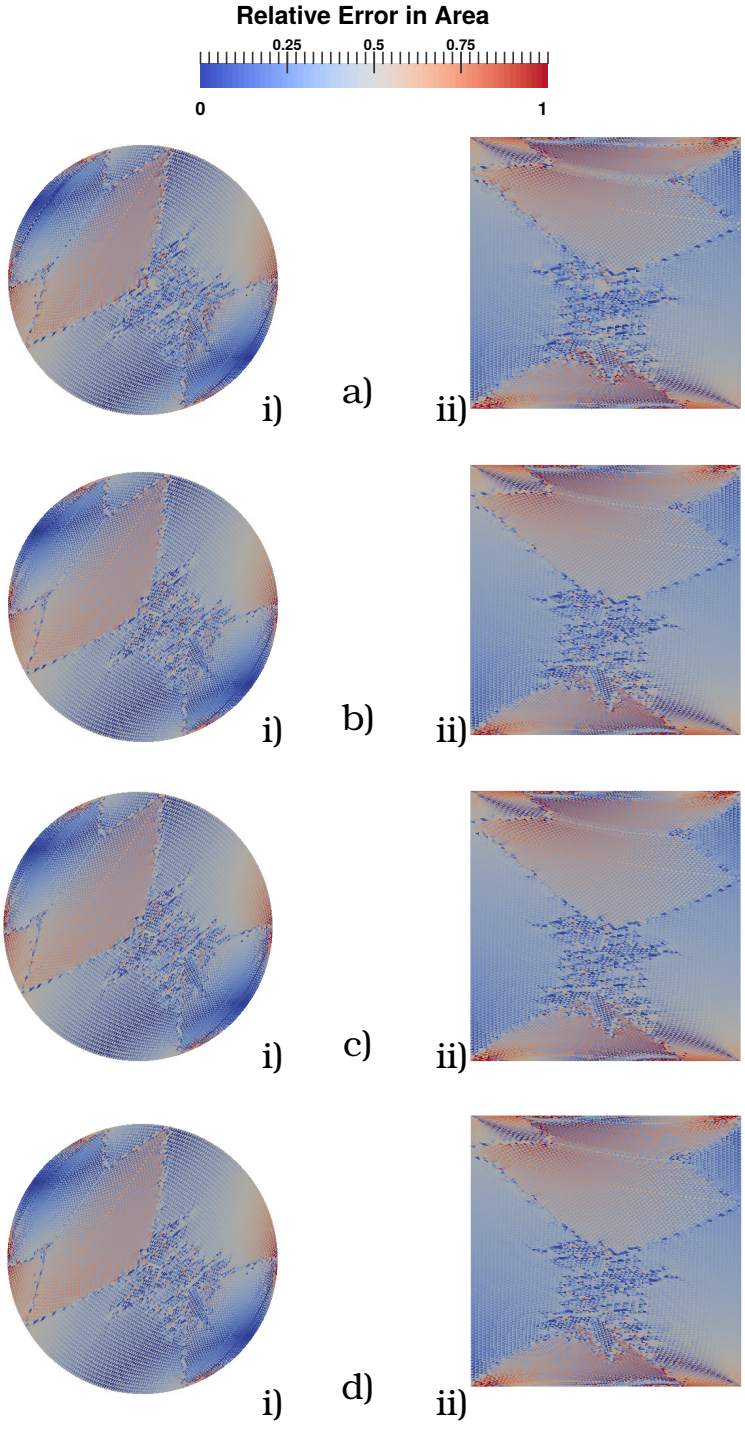

Figure 5.13: Distribution of relative error in area for fixed boundary parameterisations a) Uniform b) DAP c) DHP d) Mean Value, for the flexible sheet meshed using 17882 faces, using circular and square parameter domains.

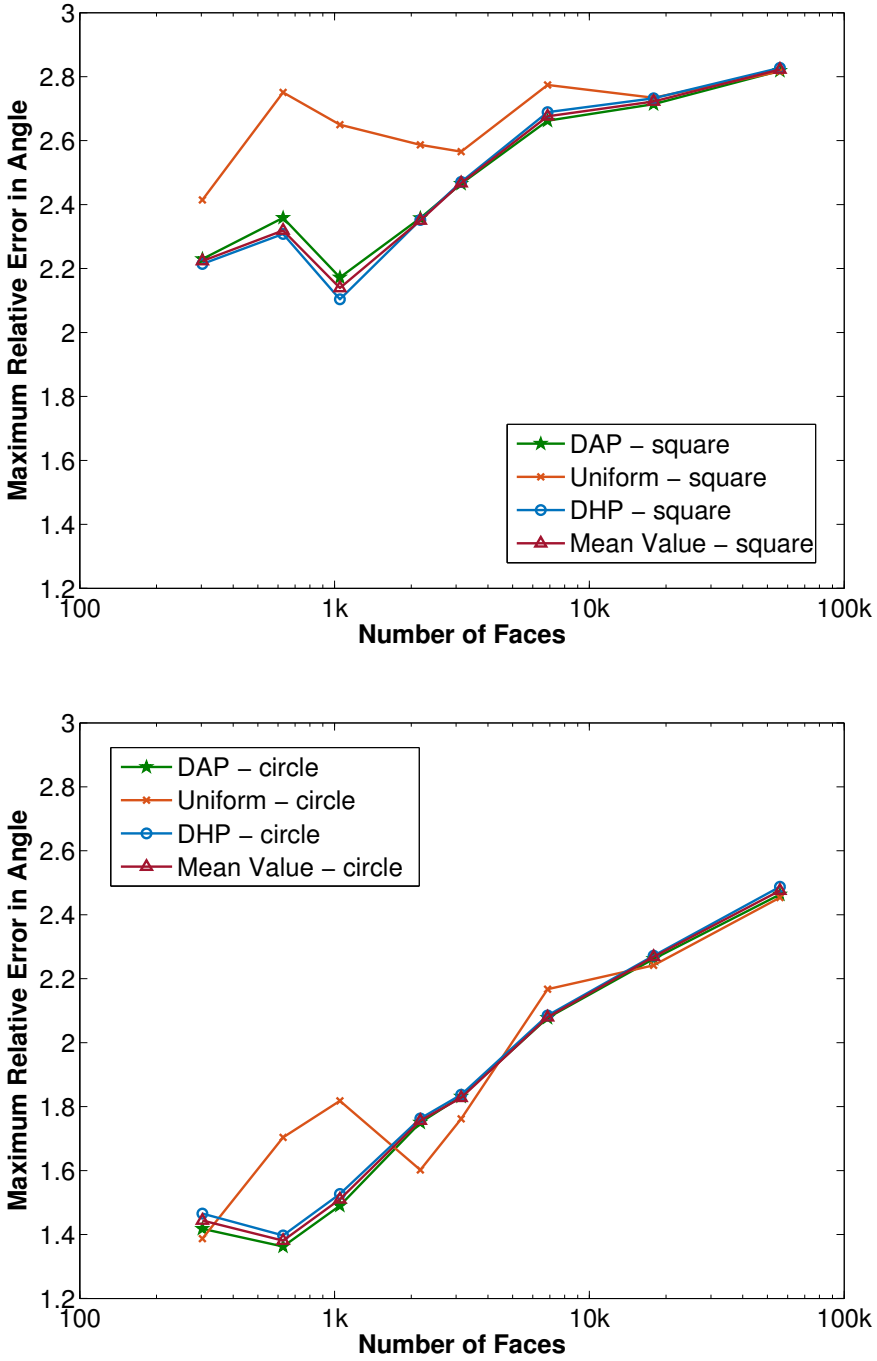

Figure 5.14: Maximum relative error in angle experienced by a triangle as the mesh density is increased for the fixed boundary parameterisations of the flexible sheet : a) shows the error using a square parameter domain, and b) uses a circular domain.

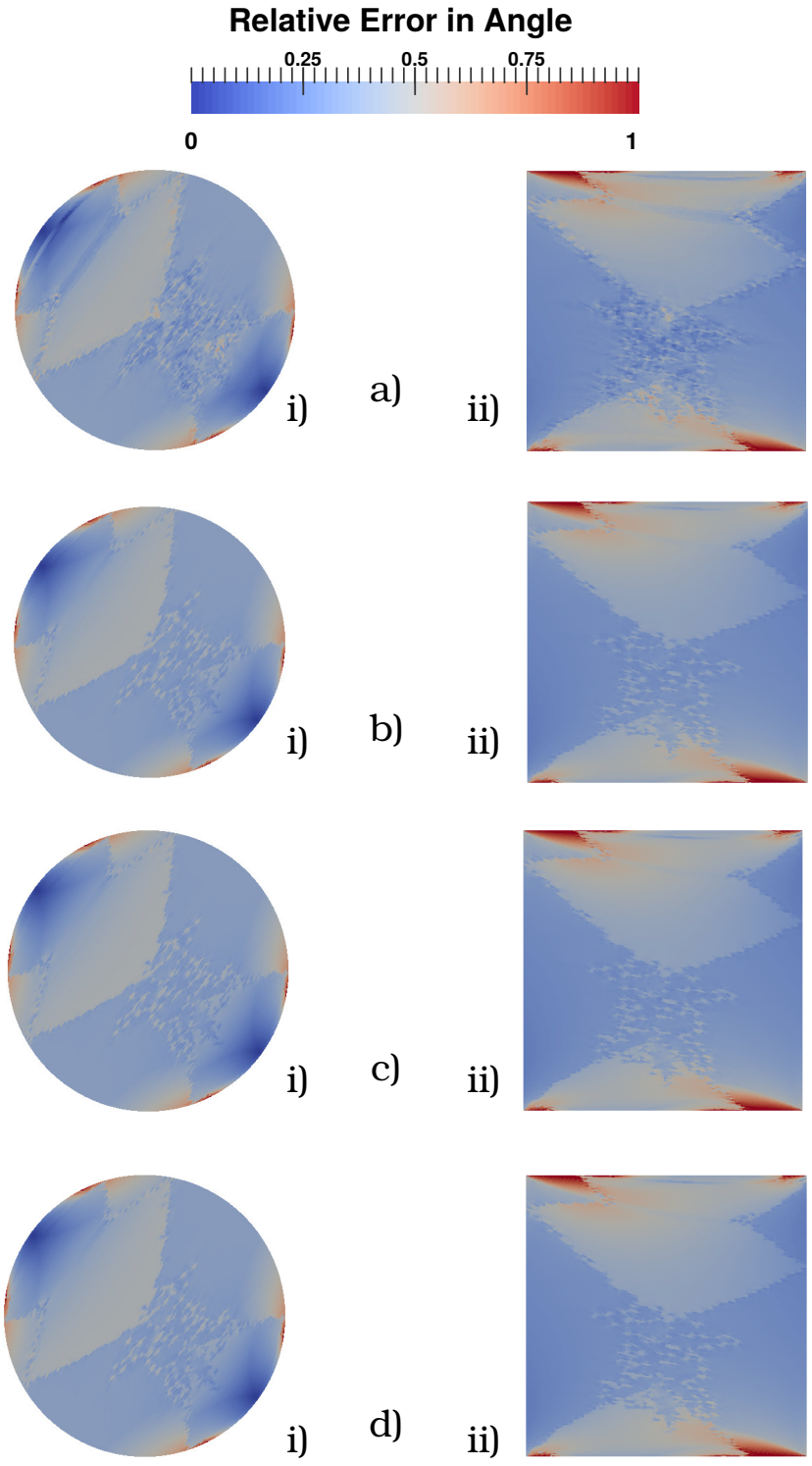

Figure 5.15: Distribution of error in angle resulting from fixed boundary parameterisations: a) Uniform b) DAP c) DHP d) Mean Value, for the flexible sheet meshed using 17882 faces, using i) circular and ii) square parameter domains.

to that of the the area distortion induced by each method, such that the maximum deviation from the original angles in the 3D mesh triangles are seen at the boundary of each parameter domain. However, the magnitude of error in angle for the interior faces appears to be less than the error in area, though the 25% to 50% relative error is still significant when preservation of the shape of each triangle is desired. The distribution of distortion, for both angle and area, highlights the importance of choosing a boundary that at least modestly describes the boundary of the original geometry in the 3D domain.

Figure 5.16 demonstrates how the resultant distortion of area and angle impacts the link lengths of the UTLM network for the fixed boundary parameterisation methods. Shown is the parameterisation of the flexible sheet constituting 17882 faces, using a) Uniform, b) DAP c) DCP and d) Mean Value methods, fixing the planar boundary to be i) circular and ii) square. Distortion to the link lengths in each case is shown to exceed a 100% error from the original mesh in 3D space, which will result in large differences in the electrical parameters of each link line.

With this in mind, the investigation now focuses on the free boundary techniques. Figure 5.17 shows the maximum relative error in area experienced by a triangle as the mesh density is increased, using the free boundary parameterisation methods, for the flexible sheet. In contrast to the fixed boundary techniques, the maximum error does not exceed 0.5%, which results from a parameterisation of the coarsest mesh, constituting 303 faces. The Least Squares Conformal Map presents a good attempt in matching ABF at minimizing authalic distortion. The maximal relative error falls to 0.05% parameterising the mesh constituting 3140 faces using ABF and LSCM. This relationship diminishes slightly as the mesh is subdivided further, resulting in a maximum relative error of 0.25% for the LSCM, while ABF achieves 0.13%; however the variation in maximal distortion is just over a tenth of a percent. It should also be noted that the maximum error imposed on the area of individual

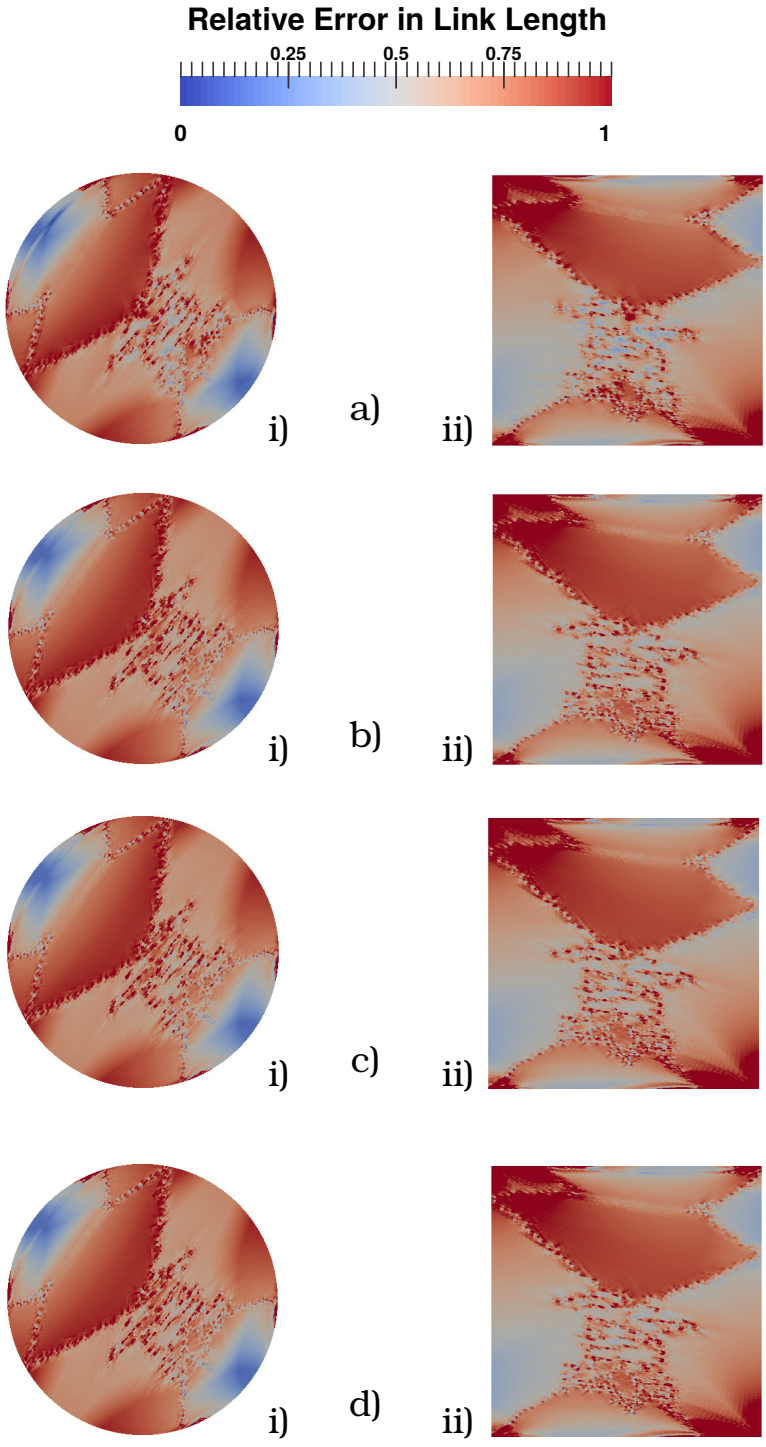

Figure 5.16: Distribution of relative error in link length for fixed boundary parameterisations a) Uniform b) DAP c) DCP d) Mean Value, for the flexible sheet meshed using 17882 faces, using (i) circular and (ii) square parameter domains.

triangles is less than half of a percent, regardless of the mesh density and free boundary parameterisation methodology used.

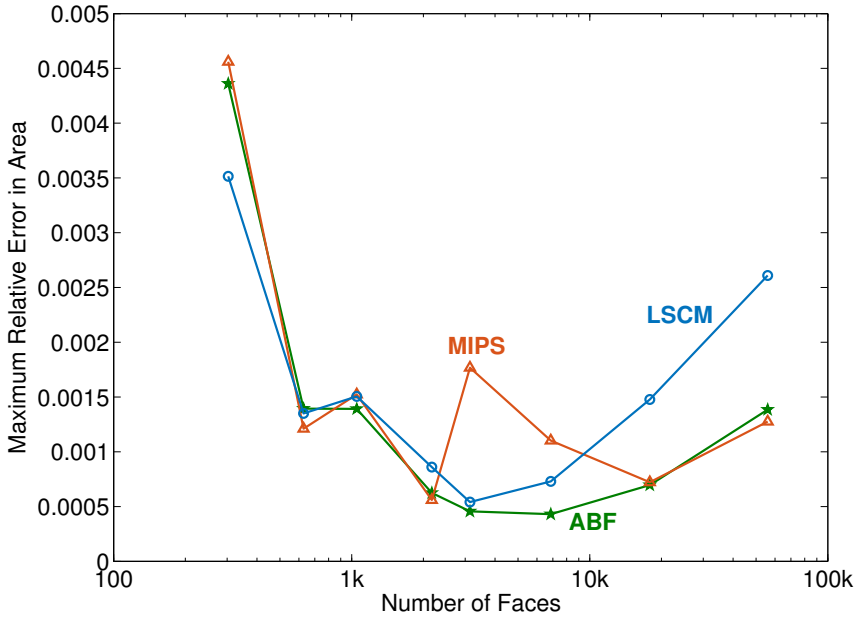

Figure 5.17: Maximum relative error in area experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible sheet

Figure 5.18 provides a graph showing the maximum relative error in angle induced by the free boundary parameterisation methods as mesh density is increased. The trend in this graph coincides almost exactly with the graph for maximum relative error in area in Figure 5.17. The maximum error in angle is shown to be consistently less than 0.05% across all of the meshes and LSCM, the linear parameterisation method, performs well at minimising angular distortion when compared to the non-linear parameterisation formulations of ABF and MIPS.

Figure 5.19 shows how the relative error in area of individual triangles is distributed across the mesh, for the flexible sheet constituting 17882 faces. Figure 5.19 a) is the parameterisation using ABF, b) MIPS and c) LSCM. For each of these parameterisations, i) shows the distribution on a scale of 0 to 100%, such that it can be compared to the fixed boundary parameterisations shown in Figure 5.13. This

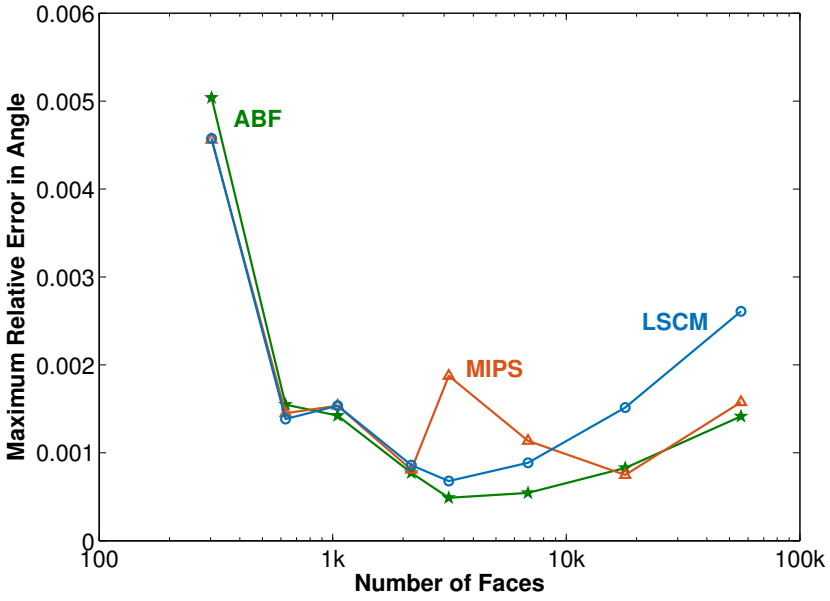

Figure 5.18: Maximum relative error in angle experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible sheet

comparison shows that the distortion in area is small compared to that induced by using a fixed boundary parameterisation method. Of course, because the maximum resultant relative error induced is less than 0.02% for this mesh, regardless of which free boundary technique is used, no detail of the distribution can be seen at this scale. Hence, Figure 5.19 ii) shows the same parameterisation using a scale range of 0 to 0.02% for each method, allowing a visualisation of how the error in area is distributed across the surface. The distribution of relative error resulting from each parameterisation is clearly not the same. The bottom of each mesh displays an almost uniform distribution of very close to zero, and increases as the top of the planar region is approached. The increase in relative error does not appear to be gradual, but presents itself as distinctly abrupt bands, with LSCM exhibiting the most relative error at the top, where small amounts of red can be seen, this error is still just over one tenth of a percent.

Comparing the distribution of area distortion with the distribution in angle distor-

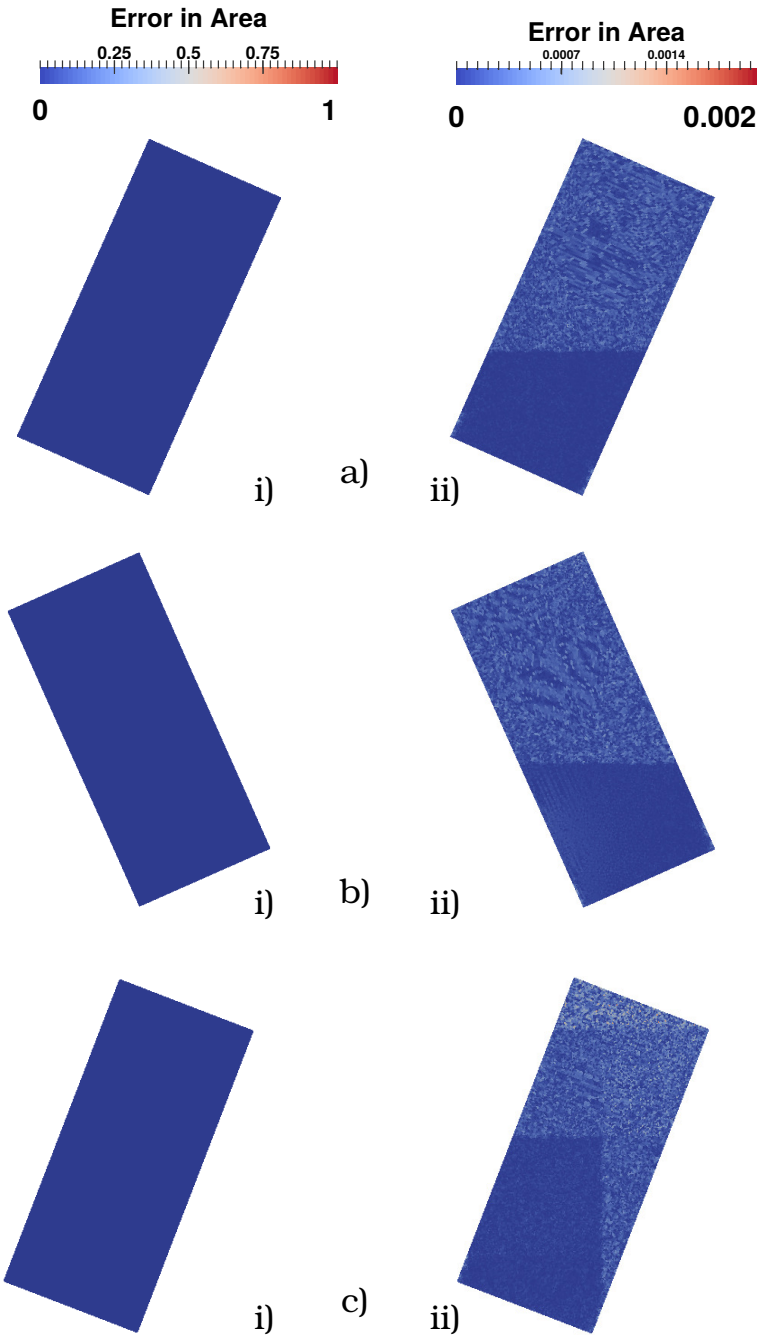

Figure 5.19: Distribution of relative error in area induced by free boundary parameterisations a) ABF b) MIPS c) LSCM for the flexible sheet meshed using 17882 faces. i) shows the error on a scale from 0 to 1. ii) shows the distribution on a scale from 0 to 0.002

tion, shown in Figure 5.20, a similar result can be seen. The parameterisations are ordered in the same manner as Figure 5.19 where the parameterisations are a result of a) ABF b) MIPS and c) LSCM free boundary methods for the flexible sheet meshed using 17882 faces. Figure 5.20 i) in each case, again, shows that the relative error is minimal in comparison to the fixed boundary parameterisations, displayed in Figure 5.15. Changing the scale in Figure 5.20 ii), such that the range is from 0 to 0.02%, presents the same distribution of angle relative error as the distribution in area. The key difference is the magnitude of the distortion, which is slightly less than that of area. This can be primarily seen comparing the area and angle metrics for the LSCM method, where relative error in angle is confined to less than one tenth of a percent, and the magnitude of authalic distortion is slightly greater than this.

Figure 5.21 shows the median relative error in link length plotted against the mesh density. The median shows the centre of the numerical data set, such that 50% of the parameterised triangles constituting each mesh exhibit an error in link length less than that shown in the graph. All of the free boundary techniques, except for the MIPS method, ensure that at least half of the link lengths constructing the UTLM network are distorted by no more than 0.15%. In most cases, this is reduced to a relative error very close to zero. It can be seen that the medians for all of these techniques are incredibly similar. The MIPS method results in a higher median value for meshes above 2000 faces, though 50% of the mesh elements still exhibit less than a 0.8% difference relative to the corresponding triangles in the 3D domain.

Using the median as opposed to the average removes the affect of any outlying relative errors in the data set, and provides a better impression of the distribution of error in UTLM link lines. The maximum error in link length is an important metric to measure, however, as this can negatively impact the minimum link length

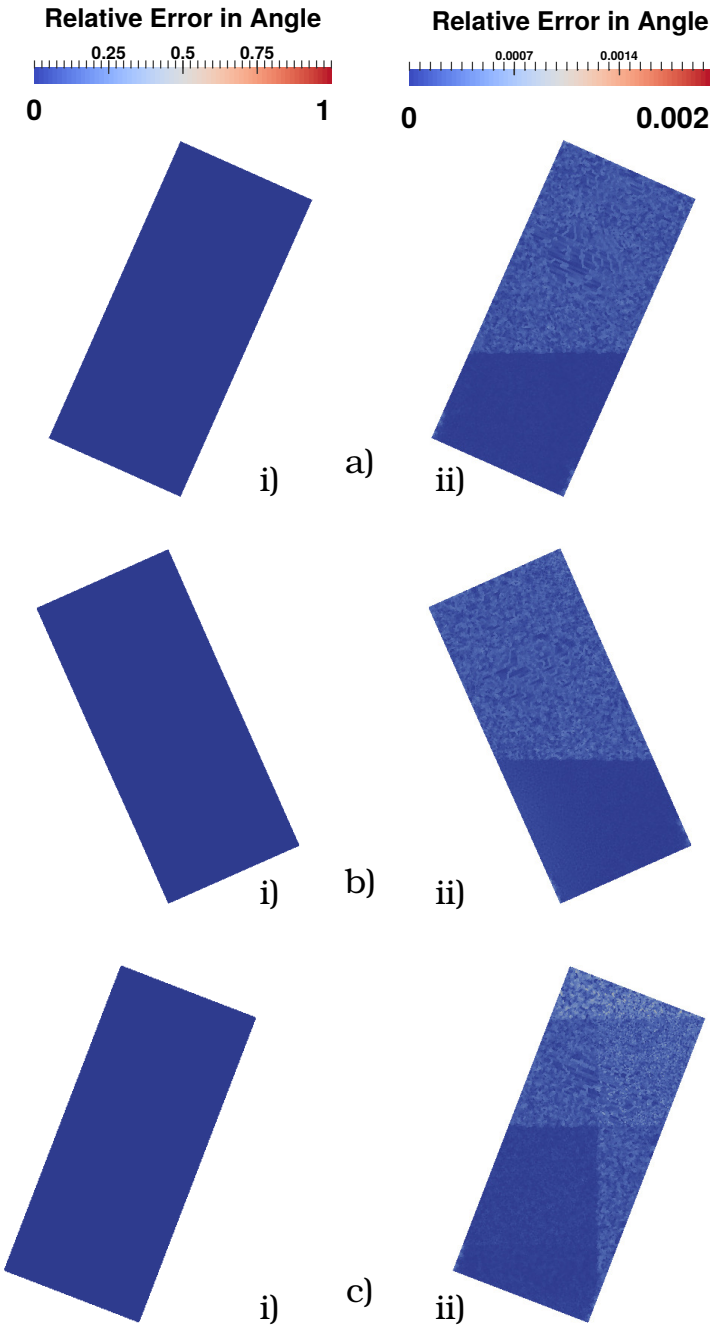

Figure 5.20: Distribution of relative error in angle induced by free boundary parameterisations a) ABF b) MIPS c) LSCM for the flexible sheet meshed using 17882 faces. i) shows the error on a scale from 0 to 1. ii) shows the distribution on a scale from 0 to 0.002

across the entire UTLM network, affecting the maximum allowable time step that can be used for a UTLM simulation. The maximum error to an individual link length constituting the TLM network is shown in Figure 5.22. The linear, and thus simpler, implementation offered by the LSCM performs very similarly to the ABF method, with respect to minimizing distortion to the link lengths. Neither of these methods induce a relative error greater than 4% when parameterising extremely coarse meshes, and this maximum error reduces to less than 1% and 0.5% for LSCM and ABF respectively, as more faces are used to describe the geometry. The MIPS method does not behave in the same manner as the ABF and LSCM parameterisations. The mesh faces experience more distortion to link lengths than the other free boundary methods as mesh density increases, however this is kept to 5% in the worst case, pertaining to the mesh constituting 55938 faces.

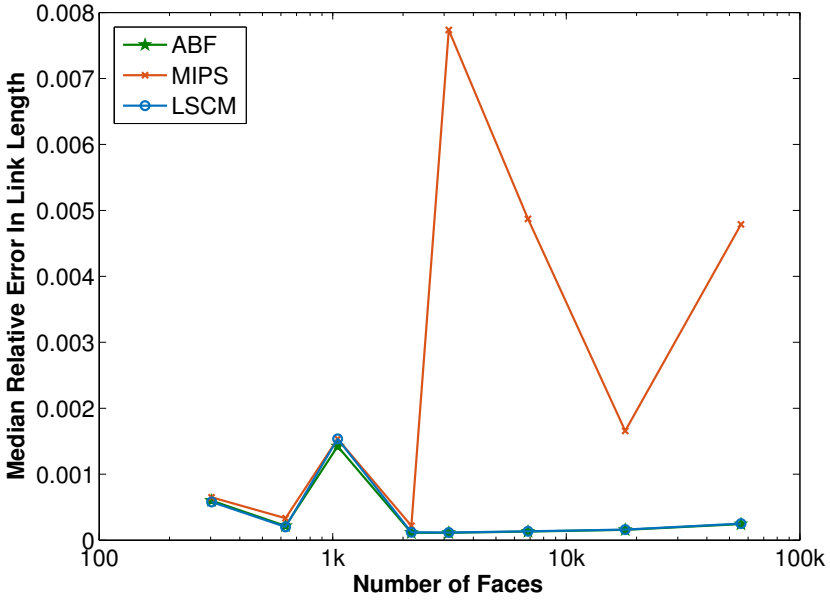

Figure 5.21: Median relative error in link length experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible sheet

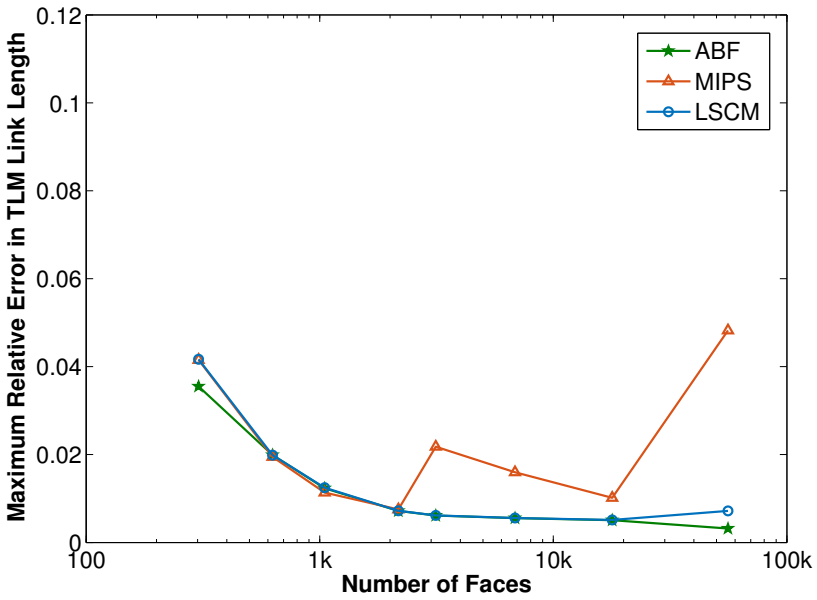

Figure 5.22: Maximum relative error in link length experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible sheet

5.2.1 Distorting the flexible sheet

The flexible sheet, shown in Figure 5.1 b) i) is now arbitrarily distorted into a new shape in the 3D domain, giving rise to Figure 5.1 b) ii), maintaining the original dimensions of the geometry, but modifying the curvature. This facilitates a comparison between this new curvature and the original curvature of the flexible sheet. This newly deformed sheet is meshed using 284, 560, 926, 1974, 3078, 6206, 15802 and 50298 faces. Three of these meshes are shown in Figure 5.23, where a) is the the mesh constituting 926 faces, b) 3078 faces and c) 15802 faces.

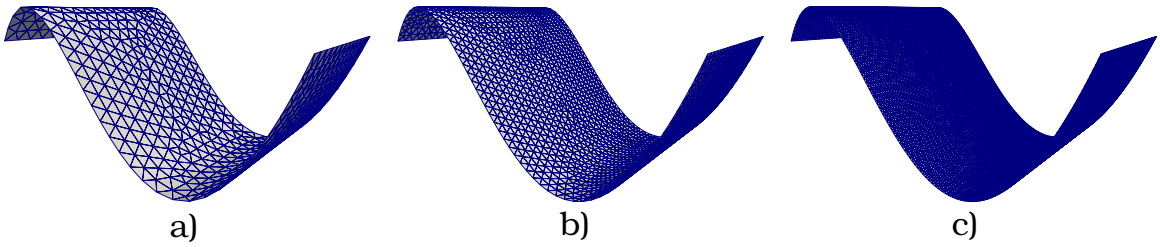

Figure 5.23: Flexible Metallic Sheet, deformed into a new curvature and meshed using a) 926, b) 3078 and c) 15802 faces

Figure 5.24 shows a graph of the relative error in area against the number of mesh faces. Comparing this to the graph in Figure 5.17 which displayed the relative error in area for the original flexible sheet, the trend of the relative error does not reveal large differences when the curvature is changed, however,the magnitude of the maximum relative error has increased slightly, especially for coarse meshes.

This same comparison is made for the relative error in angle. Figure 5.25 shows the relative error for the newly deformed sheet for the free boundary techniques. Comparing this graph with the equivalent for the original curvature in Figure 5.18, shows that the error in angle is within the same range. Variations exist, but these are very small (in the order of a tenth of a percent) compared to the magnitude of change to the curvature of the geometry. It should be noted that while every effort was made to incrementally mesh the newly deformed geometry with an element

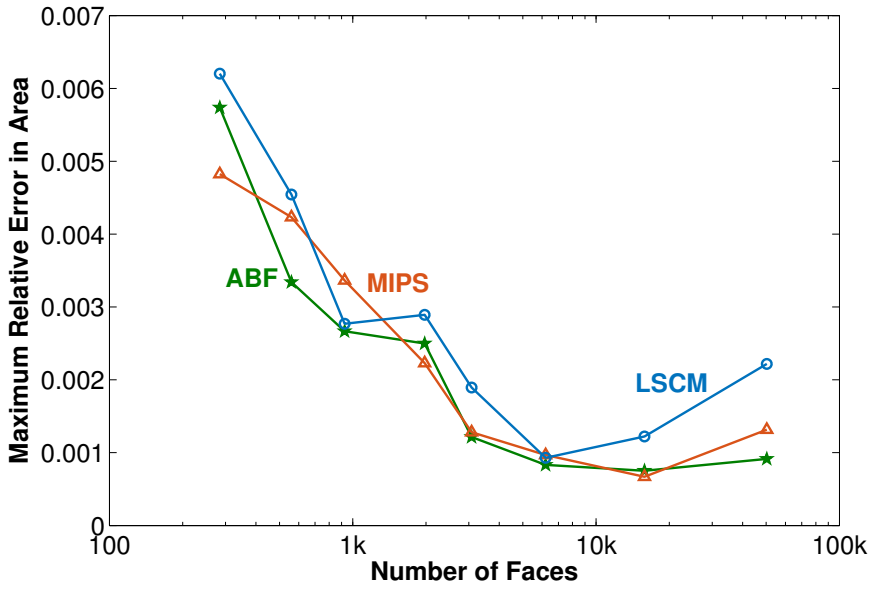

Figure 5.24: Maximum relative error in area experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the deformed flexible sheet, with new curvature

density that matches the original test case, it is very difficult to match these exactly, and as a result the number of faces constituting the mesh will also vary. This may explain the slight differences in the induced distortions when comparing the two curvatures of the flexible metallic sheet. A different mesh will result in a different parameterisation.

Figure 5.26 shows the resultant distribution of angle and area relative error of the free boundary parameterisations for the newly deformed flexible sheet. The parameterisations are performed on the mesh constituting 15802 faces shown in Figure 5.23, where Figure 5.26 a) is the parameterisation resulting from the ABF method, b) the MIPS method, and c) the LSCM method. In each case, i) shows the distribution in relative area distortion on a scale from 0 to 0.002, matching the scale from the original flexible sheet test case in Figure 5.19 ii). It can be seen from this comparison that the distribution in area distortion is very similar to the previous curvature of

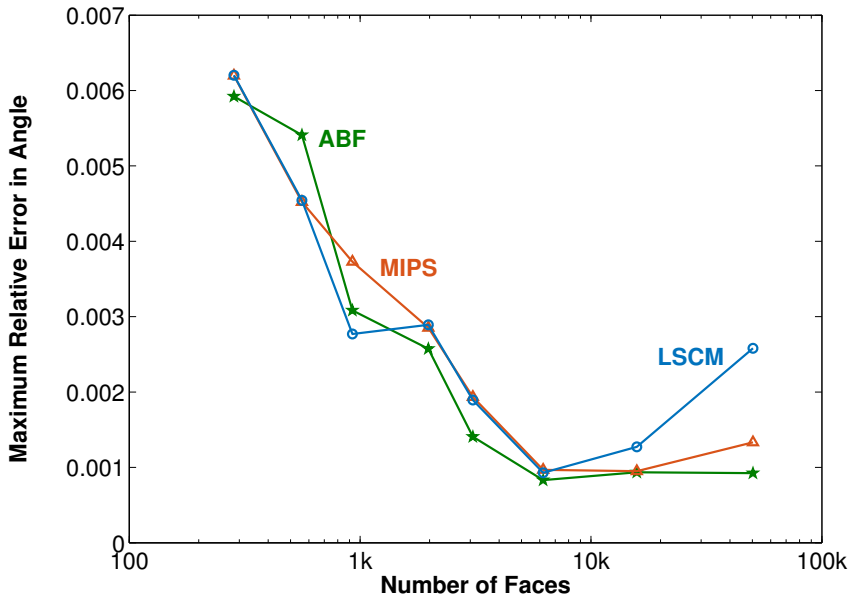

Figure 5.25: Maximum relative error in angle experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the deformed flexible sheet

the flexible sheet. A slight increase in relative error can be seen towards the bottom of each parameterisation result at the boundary, highlighted with black circles. This coincides with the new apexes of the newly curved surface. This can also be seen when comparing the distribution in angle distortion for the meshes in Figure 5.26 ii) with Figure 5.19 ii). It is also noticed that the distribution in angle and area, produced by the LSCM method in c) appears to be consistent with the original flexible sheet test case, though flipped about an axis. This is due to the choice of the two boundary vertices used to fix the parameter domain in 2D space; this property is consistent with the literature which states that a differing LSCM parameterisation can result based on the choice of these two vertices [5.2].

In a similar manner to the original flexed surface, the effect the area and angular distortion, resulting from the free boundary parameterisation methods, has on the UTLM link lengths is shown in Figure 5.27 as mesh density is increased. The graph shows that an increase in the maximum distortion in link length to 11%

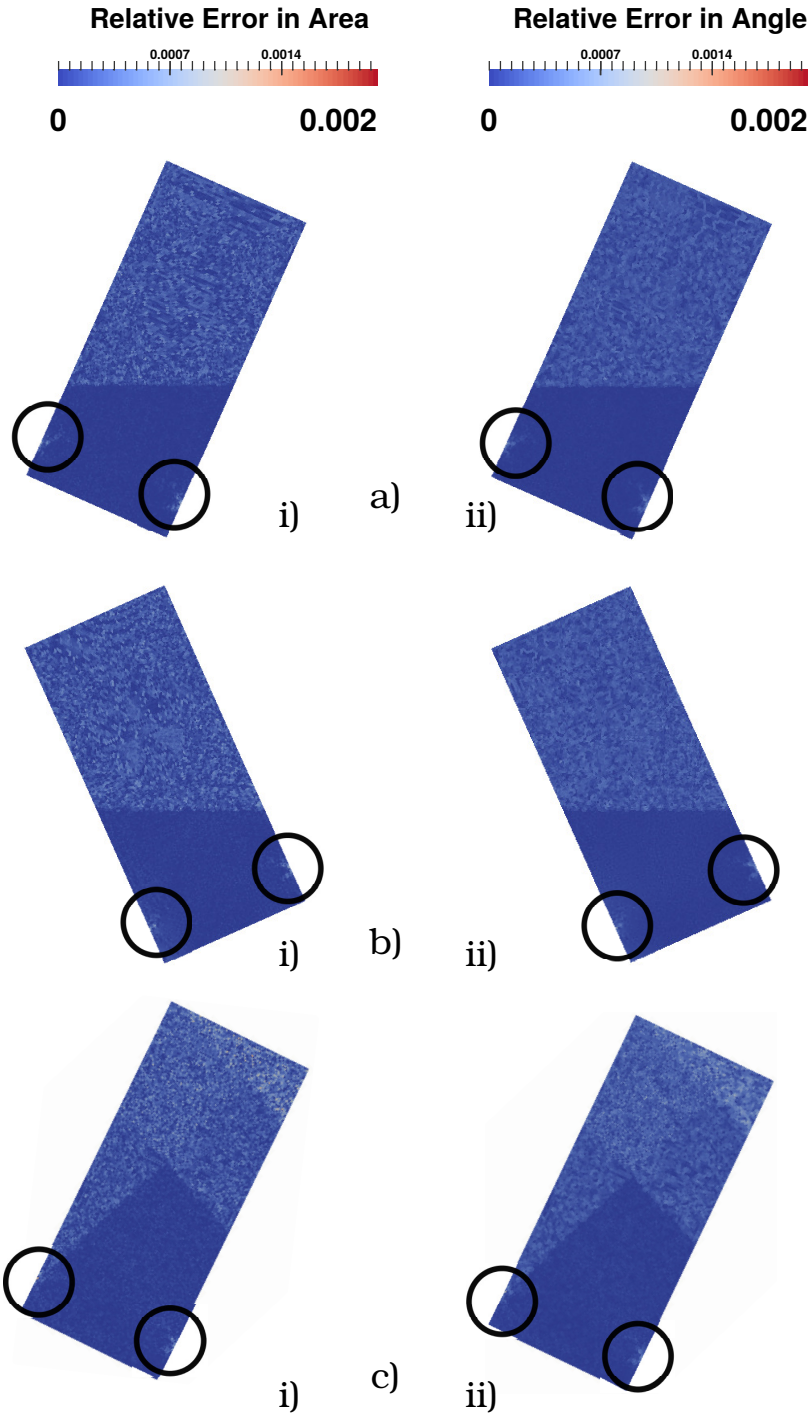

Figure 5.26: Distribution of relative error in i) area and ii) angle induced by free boundary parameterisations a) ABF b) MIPS c) LSCM for the flexible sheet meshed using 15802 faces. The scale represents the relative error from 0 to 0.002.

for an extremely coarse mesh constituting 300 faces, for all of the free boundary parameterisation methods. The induced distortion is rapidly reduced with increasing mesh refinement, better converging with the trend seen for the original curvature of the flexible sheet in Figure5.22. MIPS demonstrated a decrease of 1.5% at the upper limit of mesh refinement, for the mesh consistuting 50298 faces, but still did not perform as well as the other free boundary techniques at minimising the distortion to the TLM link lengths.

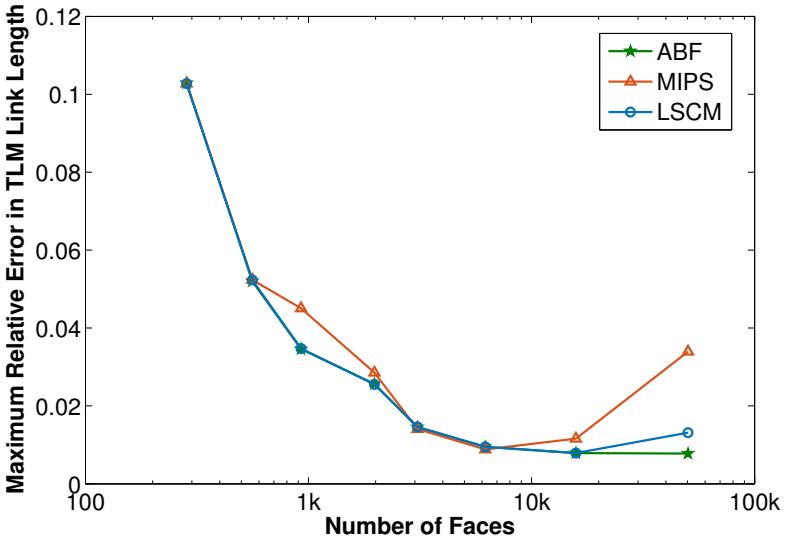

Figure 5.27: Maximum relative error in link length experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the newly deformed flexible sheet

The median error induced by the link lengths constituting the TLM network is shown in Figure 5.28. LSCM provides a median error which almost exactly matches that obtained using the ABF method. A comparison of this graph and that representing the median link length error for the original curvature, presented in Figure 5.22, also demonstrates that the change in curvature has little effect on the resultant parameterisations produced by ABF and LSCM. Both of these techniques provide a median distortion between 0.1% and 0.003%. MIPS provides a similar median of link length distortion when parameterising the newly deformed flexible sheet,

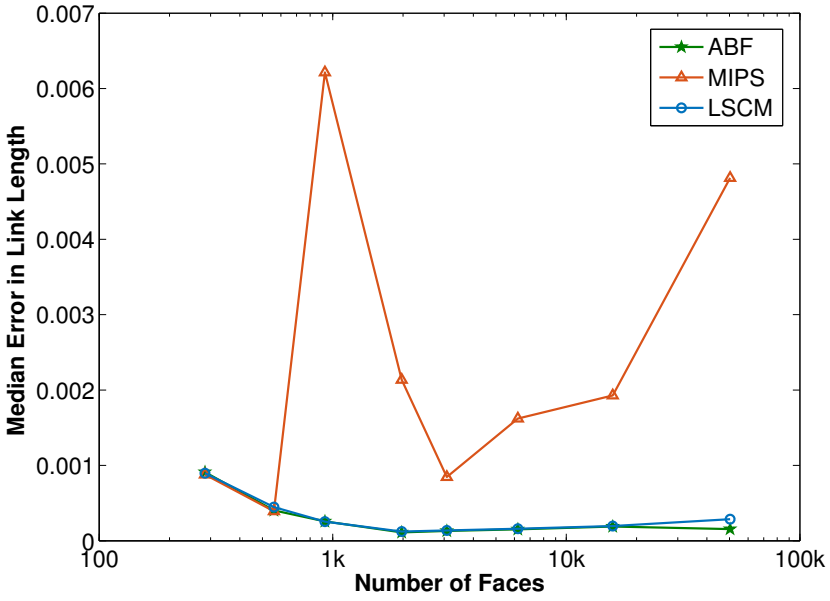

Figure 5.28: Median relative error in link length experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the newly deformed flexible sheet

compared to the original curvature. However, with both curvatures, the median level of distortion induced to the link lengths by MIPS is consistently greater than that provided by the ABF and LSCM methods.

The effect of the higher median distortion to link lengths exhibited by the MIPS method can be seen in Figure 5.29. This figure shows the distribution of the link length error, for i) the original curvature of the flexible sheet meshed using 17882 faces, and ii) the new curvature meshed using 15802 faces. The scale shows the relative error distribution from 0 to 0.002. Figure 5.29 a) is the resultant parameterisation using ABF, b) is using the MIPS method, and c) is using the linear LSCM method. The MIPS method shows that more link lengths constituting the TLM network have experienced a greater error relative to the original link lines in the 3D domain. Although the maximum magnitude of relative error in link lines has not changed significantly between the two curvatures, a change in curvature induces

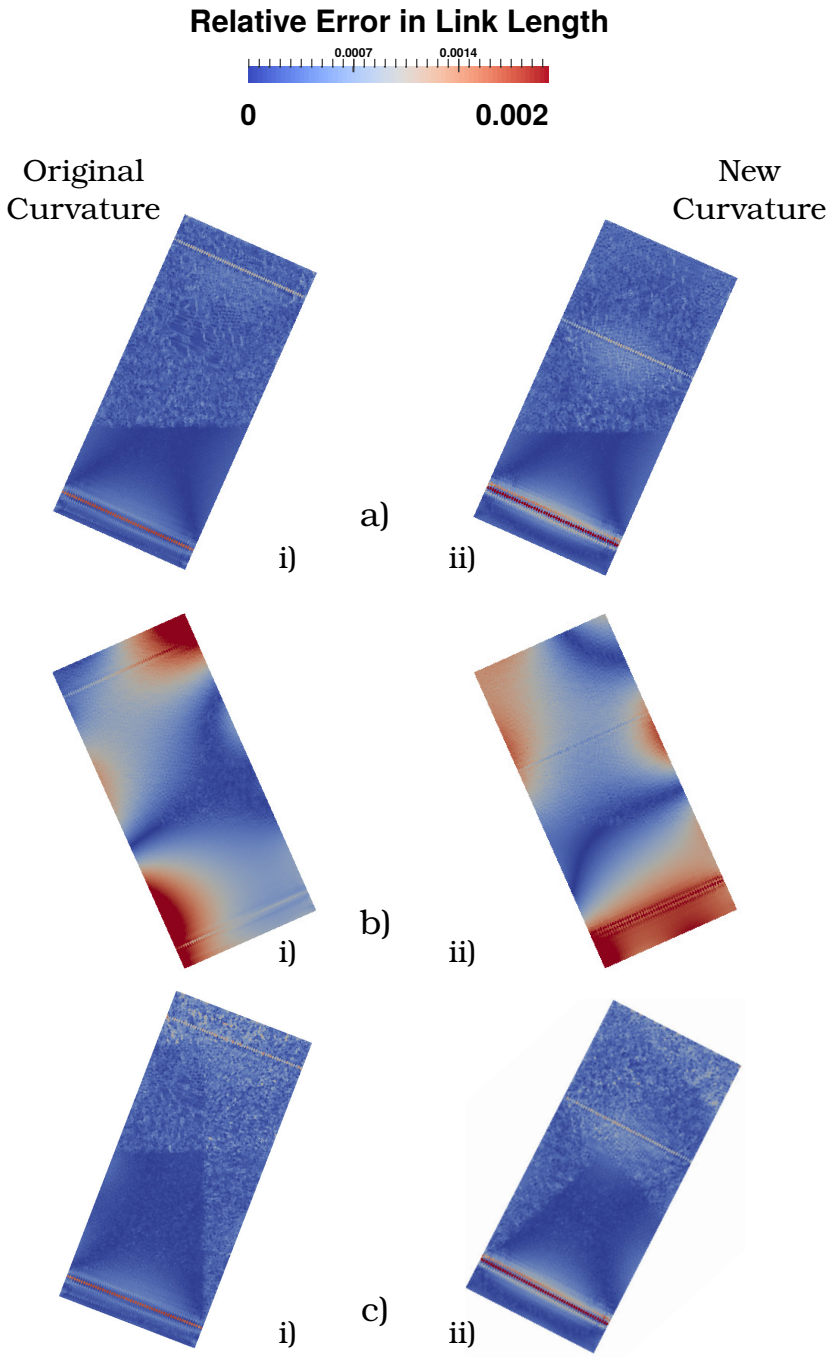

Figure 5.29: Distribution in relative error in link length for the metallic flexible sheet. a) is the parameterisation using ABF, b) MIPS and c) LSCM. i) are the parameterisations of the flexible sheet using its original curvature, and ii) is the new curvature.

a notable change in the distribution of this error.

The ABF and LSCM schemes provide a very similar distribution in relative error, which is mainly concentrated at the maxima and minima of the original flexed geometries in the 3D domain. By changing the curvature, the distribution of the distortion is seen to realign with the new maxima and minima. This distribution in error shows that the resultant parameterisations of these geometries produced by the linear and simpler implementation of the LSCM method compare very well with those obtained using the non-linear formulation of the ABF method. MIPS, on the other hand, fails to demonstrate as good a performance as the other free boundary methods. Combined with its non-linear methodology and comparatively arduous implementation, there is little reason to implement this method for the purposes of UTLM, over the other free boundary techniques. The next test case presented aims to further validate this.

5.3 Case 3: Flexible PCB

The next test case is a flexible PCB, shown in Figure 5.1 c) i). This surface is non-uniform in material properties, consisting of 3 metal wires, separated by substrate.

The flexible PCB was meshed using 1124, 1760, 2092, 2948, 5796, 14454, 44456 and 74384 faces. Figure 5.30 shows the PCB meshed using a) 1760 b) 2948 and c) 14454 faces.

In a similar fashion to the preceding test cases, the maximum relative error in area for each mesh is shown in Figure 5.31. The maximum relative error in area induced for the flexible PCB approximated using around 1000 faces is less than 0.0003, or 0.03%. As the mesh is refined, this relative error rises to 0.3% for extremely

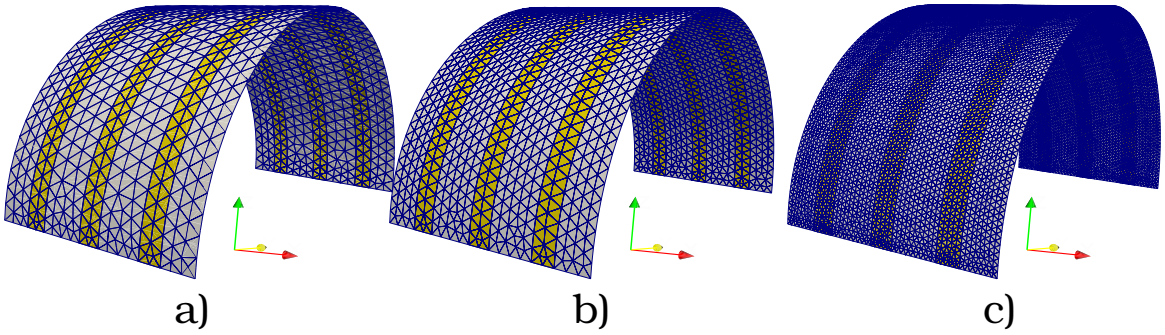

Figure 5.30: Flexible PCB meshed using a) 1760 b) 2948 and c) 14454 mesh faces.

fine meshes, which constitute 75,000 faces, when the LSCM method is used. The resultant parameterisations produced by the non-linear methods, ABF and MIPS, demonstrate the least amount of distortion as the mesh becomes more refined, as area distortion increases from 0.05% to 0.1% and 0.13% respectively.

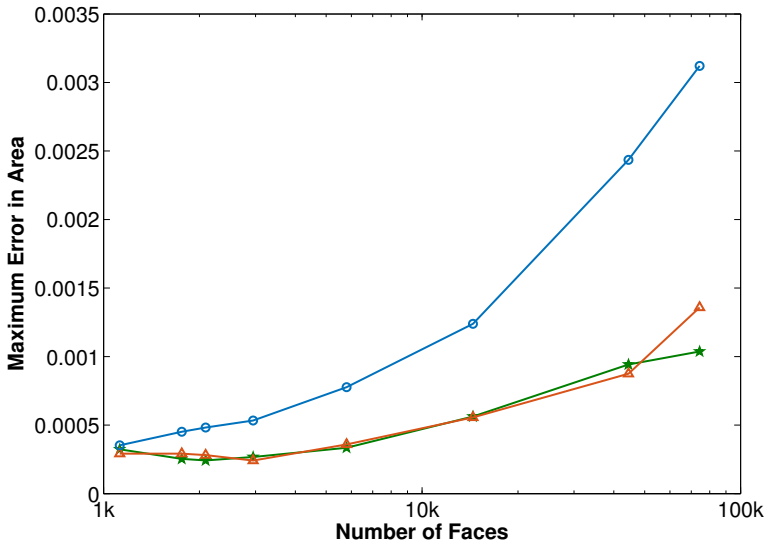

Figure 5.31: Maximum relative error in area experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible pcb

Figure 5.32 shows the maximum relative error in angle compared with the angles of individual triangles in the 3D geometry. The maximum deviation to angles follows a very similar trend to that observed for the area. All free boundary methods under test resulted in a maximum relative error of 0.05%, which slowly rises as the mesh

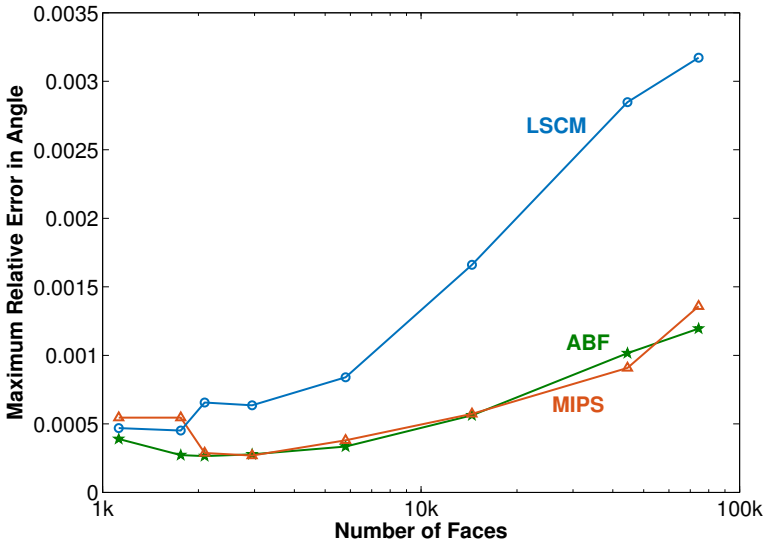

Figure 5.32: Maximum relative error in angle experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible pcb

becomes more densely populated with triangles, to 0.11% and 0.13% when ABF and MIPS are respectively used; the corresponding relative errors for the LSCM parameterisations rise to 0.32%.

The effect to the link lengths that results from the deformation of the metric properties of the individual faces that constitute a mesh is shown in Figure 5.33 and Figure 5.34. Figure 5.33 shows a plot of the median relative error in TLM link lengths as the mesh density is increased. Utilising ABF and LSCM demonstrates that at least half of the link lengths constituting the mesh result in a relative error in the range of 7×10^{-6}% to 0.003%, regardless of the mesh density. MIPS provides a slightly more unpredictable outcome, as the mesh density is varied, however the graph demonstrates that at least half of parameterised link lengths deviate by a maximum error of 0.27%, relative to their corresponding lengths in the 3D domain.

Taking into account the maximum magnitude of distortion, Figure 5.34 shows the

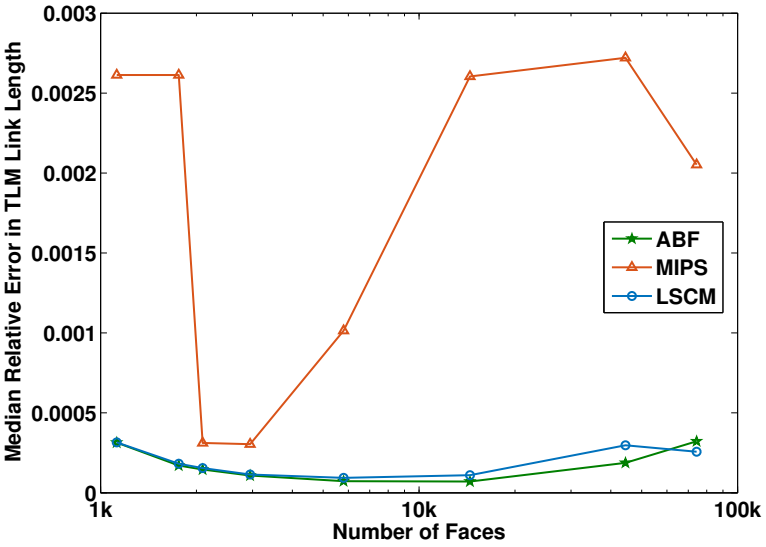

Figure 5.33: Median error in TLM link lengths experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible PCB

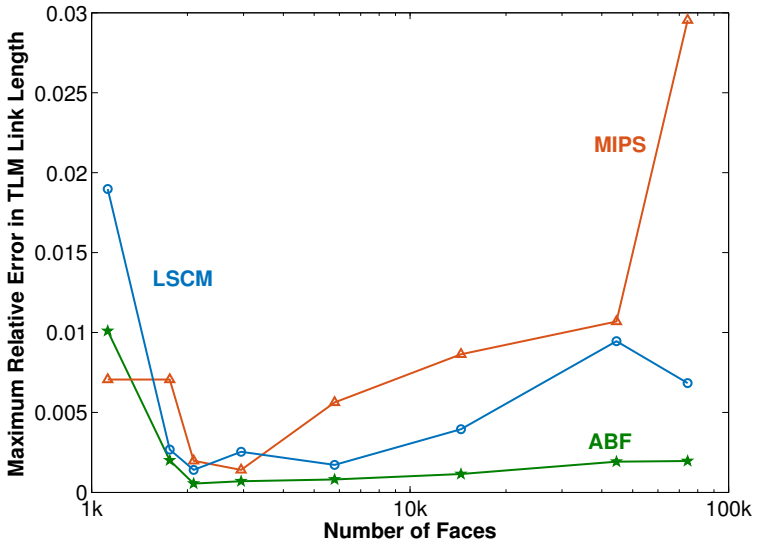

Figure 5.34: Maximum error in TLM link lengths experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible PCB

maximum relative error in link lengths as the number of mesh faces is increased. The MIPS method performs well in minimising the relative error in link lengths for coarse meshes, resulting in a maximum error of 0.7%. However as the mesh density increases beyond 10000 faces the maximum error induced escalates beyond 1% towards a 3% difference in the link lengths. Applying the ABF and LSCM methods reduced the impact on the maximum relative error in the the TLM link lines. A LSCM parameterisation of this geometry results in 1.8% maximum relative error for the coarsest mesh, consisting of 1124 faces. The error ranges from 0.15% to 0.8% for the remainder of the meshes under test. ABF provides the best parameterisation resulting in a maximum deviation from the corresponding link length of 1% for the coarsest mesh, and consistently less than 0.2% across the remainder of mesh densities. The parameterisation of the mesh constituting 2948 faces resulted in a maximum relative error in link length of 0.05%. This provides a good quality parameterisation with little difference between the original mesh elements and the parameterised triangles in the 2D domain. Figure 5.35 shows the resultant parameterisation between the flexible PCB, meshed using 2092 faces (shown in a)), and the 2D mapping after applying the ABF technique (shown in b)). Parameterisation the geometry conserves a good quality triangulation, maintaining the original connectivity of the mesh; the material properties assigned to the original geometry are also mapped to the new domain.

The angle and area distortions experienced by the parameterised meshes of the flexible PCB are minimal. It was discussed in Chapter 4 that only surfaces with zero Gaussian curvature can be parameterised isometrically, that is with zero angular and authalic distortion. Because meshes are approximations of smooth surfaces, an entirely isometric parameterisation is still not attainable, however a parameterisation that tends towards isometry can be realised. The curvature of this flexible PCB is the same as that exhibited by a cylinder, and therefore the Gaussian curvature tends to zero.

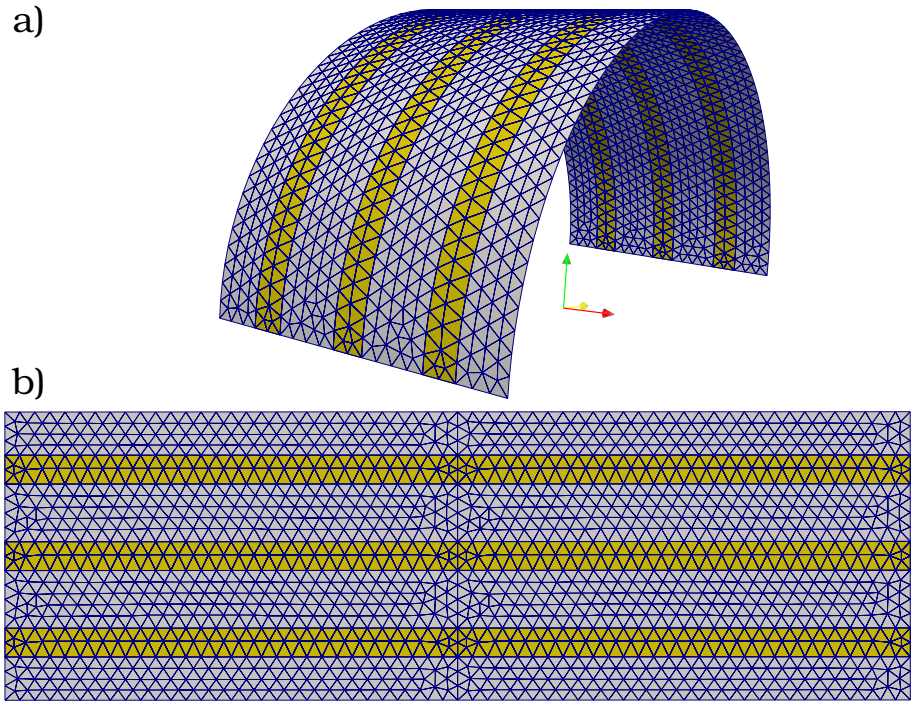

Figure 5.35: Flexible PCB in the 3D domain, meshed using 2948 faces (a) and parameterised using the ABF method (b), to the 2D flat plane

If the curvature of the PCB is changed, such that the dimensions of the geometry, if it is laid flat, remains unchanged, but the Gaussian curvature is no longer zero, a difference in distortion should result.

5.3.1 Distorting the flexible PCB

Plying the surface into a new curvature, results in a new geometry to compare with the original. The newly flexed surface was meshed using 1084, 1772, 3392,7616, 12416, 23488, 44474 and 74396 mesh faces. Figure 5.36 shows three of these meshes; a) consists of 1772 faces, b) has 3392 faces and c) constitutes 12416 faces.

The parameterisation of the flexible PCB with the new curvature can be seen in Figure 5.37, which shows the resultant parameterisation for the PCB in the 3D domain (shown in a)), meshed using 3392 faces, applying an ABF parameterisation.

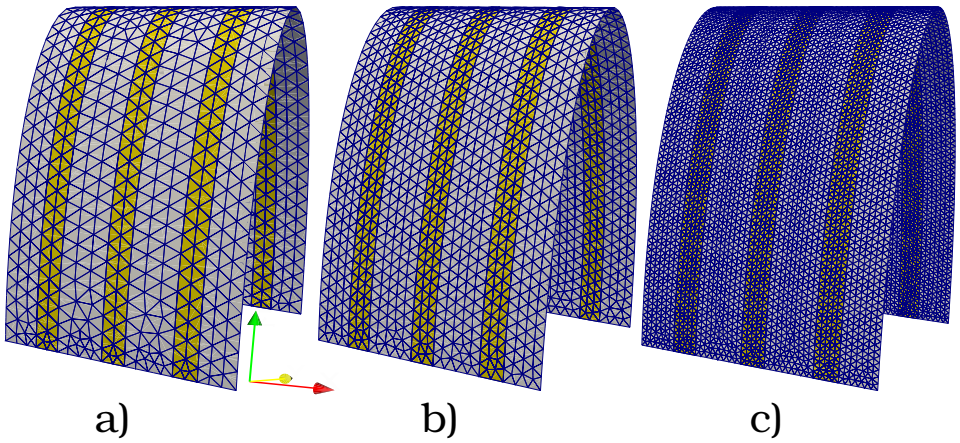

Figure 5.36: Flexible PCB flexed into a new curvature and meshed using a) 1772 b) 3392 and c) 12416 mesh faces.

Again, the parameterisation maintains a good quality triangulation, with the original connectivity intact in the 2D domain (shown in Figure 5.37 a)).

Repeating the investigations of Section 5.3 for this new curvature, Figure 5.38 shows the maximum relative error in area experienced by the meshes of this new geometry. Parameterisations of the coarser meshes do experience a higher magnitude of authalic distortion. However as the mesh is refined, the maximum magnitude of distortion falls to similar levels to that experienced by the original curvature. MIPS behaves more erratically with the maximum error in area it induces, but these differences are only of the order of a tenth of a percent.

A similar trend is witnessed for the maximum error in angle, shown in Figure 5.39. For the coarsest mesh (1000 faces) the maximum magnitude of distortion has increased from approximately 0.05% to 0.5%. This maximal magnitude is reduced to below 0.1% using ABF, and distortion is contained below 0.35% for denser meshes using any of the free boundary parameterisation techniques. ABF consistently reduces the maximum magnitude of distortion with respect to the other methods, and MIPS, again exhibits a slightly more erratic behaviour compared to the trend in conformal error for the original curvature of the geometry. Angular distortion

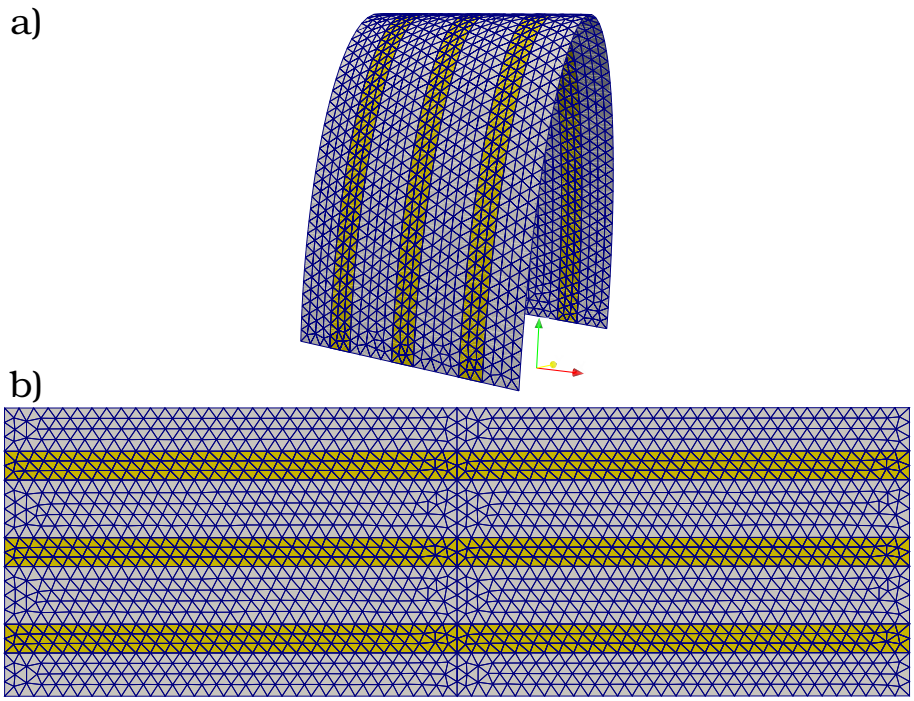

Figure 5.37: Flexible PCB in the 3D domain, flexed into a new curvature meshed using 3392 faces (a) and parameterised using the ABF method (b), to the 2D flat plane

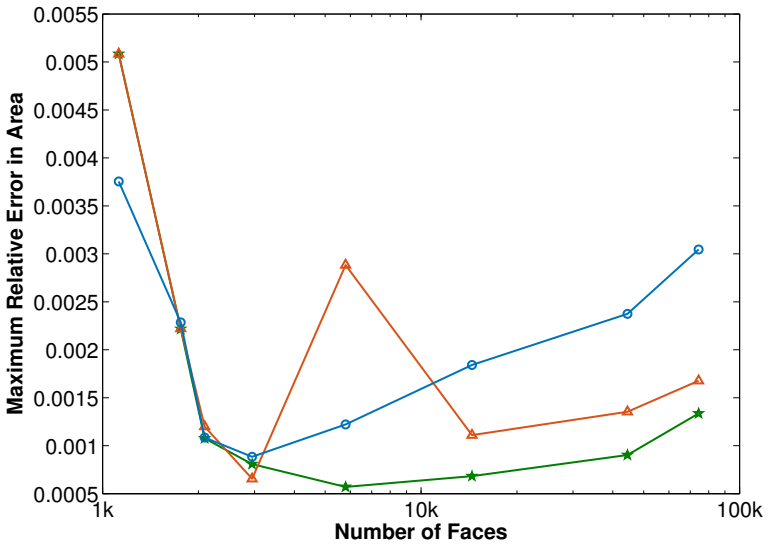

Figure 5.38: Maximum relative error in area experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the newly deformed flexible PCB

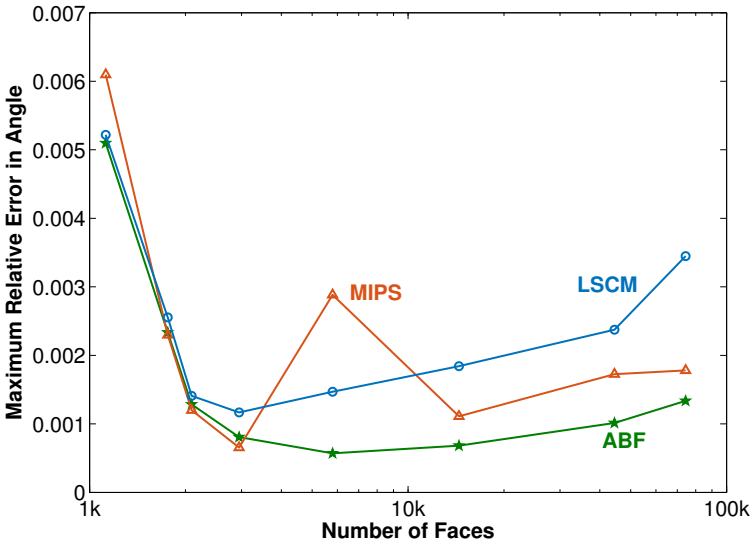

Figure 5.39: Maximum relative error in angle experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the flexible PCB with its new deformation

has increased for parameterisations of the meshes representing the new curvature, however the differences are of the order of a tenth of a percent.

The median magnitude of the distortion for angular and authalic distortion remains very similar between the two curvatures, suggesting the possibility that the increase in maximum error can be attributed to outlying values, resulting from the new higher curvature at the apex of the flexible PCB. This observation is consistent with the median and maximum relative errors in the UTLM link lengths, which are plotted in Figure 5.40 and Figure 5.41 respectively. The median error in the UTLM link lengths for the parameterisation of the newly deformed surface is very similar with the ABF and LSCM implementations. MIPS is the exception which induces a greater median error across the whole range of mesh densities studied. The change in the maximum relative error to the link length is shown in Figure 5.41. The greater magnitude of area and angle deformation for the coarser mesh induces a greater maximum error in link lengths, increasing the relative error to 5.5%. A

small increase in the maximum relative error is experienced as mesh density increases for the parameterisations by both, ABF and LSCM, both of which perform better than MIPS, as the mesh becomes more refined, up until the parameterisation of the finest mesh (74396 faces), where the largest increase in error is experienced by ABF and LSCM. LSCM experiences an increase of 2.75%, and ABF increases by 0.75%. MIPS is the exception where the largest amount of relative error in link lengths experienced by the finest mesh is reduced by 0.8%.

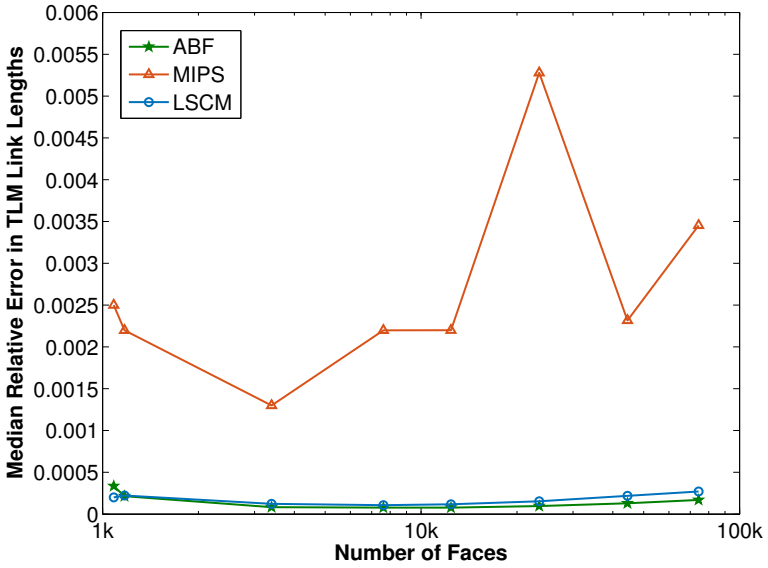

Figure 5.40: Median error in TLM link lengths experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the newly deformed flexible PCB

To better appreciate where this increase in distortion resides on a parameterised mesh, Figure 5.42 provides a comparison of the magnitude of link length error between the original curvature of the flexible PCB and the deformed variation. The original flexible PCB is meshed using 14454 mesh faces, and the newly deformed geometry is meshed using 12416 faces. Figure 5.42 a) and b) show the parameterisations resulting from the ABF and LSCM methods respectively. It can be seen that the distortion in link length induced by ABF for the original curvature (i) of the

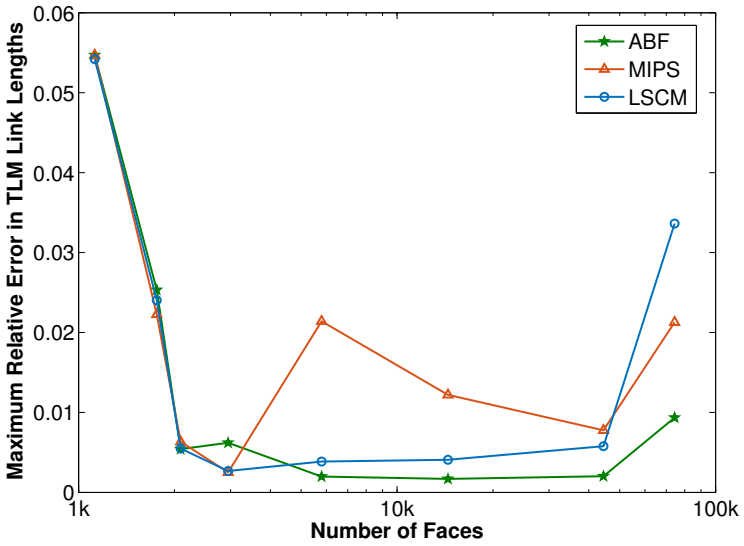

Figure 5.41: Maximum error in TLM link lengths experienced by a triangle as the mesh density is increased for the free boundary parameterisations of the newly deformed flexible PCB

geometry is minimal. LSCM induces more distortion, with the distribution showing more relative error surrounding the apex of the original curvature and the top left corner. Deforming the geometry shows an increase in link length distortion at the apex of the flexible PCB, at the point of increased curvature. As mentioned, the median error in the link length for the original curvature, shown in Figure 5.33, and the newly deformed flexible PCB (Figure 5.40) are similar. The differences in the link length distortion reside in the maximum relative error induced for the new curvature of the PCB, plotted in Figure 5.41. This is reflected in the distribution of error, shown in Figure 5.42 a)ii) and b)ii) where more distortion is witnessed at the portion of the mesh corresponding to the apex of the geometry. The median error in link length resulting from the MIPS method is consistently above 0.1% (shown in Figure 5.40) and this is reflected in the visualisation of the distribution in Figure 5.42 b), where the magnitude of the distribution is seen to be almost uniformly above the scale presented. From these results, it is observed that the ABF and LSCM schemes provide consistently better parameterisations. Due to the non-linearity of

the ABF method, one would expect this to take a longer time to arrive at a solution, compared to the linear solution provided by the LSCM method. This is investigated in the following section.

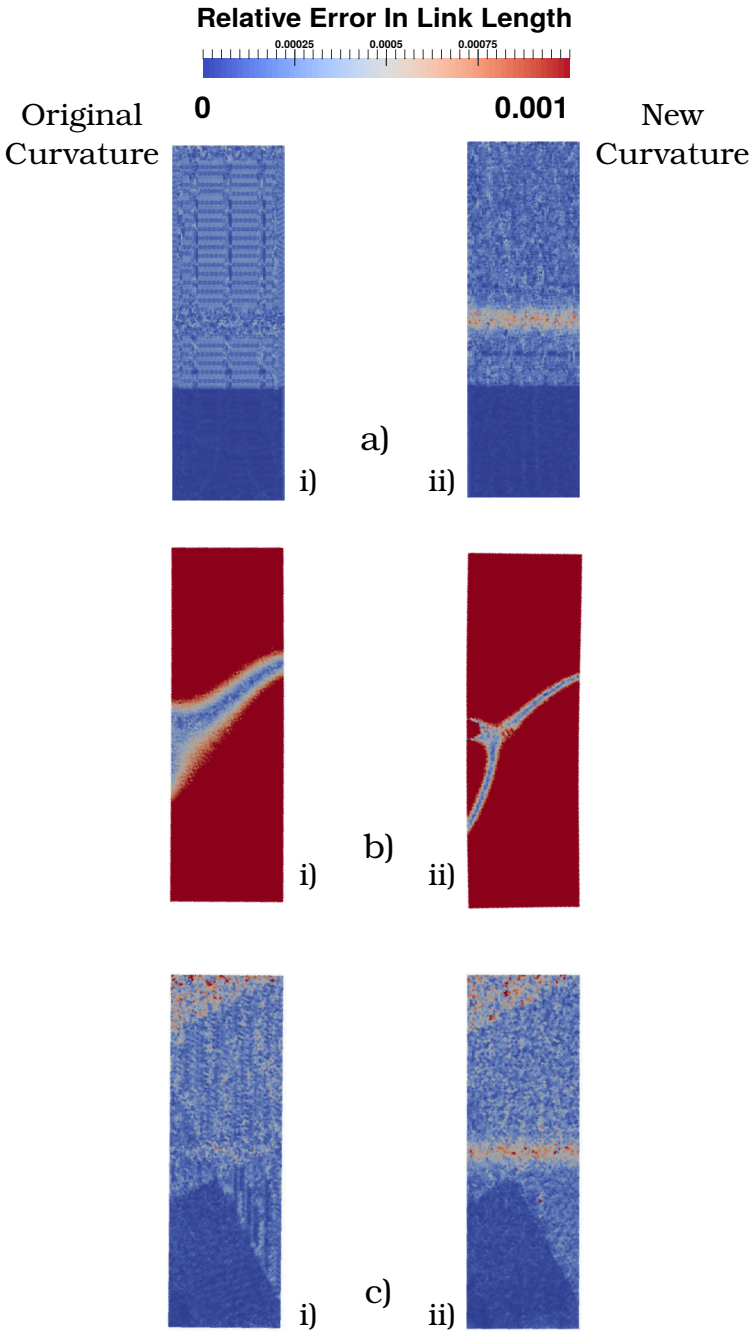

Figure 5.42: A comparison of the distribution in UTLM link length relative error between parameterisations of the flexible PCB with its original curvature (i), meshed using 14454 faces, and the new curvature of the PCB (ii) meshed using 12416 faces. The parameterisation methods applied are a) ABF, b) MIPS and c) LSCM. The relative error is displayed on a scale from 0 to 0.001

5.4 Timing Results

Now that an impression of the degree of distortion each parameterisation has been gained, this section takes a look at how long the parameterisation techniques require to arrive at their solution, for each of the test cases.

Beginning with the meshed surface of the hemisphere, Figure 5.43 shows a comparison of the length of time taken, in seconds, for each parameterisation method. It can be seen from this graph that the linear methods complete the parameterisation in a short amount of time, compared to the non-linear solutions provided by the ABF and MIPS methods. The coarse meshes are parameterised in only a few milliseconds, and this increases to 10 seconds for a mesh constituting 152912 faces for the fixed boundary methods. The free boundary method and linear method, LSCM requires 20 seconds to arrive at a solution. As expected, the non-linear methods, MIPS and ABF, each require a longer amount of time to parameterise the hemisphere. For ABF, this time is only 11 seconds for the coarsest mesh, which rises to 23 seconds for a mesh constituting 45648 faces, whilst 133 seconds is required to parameterise the hemisphere for the finest mesh of 152912 faces. The time required for the MIPS method to parameterise the geometry to the plane is similar to the ABF method, up to a limit of 45648 mesh faces, but it clearly requires significantly more time than the ABF method as the mesh becomes more refined. The parameterisation takes 226 seconds at the upper limit of mesh density studied using 152912 faces for the MIPS method.

The timings for mapping the metal flexible sheet to the plane, for a varying number of mesh densities, is shown in Figure 5.44. A similar trend is seen, whereby the linear methods are able to complete the parameterisation in significantly less time than the non-linear methods. Like the linear methods, the non-linear parameterisation provided by ABF again requires only a few milliseconds to parameterise the coarser

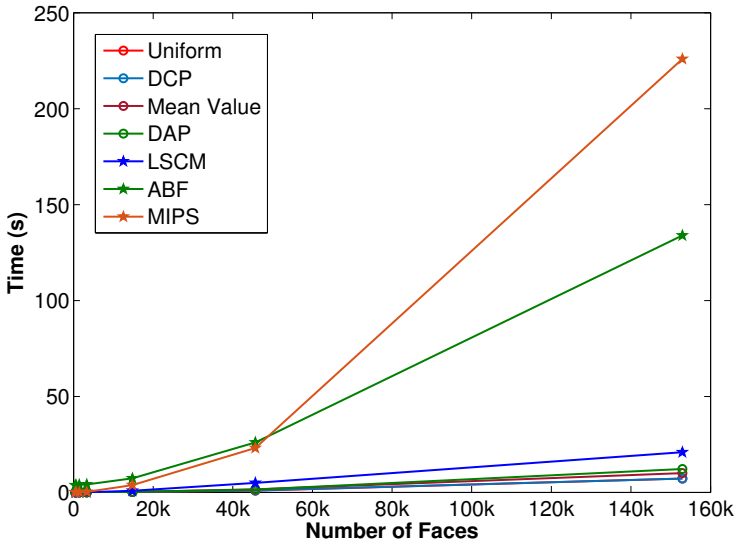

Figure 5.43: The amount of time, in seconds, for each parameterisation method to map the hemisphere to the plane, as mesh density is increased.

meshes, up to 6860 mesh faces. Increasing the mesh density to 55938 faces, ABF now requires 16 seconds, where the linear methods only require 2 seconds. This includes the free boundary method LSCM, whose timings coincided nicely with the fixed boundary parameterisations. MIPS is consistently the slowest method for this geometry, beginning at 3.5 seconds for coarse meshes, and requiring a runtime of 40 seconds for meshes refined to 55000 faces.

Deforming the flexible sheet into a new geometry and timing the parameterisations again gives rise to Figure 5.45. The parameterisation timings for this newly deformed sheet are consistent with those found with the original shape of the flexible sheet. As the mesh is refined, the non-linear methods require more and more time to complete the mapping compared to the linear methods. The MIPS method still remains the slowest, requiring 26 seconds to parameterise 50298 faces while ABF requires 15 seconds.

Turning attention to the flexible PCB test case, the time required to parameterise

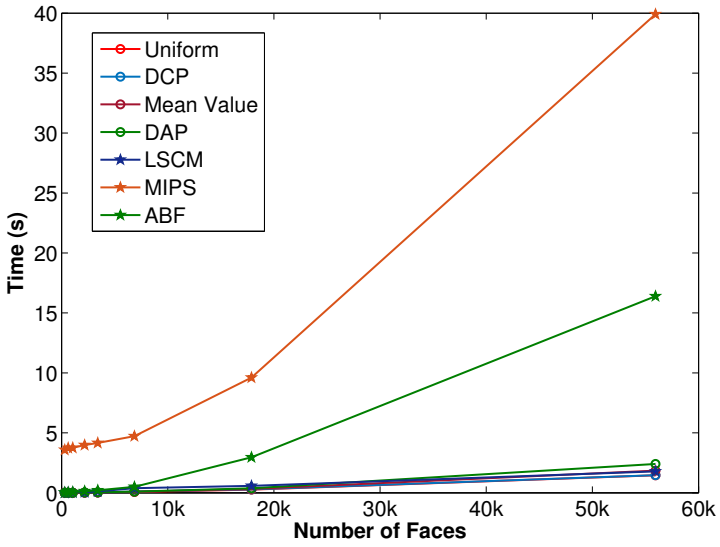

Figure 5.44: The time, in seconds, for each parameterisation method to map the flexible metallic sheet to the plane, as mesh density is increased.

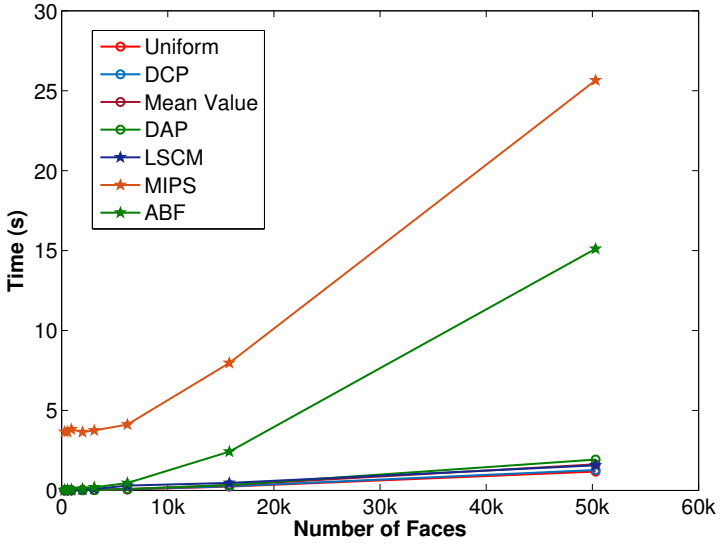

Figure 5.45: The time, in seconds, for each parameterisation method to map the newly flexed flexible metallic sheet to the plane, as mesh density is increased.

the surface from the 3D domain is no more than 3 seconds for the linear techniques, as shown in Figure 5.46. ABF shows a good runtime performance that coincides with the times for the linear methods for coarse representations of the geometry. As the mesh is refined further, ABF does require much more time to map the geometry, compared with all of the linear methods, rising to a runtime of 30 seconds for the mesh constituting 74384 faces. MIPS requires the longest runtime. Even for the coarse meshes, the parameterisation requires between 27 and 28 seconds to complete, and this runtime increases to 40 seconds for the highest mesh density studied.

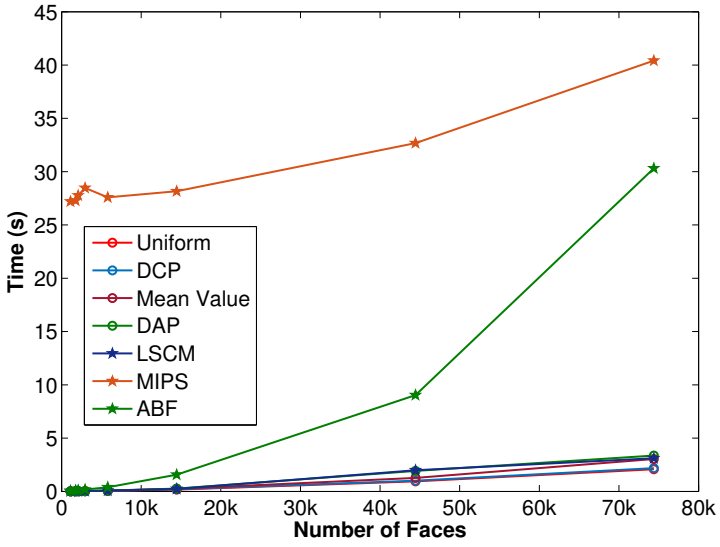

Figure 5.46: The time, in seconds, for each parameterisation method to map the flexible PCB to the plane, as mesh density is increased.

As Figure 5.47 shows, when comparing to Figure 5.46, changing the curvature of the flexible PCB has little impact of the amount of time required by the linear parameterisation methods. Differences are noticed when analysing the runtime of the non-linear methods. The MIPS method requires a substantially longer time to parameterise even the coarsest of meshes, which now takes 97 seconds map the mesh to the plane. This increases to 308 seconds when the finest mesh, constituting 74396 faces, is considered. The effect of the curvature of the geometry can also be seen in

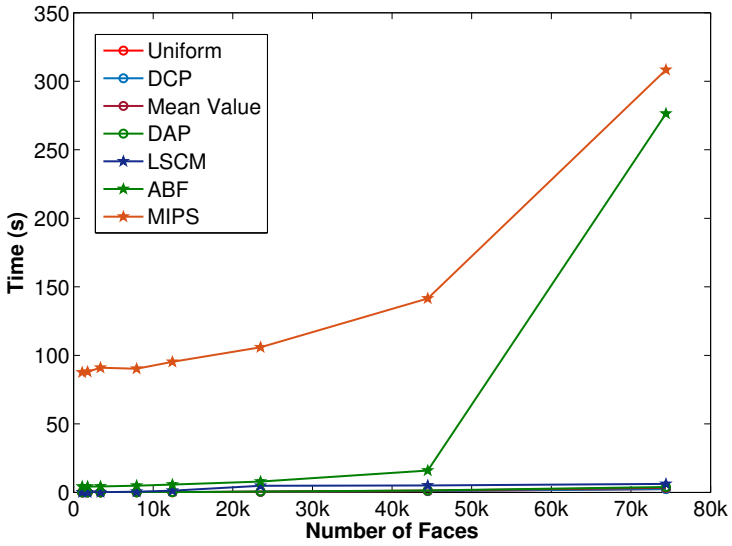

Figure 5.47: The time, in seconds, for each parameterisation method to map the flexible PCB to the plane with a new curvature, as mesh density is increased.

the timings for the ABF method, which somewhat coincide with the runtimes for the linear parameterisations of the coarse meshes. However, as the mesh is further refined, to above 40,000 faces, the necessary time for the mapping deviates and increases to 15 seconds, compared to the 5 seconds for the LSCM method. The runtime rises even further when the mesh consisting of 74396 faces is considered, which requires 276 seconds to complete the mapping.

The runtime results of these test cases demonstrate that it is not only the distortion that needs to be considered when choosing a parameterisation method, but also the time necessary for each to complete the mapping from the 3D domain to the 2D flat plane. The fixed boundary methods provide simple formulations that are very fast and efficient, however the level of distortion they induce when the boundary of the parameter domain does not closely match the boundary of the geometry in 3D space can be significant. Experimentation of the free boundary techniques leads to the conclusion that they are capable of minimising the distortion at the boundaries more effectively, however the non-linearity of the formulation not only increases the

complexity of the solutions, but, for very fine meshes, the time taken to parameterise the geometry becomes significantly greater. Curvature of the geometry also affects the runtime of the non-linear parameterisations, as demonstrated by the flexible PCB test cases. Linear methods are more immune to geometry curvature in terms of the time they require to map the surface to the plane.

5.5 Summary

This chapter provided an experimental comparison of the parameterisation techniques discussed in the previous chapter. The purpose of this investigation was to realise the effects of metric distortion each parameterisation technique imposes, in addition to the time each takes to converge to a mapping solution.

The investigation was restricted to open surfaces, topologically homeomorphic to a disk. Test cases were generated and meshed, providing geometries that conform to this criterion in the 3D domain. These test cases were a hollow hemispherical surface, a flexible metal sheet deformed into an arbitrary shape, and a flexible PCB, consisting of 3 metal wire tracks separated by a substrate.

Each test case was parameterised using each parameterisation method, and subsequently analysed in terms of the induced metric distortion. The parameterisation methods can be loosely categorised into linear and non-linear methods. Linear methods typically require the boundary of the 2D domain to be predefined and fixed prior to the parameterisation. If bijectivity is to be guaranteed, these boundaries are required to be regular, convex shapes such as circles and squares, which were the boundary definitions used in this experimentation. It was seen that the fixed boundary techniques exhibited significant amounts of distortion at the perimeter of the 2D domain for all of the test cases. The exception to this was the parameteri-

sation of the hemispherical surface to a circle in the 2D domain, a result attributed to the fact that the boundary of the hemisphere in the 3D domain is itself circular, which allowed the mesh to be injected into the predefined 2D domain with minimal authalic and angular distortion.

The free boundary techniques, which include the parameterisation of the boundary vertices as part of the solution, facilitated a parameterisation resulting in very little distortion at the edge of the 2D domain.

ABF performed consistently well as a parameterisation method, ensuring that the maximum magnitude of distortion inflicted on TLM link lines was generally significantly less than 1% for moderately fine meshes, with a median distortion of less than a tenth of a percent. LSCM also performed comparably well to the ABF method, and offers a much simpler and linear solution.

This chapter concludes that the ABF method, although a non-linear solution, produces a parameterisation with the least amount of distortion, and consequently the smallest deviation from the original lengths of the link lines constituting the UTLM network. This was consistent across all of the test cases, which provided geometries of different curvatures and dimensions. For meshes which have a mesh density of more than 40000 faces, the LSCM method provides a good alternative. LSCM induces slightly more distortion than ABF, however, because of the linearity of the solution, a parameterisation is achieved in a significantly shorter amount of time for meshes with a very high face density.

The next chapter presents the results of UTLM simulations on the test geometries used in this chapter, to further the investigation of the suitability of mesh parameterisation for the purposes analysing electromagnetic behaviour on these surfaces.

References

[5.1] J. R. Shewchuk, "Reprint of: Delaunay Refinement Algorithms for Triangular Mesh Generation," *Comput. Geom. Theory Appl.*, vol. 22, pp. 21–74, 2014.

[5.2] K. Hormann and M. Floater, "Surface Parameterisation: A Tutorial and Survey," *Adv. Multiresolution Geom. Model.*, pp. 405–417, 2005.

[5.3] W. T. Tutte, "Convex Representations of Graphs," *Proc. London Math. Soc.*, vol. 10, no. February 1959, 1960.

[5.4] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '95*, vol. 6, pp. 173–182, 1995.

[5.5] M. S. Floater, "Wachspress and Mean Value Coordinates," *Approx. Theory XIV*, no. 1, pp. 81–102, 2014.

[5.6] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parametrization of Surface Meshes," *Eurographics*, vol. 21, no. 2, 2002.

[5.7] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Trans. Graph.*, vol. 21, no. 3, 2002.

[5.8] K. Hormann and G. Greiner, "MIPS : An Efficient Global Parametrization Method," *Curve Surf. Des.*, pp. 153–162, 2000.

[5.9] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov, "ABF++: Fast and Robust Angle Based Flattening," *ACM Trans. Graph.*, vol. 24, no. 2, pp. 311–330, 2005.

[5.10] K. Hormann, K. Polthier, and A. Sheffer, "Mesh Parameterization," *ACM SIGGRAPH ASIA 2008 courses - SIGGRAPH Asia '08*, vol. 2, no. December, pp. 1–87, 2008.

Chapter 6

Applying Mesh Parameterisation to UTLM

The previous chapter presented a comparison of the mesh parameterisation methods studied as part of the investigation encompassing this thesis. It was found that the Angle Based Flattening method (ABF) [6.1] for parameterising triangular meshes in the 3D domain to a 2D domain induced the least amount of distortion to area and angles. This method has been carried forward to this chapter, where the application of a 2D UTLM simulation is presented for thin flexible surfaces in the 3D domain.

The geometries under investigation were also introduced in Chapter 5 in Figure 5.1, and brought forward to this chapter. The problem spaces are a hemispherical surface, offering a simple uniform geometry as an initial test, and a flexible metallic sheet, flexed into an arbitrary shape. A more complicated case, in terms of the material parameters, is then used in the form of a flexible PCB. This provides a non-uniform material, consisting of three wire tracks, separated by substrate.

Advances in small scale fabrication processes have led to the advent of very small scale devices, such as flexible RFID tags, and smart clothing; it is not uncommon for

these devices to have a thickness of 1mm or less [6.2]. These devices present themselves as 2D surfaces embedded in a 3D domain. When simulating electromagnetic behaviour on these surfaces, at low frequencies, using 3D algorithms, large portions of the memory and runtime can be used to simulate areas of the domain that present little activity. Chapter 3 demonstrated how the mesh density and connectivity can impact the memory and runtime of a 2D Unstructured Transmission Line Modelling (UTLM) simulation.

This impact is negatively amplified in 3D space; full meshing of the domain is required using tetrahedra rather than triangles [6.3]. Figure 6.1 shows an example of a hemispherical surface, 1m in diameter in the 3D domain, coarsely meshed using 420 triangles in part a) is bounded by a box and meshed using tetrahedra in part b). This results in a significant increase in the number of mesh elements, and a greater amount of memory consumption to represent the geometry. As the surface mesh is further refined, the negative impact on memory consumption becomes greater.

Alternatively, 3D UTLM algorithms can be used by modelling the small thickness of the geometry, shown in Figure 6.2 a), where a flexible PCB is modelled with a 2mm substrate and 1mm thick wire tracks. Meshing this geometry using tetrahedra necessitates a fine mesh, such that the fine features of the geometry can be represented. The mesh in Figure 6.2 b) is meshed using 13847 tetrahedra, where the mesh is one cell thick. This is the minimum number of mesh elements necessary to maintain a good quality tetrahedral mesh for the purposes of a 3D UTLM simulation. Extracting the top surface of this mesh, yields a triangulation that constitutes 5796 faces, shown in Figure 6.2 c). A method allowing a 2D UTLM simulation to be performed on this surface, rather than in 3D facilitates a much more computationally efficient simulation; less memory is needed to represent the geometry, and the 2D UTLM algorithms are less computationally intensive resulting in a faster runtime.

A further limitation to meshing the thickness of a very thin geometry is the min-

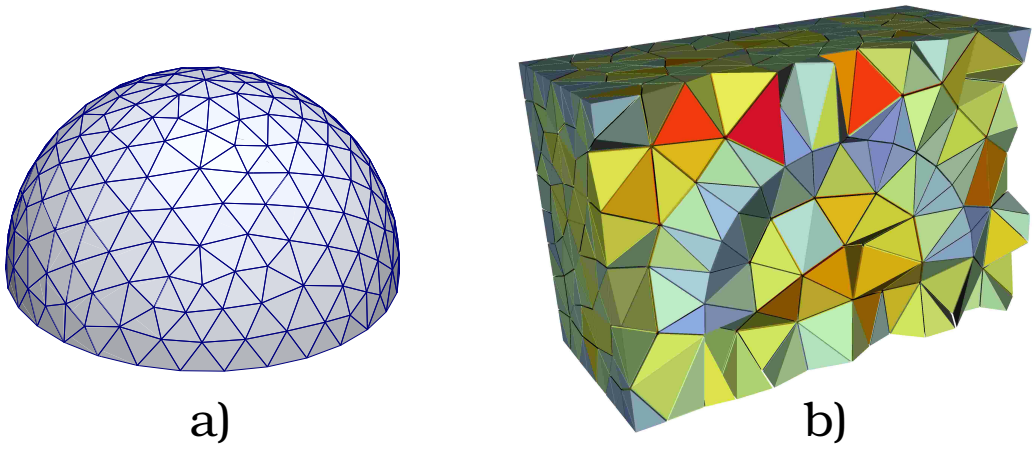

Figure 6.1: a) Hemispherical surface meshed using 420 triangles, in b) the surface mesh is bounded by a box and the entire 3D domain is meshed using tetrahedra

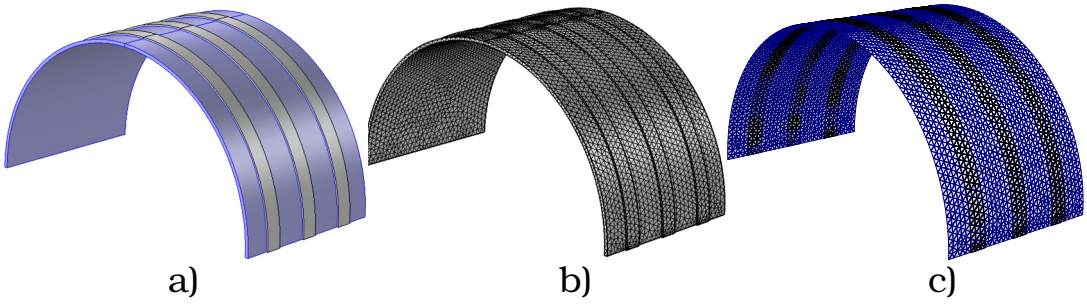

Figure 6.2: a) PCB modelled with thickness of 2mm, with 1mm thick wire tracks in 3D. b) PCB meshed in 3D domain using 13847 tetrahedra. c) surface of the PCB meshed using 5796 triangular faces

imum thickness that can be modelled such that a good quality tetrahedral mesh, conforming to a Delaunay mesh criterion [6.4] can be created. For example, popular, commercially available numerical simulation packages, such as COMSOL (as of version 4.4) [6.5] limit the maximum thickness of 3D geometry to $5.04 \times 10^{-4} m$. This limits the thickness of the geometry to only half of the thickness of the wire track modelled in Figure 6.2 a). Simulating surface behaviour on thinner geometries would require meshing the surrounding 3D domain, as was depicted in Figure 6.1 b).

By parameterising surfaces such as Figure 6.1 a), a 2D UTLM simulation can be car-

ried out, negating the need for the more computationally intensive 3D algorithms. ABF is used to parameterise the meshed geometries under test, and a 2D Unstructured Transmission Line Modelling (UTLM) simulation is then conducted on these surfaces. The results are subsequently mapped back to the original 3D geometry for visualisation. The following section begins with a UTLM simulation on the hemispherical surface.

6.1 UTLM on a Hemispherical Surface

The hemispherical surface, 1m in radius, is meshed using 1424, 3300 and 14792 faces. The material properties of the surface are defined as having a relative permittivity, $\varepsilon_r = 2$, with a resistivity of 50Ω/m, modelling a lossy medium. Each mesh is then parameterised to the 2D domain using the ABF method [6.1]. The source points are defined on the 3D surface, which are also mapped, along with the material properties, through the parameterisation process. It is these material properties that are subsequently used to derive the individual UTLM link line parameters in the 2D plane.

The excitation is defined to be a Gaussian step function, such that:

$$V(t) = \begin{cases} V_{pk} \times e^{-\frac{(t-t_0)^2}{w_0^2}} & \text{if } t < t_0, \\ V_{pk} & \text{if } t \geq w_0 \end{cases}, \tag{6.1}$$

where $V(t)$ is the source voltage at time t, V_{pk} is the peak voltage of the source, w_0 defines the width of the Gaussian pulse in seconds, and t_0 is the rise time of the pulse in seconds. In this case, the Gaussian pulse will rise to the peak voltage taking the amount of time t_0 is defined to be. For this simulation, the source peak voltage, V_{pk}, was set at 10V and w_0 was defined to be the total simulation time of 3×10^{-7}

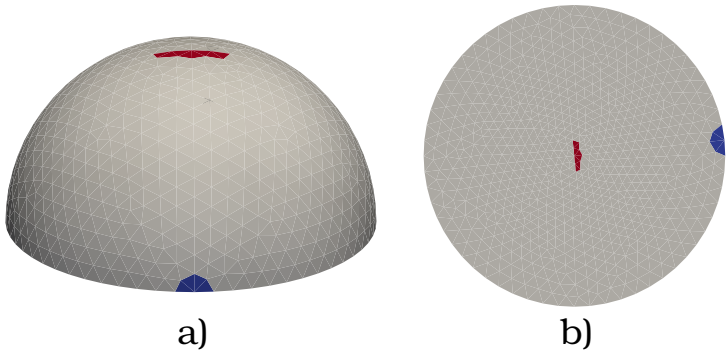

Figure 6.3: Source areas for the UTLM simulation on the hemispherical surface. The source areas are defined in 3D space, shown in a) where the red area corresponds to a positive voltage source and the blue a negative voltage source. Part b) is the parameterisation of the original mesh, in the 2D domain, showing the position of the parameterised source locations.

seconds, with t_0 set to 1.5×10^{-7} seconds. This resulted in a simulation time step of 4.7×10^{-11}, 3.6×10^{-11} and 1.56×10^{-11} seconds, for the meshes constituting 1424, 3300 and 14792 faces.

The position of the excitation source is defined in the 3D domain, and shown in Figure 6.3 a), with the red area corresponding to a positive voltage excitation (with a peak nodal voltage of +10V), and the blue area defining a negative voltage excitation (peaking at -10V), providing a voltage potential between these two areas. Figure 6.3 b) is the parametrisation of the original surface, showing the two parameterised source areas in the 2D domain. Finally the boundary of the geometry is defined to be a matched boundary, where the circumference of the hemisphere is defined to have matching resistivity to the surface. From these initial conditions, the simulation was run in the 2D domain, and the results were mapped back to the original surface for visualisation.

Figure 6.4 shows the nodal voltage distribution across the parameterised mesh at the last time step of the simulation, which coincides with a steady state for the different mesh sizes. The voltage potential between the source points can be seen for each

mesh case. Figure 6.4 a) is the mesh consisting of 1424 faces, b) consists of 3300 faces and c) is the mesh constituting 14792 faces. For each case, i) presents the simulation results at steady state in the 2D domain. The results of the parameterisation were then mapped back to the original surfaces in the 3D domain, shown in part ii) for each case. The voltage distribution converges quickly as the mesh in refined. The voltage drop created on the surface is where the distribution of current is expected to reside and this is confirmed in Figure 6.5, which shows the magnitude of current on the surface, where the meshes coincide with those in Figure 6.4. The peak magnitude of current quickly converges to $2 \times 10^{-2} A$ and as the mesh is refined, the visualisation of the current distribution also becomes more refined. It is noticed that the voltage distribution in Figure 6.4 appears to be smooth where the distribution in current shown in Figure 6.5 does not show as smooth transition between the source points as the mesh is refined. This is attributed to the voltage is being continuous from circumcentre to circumcentre of the triangulation, which are the UTLM node centres, and the point the field is sampled for each triangle constituting the triangulation. As discussed in Chapter 2, the current is continuous across the edges of each triangle. Despite the fact the current sampled across each edge of the triangulation is continuous, and the distribution is as expected, the total magnitude of the current sampled at the centre node (the circumcentre) is an ensemble of the current at the three edges, and will not necessarily appear as continuous when visualised.

To provide a comparison of the current distribution resulting from the 2D UTLM simulations on parameterised surfaces, a simulation using a validated 3D structured TLM solver (GGI TLM [6.6]) was performed. The geometry is given a thickness and mesh size of 0.025m, such that the geometry is now meshed using cubes rather than triangles. The mesh constitutes 3833 elements; a similar density to the hemisphere mesh is Figure 6.5 b)i). The material and simulation properties are left unchanged. However, in order to excite the geometry, a cable connecting the source points

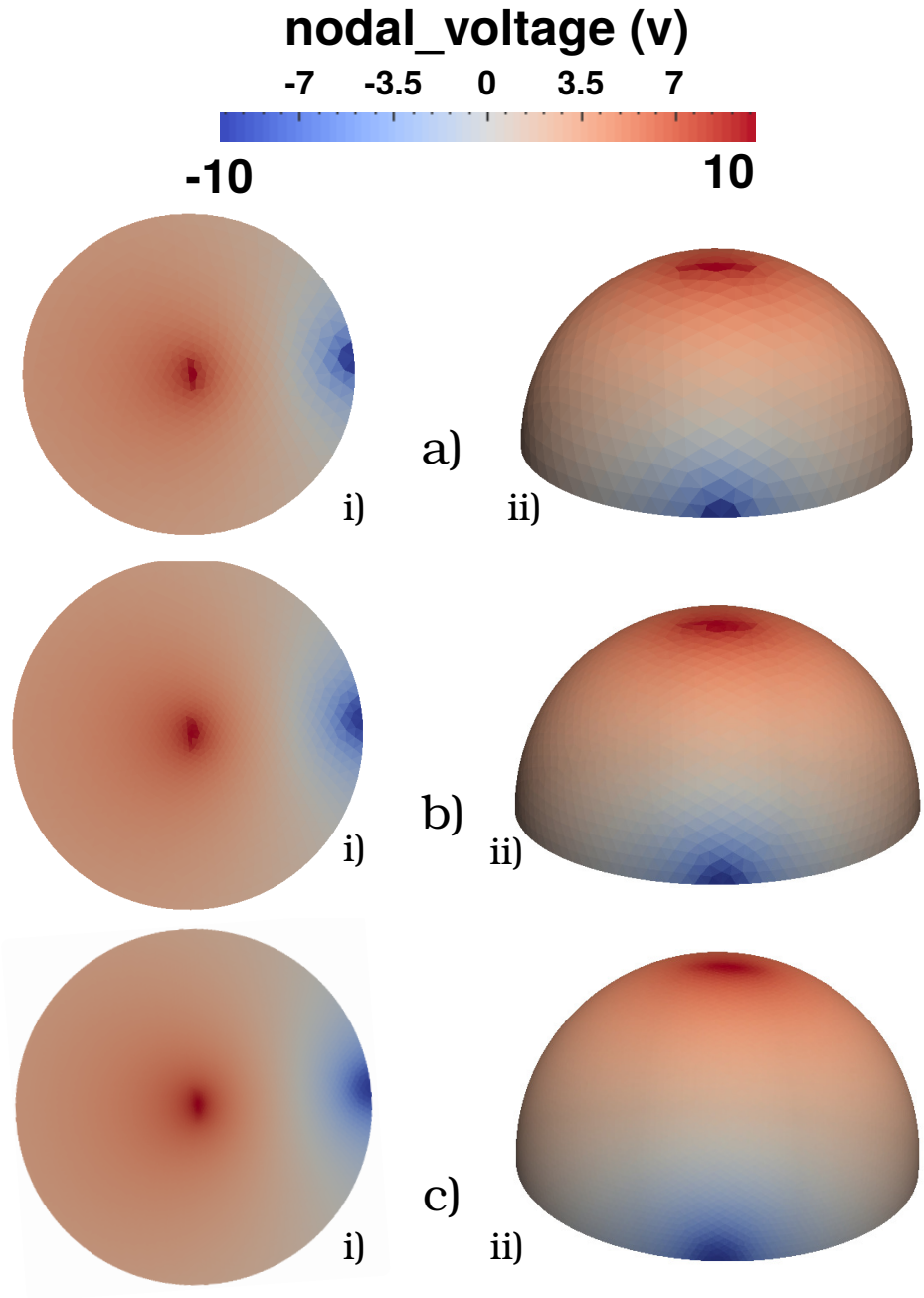

Figure 6.4: Visualisation of voltage potential across the surface of the hemisphere. Part a) is the mesh with 1424 faces, b) consists of 3300 faces and c) constitutes 14792 faces. The UTLM is conducted in the 2D domain (i) before being mapped back to the 3D domain (ii).

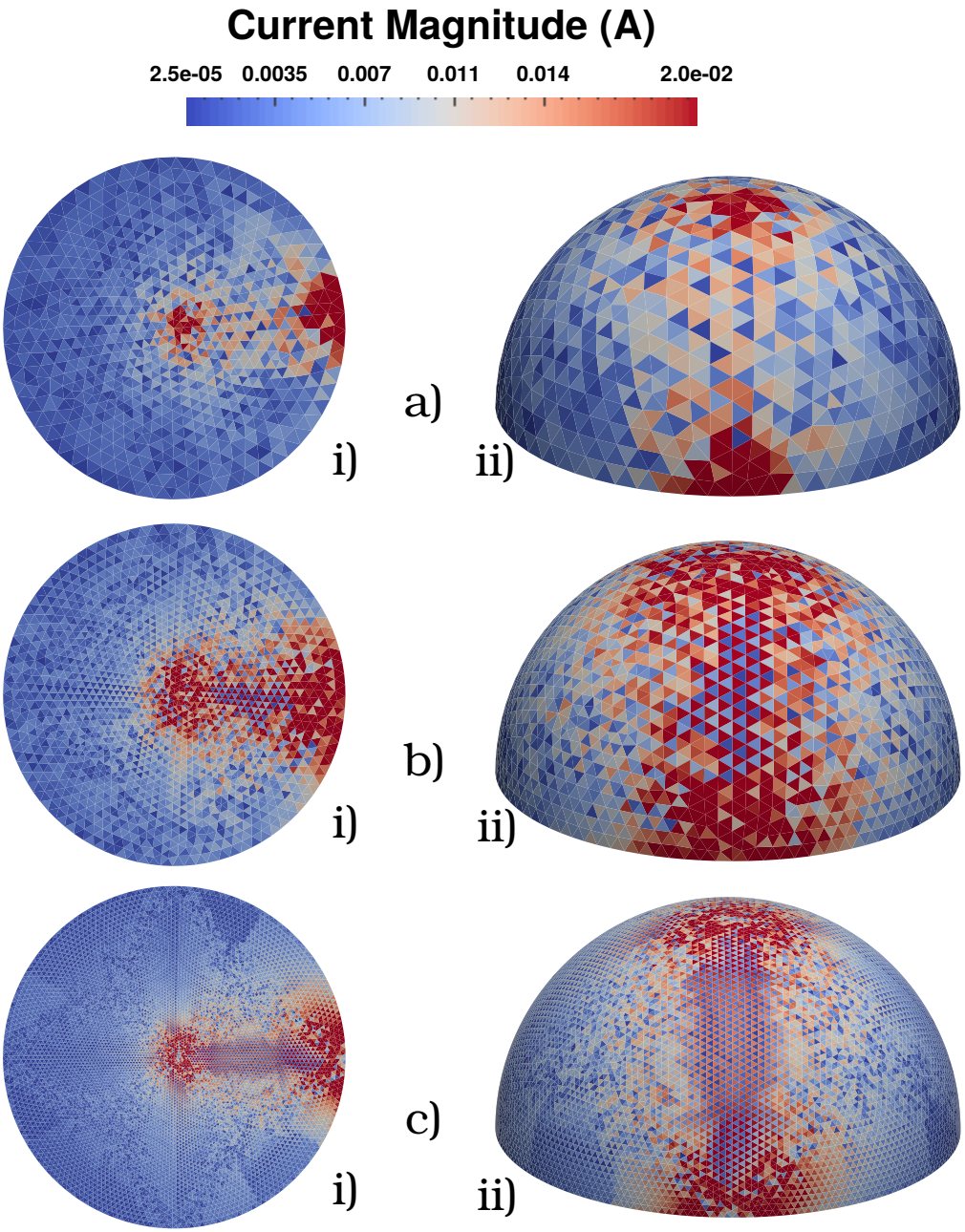

Figure 6.5: Visualisation of the steady state current distribution resulting for the UTLM simulation on the hemispherical surface. Part a) is the mesh with 1424 faces, b) consists of 3300 faces and c) constitutes 14792 faces. The UTLM is conducted in the 2D domain (i) before being mapped back to the 3D domain (ii).

becomes necessary to provide a closed system, fixing the voltage potential between these two points. This was not necessary in the 2D domain, offering the advantage of modelling sources numerically without the necessity of modelling cables, as required in the 3D case.

The simulation is run for the same simulated time of 3×10^{-7} seconds, and the visualisation of the current distribution is shown for the final time step in Figure 6.6. Part a) shows the results from the front, where the current distribution is seen to lay between the two source points creating the voltage potential, and part b) is the visualisation from the back. The behaviour of the current coincides with that seen in Figure 6.5, with a very similar peak current magnitude. The results demonstrate that the behaviour of the current distribution resulting from the 2D UTLM simulations in Figure 6.5 is as expected, with a peak magnitude of current similar to that resulting from the 3D simulation. However, differences can be seen due to the stair-casing effect as a result of the structured mesh Figure 6.6, and its failure to conform to the curvature of the geometry. A mesh density far greater than the triangulations used for the 2D UTLM simulations would be necessary to conform to the curvature.

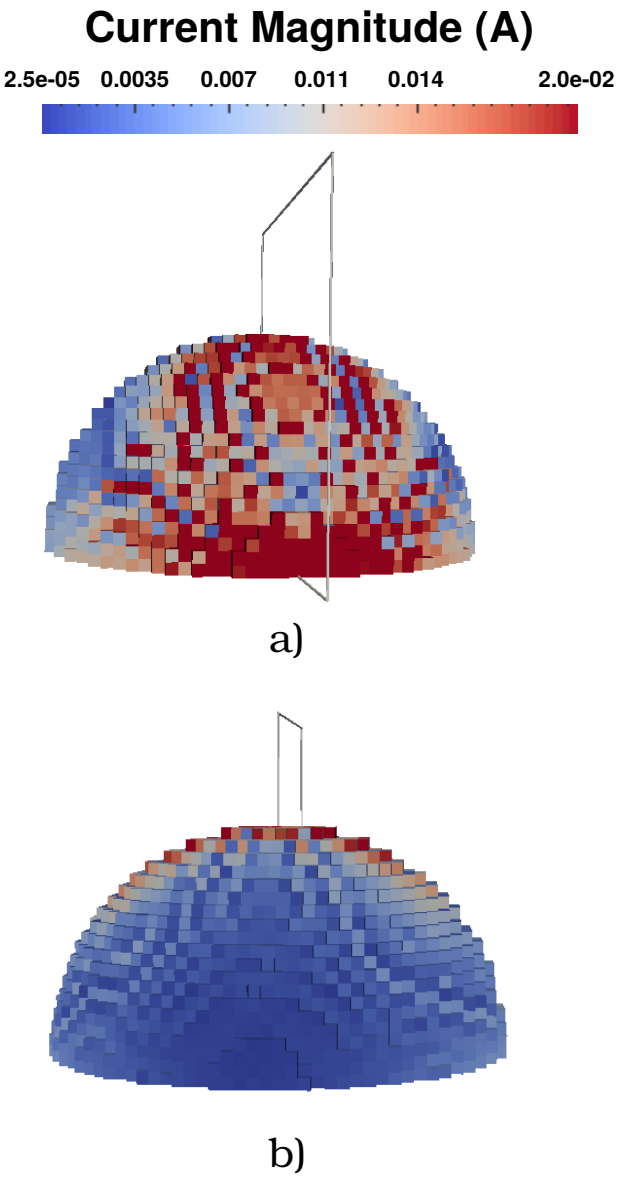

Figure 6.6: Recreating the simulation of the hemisphere using a 3D structured simulation performed using GGI TLM [6.6]. The geometry is meshed using a cell size of 0.025m. a) shows the distribution of current from the front of the geometry, and b) is the visualisation from the back.

6.2 UTLM on a Flexible Metallic Sheet

The next test case is a flexible metallic sheet. The dimensions of this geometry are 0.1m in width and 0.2249m in length. The geometry is meshed using 658, 1050, 2170 and 6860 faces. Figure 6.7 shows the meshed geometry using these varying degrees of refinement. Parts a) and b) are the meshes consisting 658 and 1050 faces, which barely conform to the curvature of the original geometry. Figure 6.7 c) and d) are the meshes constituting 2170 and 6860 respectively, which descritise the curvature more accurately.

Also depicted in Figure 6.7 are the source points used to excite the mesh prior to the UTLM simulation. The red area highlighted on each mesh is the location of the positive voltage source and the blue area negates the magnitude of the positive source with the application of a negative voltage.

The material parameters of the metallic flexible sheet are those that coincide with aluminium; the relative permittivity of the material, ε_r, was defined to be 1, and the surface was given a conductivity of $3.50 \times 10^7 S/m$, which translates to a resistivity of $2.85 \times 10^{-8} \Omega/m$ [6.7]. The boundary conditions were defined to be a short circuit around the entire parameter of the geometry. Once the material parameters were defined in 3D space, the surface was parameterised to the 2D domain, using the ABF method [6.1]. The resultant meshes are presented in Figure 6.8, where part a) is the mesh constituting 658 faces, b) is the mesh consisting of 1050 faces, c) is the parameterisation of 2170 faces, and d) has 6860 faces.

The 2D surface was excited with a Gaussian step voltage defined in equation (6.1), where t_0 is defined to be 1×10^{-8} seconds and w_0 is 2×10^{-8} seconds. The voltage peak was 10V, where $+10V$ was applied to the area shown in red in Figure 6.8 and $-10V$ was applied to the area in blue. The UTLM simulation was then run for a simulated time of 2×10^{-8} seconds, using a time step of 1.81ps, 2.29ps, 2.88ps and

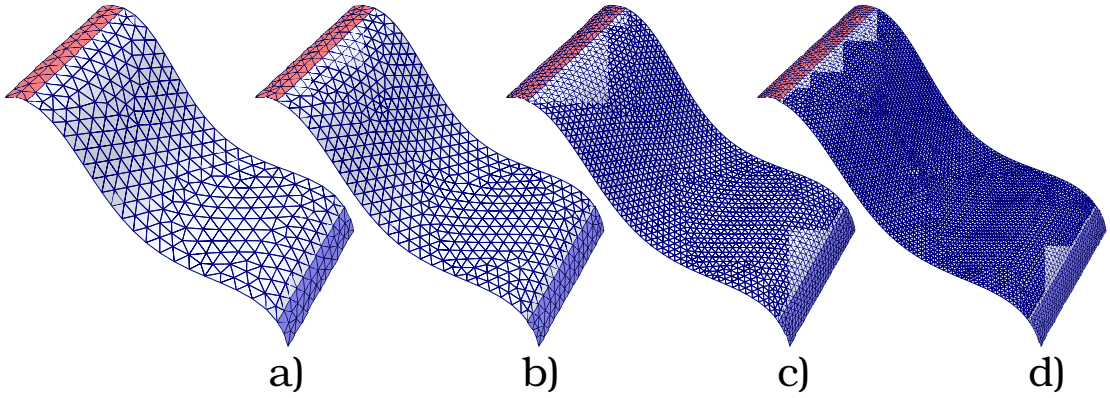

Figure 6.7: The flexible sheet meshed in the 3D domain using a) 658 faces, b) 1050 faces, c) 2170 faces and d) 6860 faces.

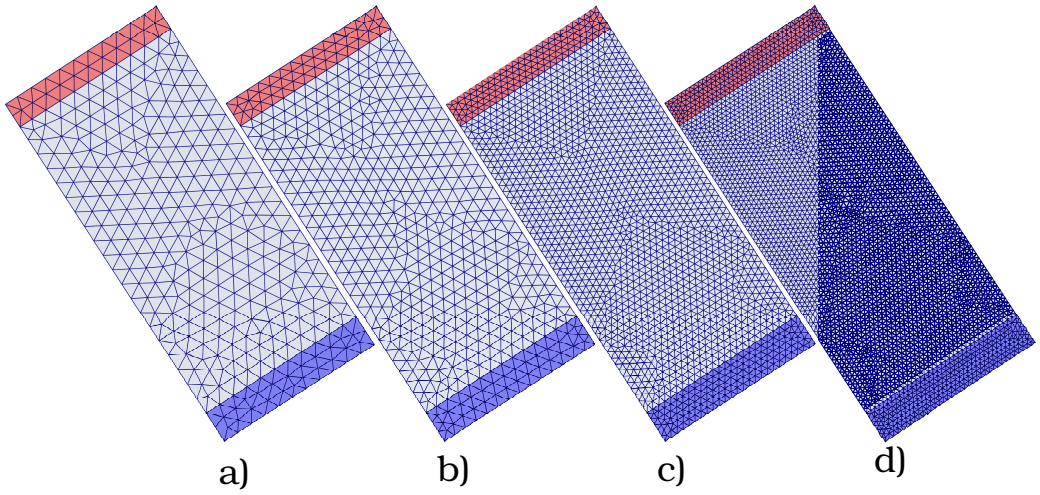

Figure 6.8: The flexible sheet parameterised to the the 2D domain using the ABF method, where a) has 658 faces, b) has 1050 faces, c) has 2170 faces and d) has 6860 faces.

1.44ps for the meshes constituting 658, 1050, 2170 and 6860 faces respectively.

The voltage distribution across the surface is shown in Figure 6.9 where part i) for each case are the UTLM results at the end of the simulated time, on the parameterised surfaces, and part ii) presents the results mapped back to the original surface in the 3D domain. Figure 6.9 parts a) to d) are the results using meshes with a density of 658, 1050, 2170 and 6860 faces respectively.

Analysing the results, it can be seen that for the coarsest mesh, artificial behaviour is witnessed at the bottom left corner of Figure 6.9 a) i). This could be due to a combination of using a very low resolution mesh, and the reflection from the short circuit boundary. Using a source applied symmetrically, on a symmetrical geometry, the propagation is also expected to be symmetrical. This is a very coarse mesh which barely conforms to the curvature of the geometry, and it is expected that this artefact should disappear as the mesh is refined. This is the case in the subsequent visualisations of the voltage in parts c) and d). As the mesh is refined, beyond 1050 faces, the peak magnitude of voltage quickly converges towards $4.97 \times 10^{-2} V/m$, which can be seen between the simulations on the meshes consisting of 2170 and 6860 faces, seen in Figure 6.9 c) and d). The detail in the distribution of the nodal voltage also becomes more refined as the mesh is refined. The short circuit perimeter of the geometry means that the applied voltages on the surface will be totally reflected, yielding a voltage distribution that is not a smooth transition between the positive and negative source areas, as was seen for the matched boundary condition applied to the hemispherical surface in Figure 6.4.

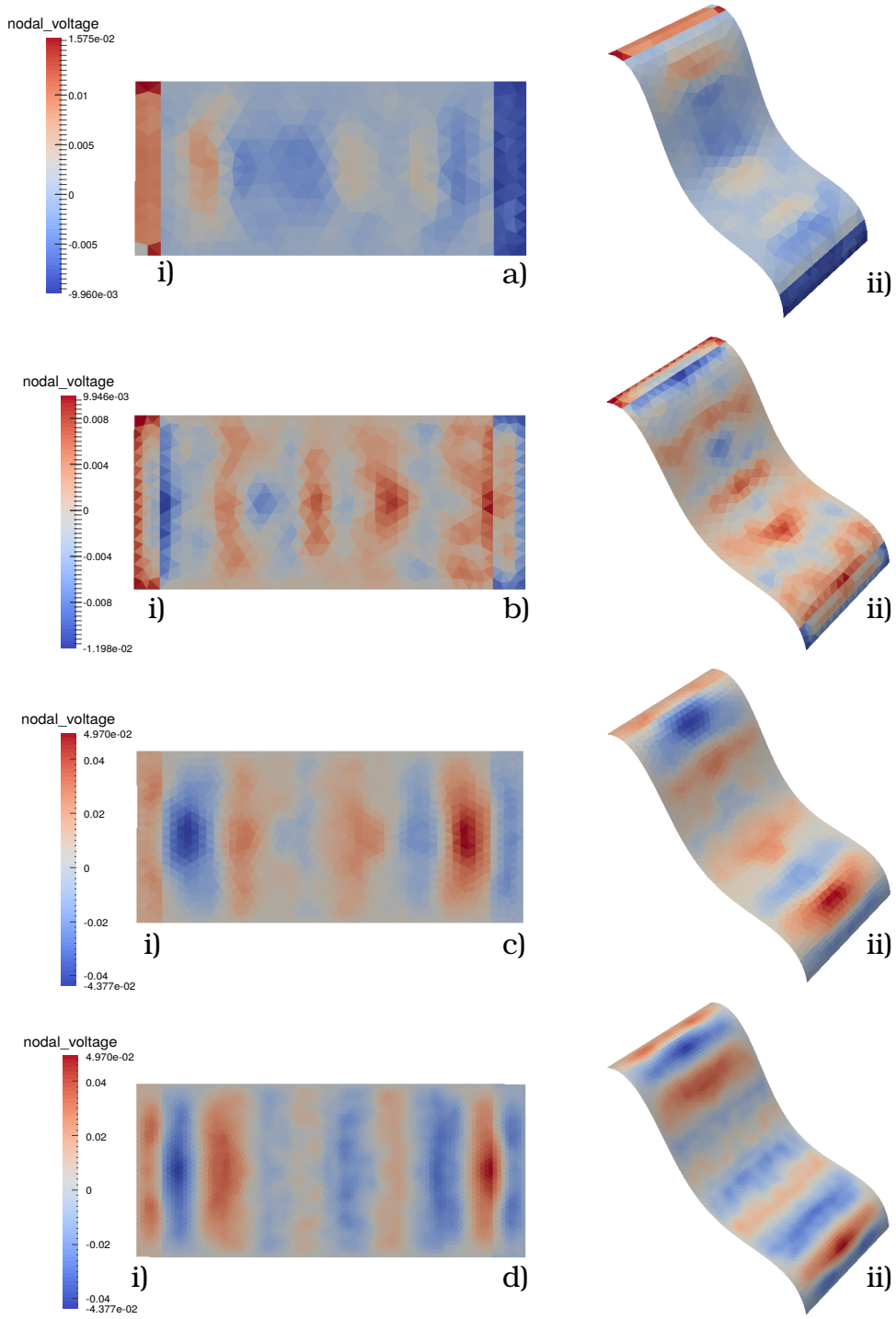

Figure 6.9: UTLM simulation results for the flexible metallic sheet at the final time step, using meshes of density a) 658, b) 1050, c) 2170 and d) 6860 faces. Part i) in each case are are the results in the 2D domain, before being mapped back to the original 3D geometry (ii)

6.3 UTLM on a Flexible PCB

The next test case is a flexible PCB. This provides a test case with non-uniform material. This geometry is constructed as a curved surface in the 3D domain, using three metal wires, separated by substrate. The geometry has a flat length of 0.3m and width 0.1m. Three metal tracks are modelled; each wire is 0.01m in width and equally spaced apart. The geometry is subsequently meshed, and the resultant meshes used for this test case are shown in Figure 6.10. Figure 6.10 a) is the geometry meshed using 1124 faces, part b) consists of 2948 faces and c) constitutes 5796 faces. Two of the wires have been short circuited using a metal interconnect, which can also be seen for each mesh in Figure 6.10, with the remaining wire left unconnected. The purpose of leaving one wire free from the connected two wires is to determine if the UTLM simulation results in electrical coupling between the wires. The material parameters of the wires are defined to be that of copper, where the conductivity is $1.68 \times 10^{-8} S/m$ [6.8], with a relative permittivity, ε_r, of 1. The substrate is defined to coincide with a class of material known as FR-4 [6.2], a versatile thermoset plastic laminate with electrical properties that lend utility for a wide variety of electrical applications, including PCB substrate. The substrate parameters used for this test case have a relative permittivity, ε_r, equal to 4.8, with a resistivity of $1 \times 10^8 \Omega/m$. The boundary conditions at the perimeter of the geometry is defined to be short circuit.

The points of excitation are also shown in Figure 6.10, for each of the meshes. The red and blue selection of triangles define the location of an E_z excitation, V, where the polarity of the excitation is shown in Figure 6.10 a). This provides a fixed voltage potential between these two points.

Prior to the initiation of the UTLM simulation, the meshes are parameterised using the ABF method [6.1]. The resultant meshes are presented in Figure 6.11, where

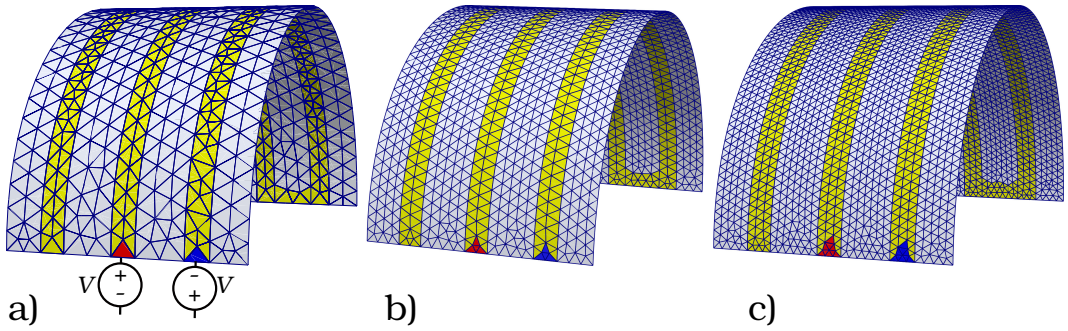

Figure 6.10: Flexible PCB meshed using a) 1124 faces, b) 2948 faces and c) 5796 faces. Source points are indicated in red and blue in each case.

part a) is the parameterisation of the mesh with 1124 faces, b) is the mesh with 2948 faces and part c) is the mesh consisting of 5796 faces.

The UTLM network is constructed on the parameterised meshes. The excitation of E_z at each source node was then applied, defined to be a sinusoidal voltage with a voltage peak of $10V/m$ and a frequency of 100MHz, such that a peak electric field of $10V/m$ is applied to the red source point, and a negation of the electric field $(-10V/m)$ is applied to the nodes of the blue source point. The sinusoidal source persists for the duration of the simulation, which was set to be a simulated time of $2 \times 10^{-8}s$. The time steps for each simulation were 42ps, 29ps, and 19ps for the meshes constituting 1124, 2948 and 5796 faces respectively.

Figure 6.12 shows the magnitude of the magnetic field across each mesh at the final time step of the simulation. As the mesh becomes more refined, the peak magnitude of the magnetic field is seen to converge to a value of $3.43 \times 10^{-6}A/m$. The PCB track left unconnected from the source loop is also seen to exhibit signs of cross talk, where a small tangential magnetic field with a peak magnitude of $5 \times 10^{-7}A/m$ is induced at the last time step of the simulation. Refinement in the distribution of the magnetic field is also seen as the mesh density increases, with the field largely confined to the wire tracks as expected, due to the high resistivity and relative permittivity of the substrate material. A symmetrical distribution of the magnetic

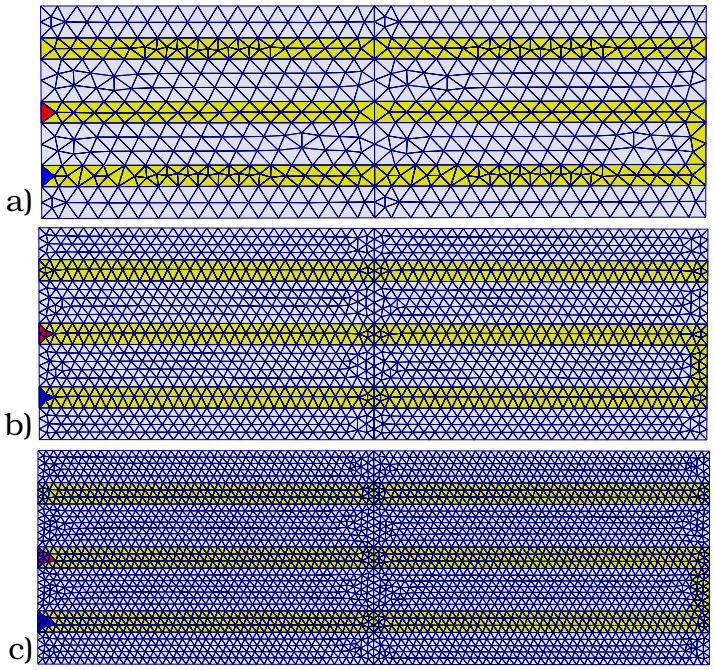

Figure 6.11: Parameterisations of the flexible PCB meshed using a) 1124 faces, b) 2948 faces and c) 5796 faces. Source points are indicated in red and blue in each case.

field is seen, which should be expected, as the source used to excite the geometry is symmetrical, coupled with the symmetry of the geometry.

Running UTLM on the parameterised mesh facilitates the modelling of the cross talk between wires in the plane. However, it should be pointed out that because the simulation is performed in two dimensions that any coupling between the two ends of the PCB in the 3D domain is not modelled. Hence, the method of using surface parameterisation is restricted to low frequency applications, where the wavelength of the incident source is greater than the dimensional order of the problem space. For higher frequency applications, where greater inductive coupling is expected, a full field profile would be required. Hence, if deforming the PCB further, while maintaining the original dimensions, material parameters and excitation, the UTLM simulation results should not differ using parameterisation and 2D UTLM algorithms, as the coupling in 3D dimensions is not modelled.

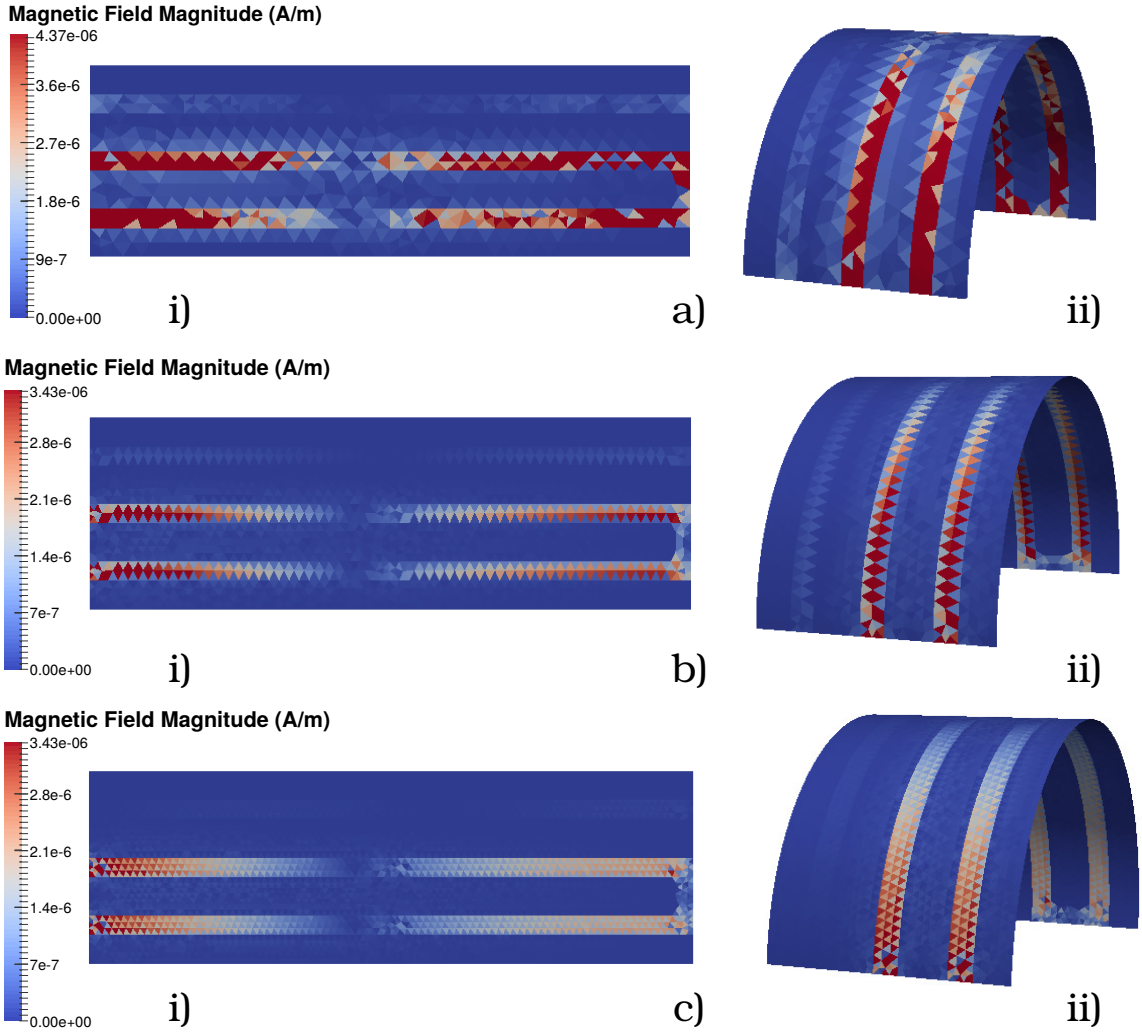

Figure 6.12: Resulting magnetic field magnitude from a UTLM simulation of the flexible PCB after 2×10^{-8} seconds and the source is a sinusoidal 10V/m E_z excitation. The simulation is performed on the parameterised meshes i) constituting a) 1124, b) 2948 and c) 5796 faces before being mapped back to the 3D domain in ii). The wire tracks are copper, with FR-4 substrate.

However, if the original surface in the 3D domain is altered in such a way that changes the electrical parameters of a material, perhaps through mechanical stress, or thermal fluctuations, where the resistivity of a material changes with a change in temperature, the application of mesh parameterisation for the purposes of a UTLM simulation can be useful. This is demonstrated in the following subsection.

6.3.1 Deforming the Flexible PCB

The flexible PCB is further flexed resulting in an increase of curvature at the apex of the geometry. The geometry is then re-meshed, giving rise to Figure 6.13, where part a) is the geometry meshed using 1084 faces, b) 3392 faces and c) 7916 faces.

The same material parameters are used for this geometry as the original flexible PCB, which are, again, applied in the 3D domain before parameterising the geometry to 2D space. Figure 6.14 shows the new parameterisation of the PCB where the ABF method has been applied. The meshes a), b) and c) correspond with the meshes

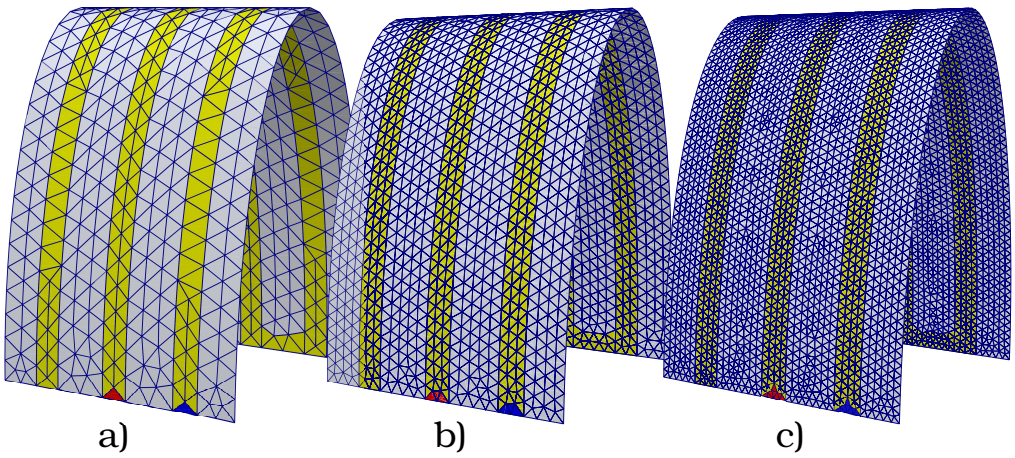

Figure 6.13: Flexible PCB deformed into a new curvature, meshed using a) 1084 faces, b) 3392 faces and c) 7916 faces. The Source points for the excitation are indicated by the selection of red and blue triangles in each case.

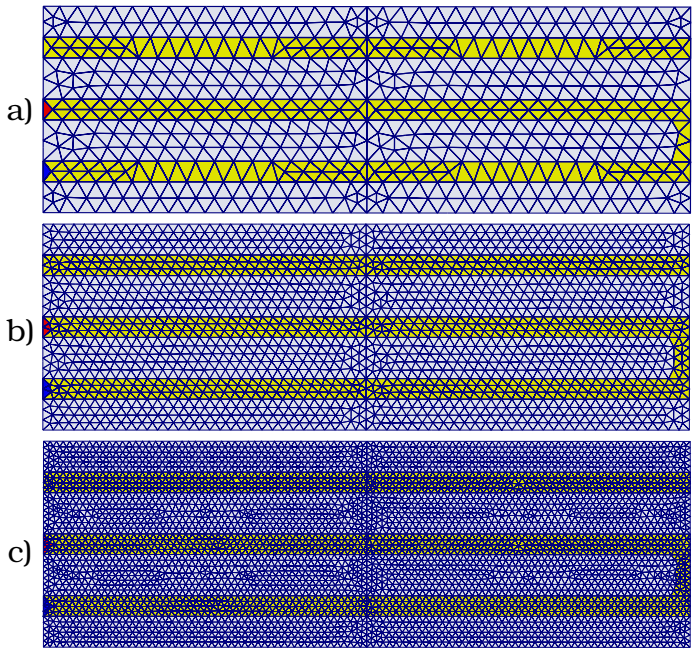

Figure 6.14: Parameterisation of the deformed flexible PCB meshed using a) 1084 b) 3392 and c) 7916 faces. The Source points for the UTLM excitation are indicated by the selection of red and blue triangles in each case.

in Figure 6.13. The UTLM simulation is initiated in exactly the same manner as before; the same sinusoidal source is used, and the simulation is run for a total of 2×10^{-8} seconds. The results are shown in Figure 6.15, where the distribution of the magnetic field for each of the meshes is shown at the final time step of the simulation. These results align with the ones presented Figure 6.12. The magnetic field converges to the same peak magnitude as the original case, with only slight differences in the distribution of the field, attributed to the use of differing mesh densities, required to conform to the new curvature of the geometry.

Because the simulation is conducted in the 2D domain, where only the transverse magnetic field components are modelled, this, of course, is expected. However, if the flexion about the apex of the PCB results in a mechanical stress that inflicts a change in the electrical parameters at this point, this can still be reflected in the simulation results.

Figure 6.16 shows a top down view of the flexible PCB, meshed using 7916 faces. The electrical parameters at the apex of each wire, highlighted in green have been modified with a resistivity of $10M\Omega/m$ and a relative permittivity, ε_r, of 4. These values are chosen to demonstrate that a change of material parameters in the 3D domain result in a change in material parameters in the 2D domain, and the values have been amplified such that the effect can be visualised, as opposed to modelling realistic changes in this case, which can amount to small changes in the conductivity. The remaining material and simulation parameters are left unchanged. The mesh is parameterised, and the simulation is run again with the new material parameters mapped to the 2D domain.

Figure 6.17 shows the results after the same simulation time of 2×10^{-8} seconds. The results are visualised on the same scale as Figure 6.15 such that a direct comparison can be formed. The effect the material change at the apex of the geometry has on the magnetic field can clearly be seen. The propagating signal from the source is now met with a point of high resistivity, and the reflection from this point clearly results in a higher concentration of magnetic field between the source and area of flexion.

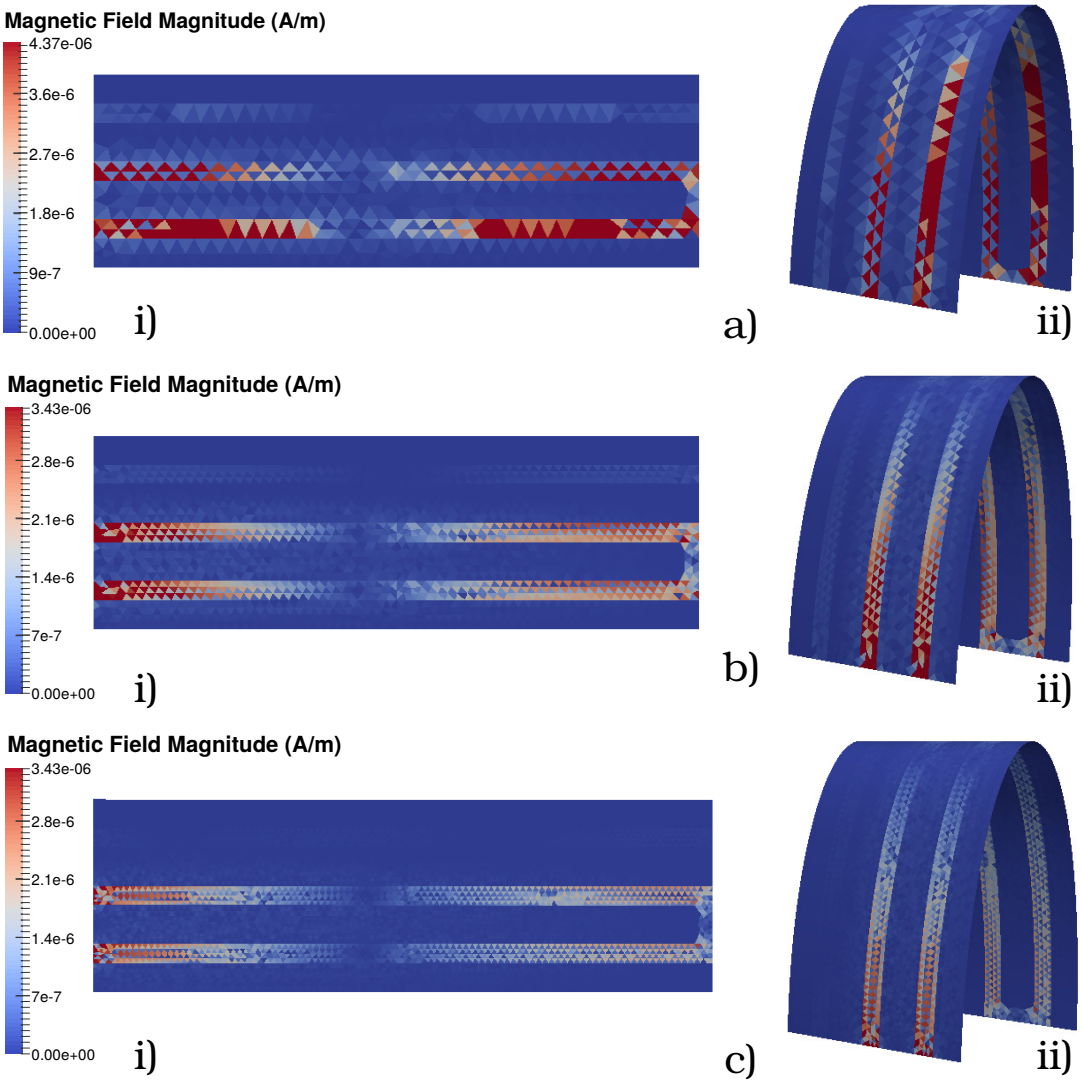

Figure 6.15: Resulting magnetic field magnitude from a UTLM simulation of the flexible PCB, flexed into a new curvature, after 2×10^{-8} seconds and the source is a sinusoidal 10V/m E_z excitation. The simulation is performed on the parameterised meshes i) constituting a) 1124, b) 2948 and c) 5796 faces before being mapped back to the 3D domain in ii). The wire tracks are copper, with FR-4 substrate.

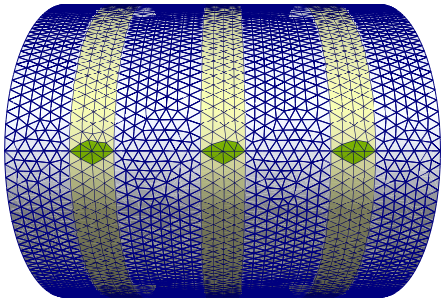

Figure 6.16: Flexible PCB with change in material parameters, conceptually representing a change in electrical properties as a result of mechanical strain.

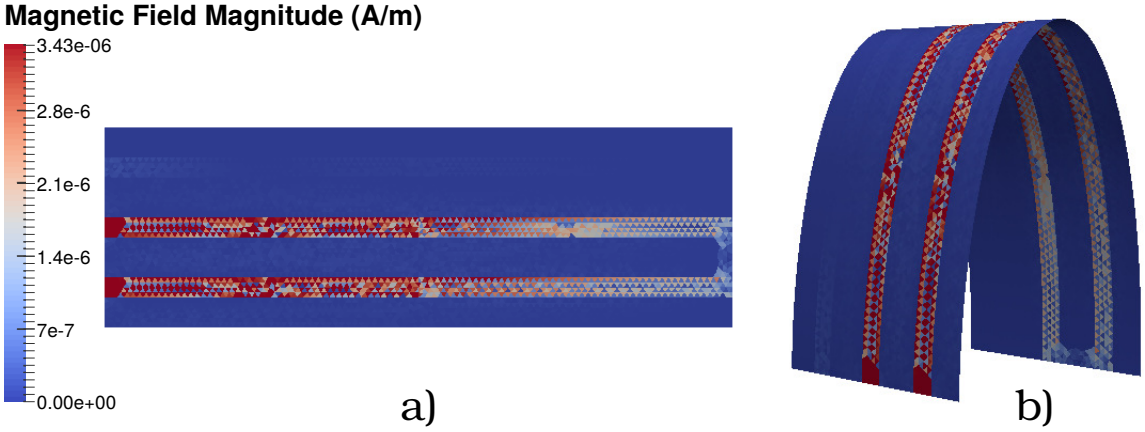

Figure 6.17: Transverse magnetic field distribution in A/m on the surface of the flexible PCB, after applying a change in electrical properties of the material. The material at the point of flexion has a resistivity of $10M\Omega$ and relative permittivity, $\varepsilon_r = 4$

6.4 Summary

This chapter presented the application of mesh parameterisation for UTLM simulations. For very thin and flexible geometries, which present themselves as 2D curved surfaces embedded in a 3D domain, low frequency electromagnetic simulations on these surfaces, using 3D algorithms may result in large portions of the memory and runtime being used to simulate areas of the domain that present little activity. By providing a one-to-one mapping of the geometry by parameterising the surface to the 2D domain, the need for computationally intensive 3D algorithms can be negated.

Four test cases have been presented; a hemispherical open surface, a flexible metallic sheet, a flexible PCB and the flexible PCB flexed into a new curvature.

Each geometry was parameterised to the 2D domain using the ABF method [6.1] before a UTLM simulation was conducted on the parameterised flat surface. The results for the simulations show convergence in the magnitude of the field, and also the distribution across the geometry as the mesh size is decreased.

The test case of the flexible PCB demonstrated that non-uniform materials can also be used. If the original surface in the 3D domain is altered in such a way that changes the electrical parameters of a material, perhaps through mechanical stress, or thermal fluctuations, where the resistivity of a material changes with a change in temperature, the application of mesh parameterisation for the purposes of a UTLM simulation can be useful. Changing the material parameters at the apex of the flexible PCB showed clear differences in the distribution of the magnetic field, compared to the original results. The following chapter concludes the work presented in this Thesis.

References

[6.1] A. Sheffer and E. de Sturler, "Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening," *Eng. Comput.*, vol. 17, no. 3, pp. 326–337, 2001.

[6.2] Teledyne, *Flexible Circuit Design Guide*, 4th ed. Teledyne Electronic Technologies, 2013.

[6.3] P. Sewell, T. M. Benson, C. Christopoulos, D. W. P. Thomas, A. Vukovic, and J. G. Wykes, "Transmission-Line Modeling (TLM) Based upon Unstructured Tetrahedral Meshes," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 6 I, pp. 1919–1928, 2005.

[6.4] J. R. Shewchuk, "Reprint of: Delaunay Refinement Algorithms for Triangular Mesh Generation," *Comput. Geom. Theory Appl.*, vol. 22, pp. 21–74, 2014.

[6.5] "COMSOL Multiphysics Simulation Software." [Online]. Available: https://uk.comsol.com/

[6.6] C. Smartt and U. o. N. GGIEMR, "GGI TLM." [Online]. Available: https://github.com/ggiemr/ggi_tlm

[6.7] R. A. Serway, *Principles of Physics*, 2nd ed. Fort Worth, Texas: London: Saunders College Pub, 1998.

[6.8] D. Giancoli, *Physics for Scientists and Engineers with Modern Physics*, 4th ed. Upper Saddle River, New Jersey: Prentice Hall, 2009.

Chapter 7

Conclusion

This chapter presents a general overview of the work documented in this thesis. The main conclusions derived from the different areas of the work completed will be discussed. In addition, areas for potential future investigations will be presented.

7.1 Overview of the Thesis

The thesis began in Chapter 2, which introduced the TLM method. This encompassed a description of the 1D and 2D formulations in a structured Cartesian environment. The discussion then moved on to the Unstructured 2D implementation of TLM, where the advantages and disadvantages compared to UTLMs structured kindred were presented. The advantage created by utilising the UTLM method is that when the boundary of the problem space is curved, the number of triangles required to accurately model the boundary can be reduced. This is due to the unstructured nature of the mesh when compared to the structured Cartesian model of the same boundary which must create a staircase approximation to the curvature.

This also creates a notable limitation; the UTLM nodes are no longer aligned in equal spacing in a Cartesian grid. Instead the placement of nodes are dictated by

a triangular mesh, resulting in varying lengths of the link lines that constitute the TLM network. This leads to a more computationally demanding analysis of the nodes at each time step of the TLM process.

This necessitated Chapter 3, which provided an investigation into mesh data structures. Fundamental to every scientific simulation is the representation of a geometrical problem space by a computer. This chapter discussed the methods that allow a computer to understand and represent the problem space in computer memory such that the algorithms which govern a simulation can be effectively applied. The methods discussed were the face-based, winged-edge and halfedge data structures; each facilitating the description of the connectivity of a mesh. Each method has differing impacts on the memory consumption necessary to represent the topology, and the impact on the runtime when executing algorithms on the mesh. The more a data structure understands the connectivity of a mesh, in terms of the adjacency information, the faster the mesh can be navigated, leading to a more efficient execution time of the UTLM algorithms. Consequently, this greater understanding of the adjacency information leads to a greater negative impact on the memory resources. The performance of a 2D UTLM simulation of a rectangular resonator was used to provide an experimental comparison between the different data structures discussed. It was shown that the face-based data structure consumed the least amount of memory to represent and store a mesh, compared to the edge-based and halfedge data structure, demonstrating a memory consumption one and a half times less than the halfedge paradigm. This low memory foot print is a result of a minimum amount of connectivity information needing to be stored, which consequently results in a much slower runtime of the UTLM simulation due to the significant amount of case distinctions in order to derive the mesh adjacency information. The storage of the full connectivity and adjacency information offered by the halfedge data structure naturally exhibited the largest memory footprint. However, this greater understanding of the adjacency information facilitated a faster runtime of the UTLM simulations.

No case-distinctions, in the form of *if-then* searches, in order to traverse the mesh as necessary, and as such, a two and a half time speed up was witnessed compared with the face-based data structure. The edge-based data structure provided a nice middle ground between memory consumption and runtime. It became clear that the underlying mesh data structures used to represent the geometrical problem space can have a huge impact on the efficiency and memory consumption of the simulation. This chapter concluded that it is not just simply the optimisation of the simulation algorithms that facilitate improvements to the runtime and memory consumption of a simulation. How a computer understands the connectivity of the mesh can have a far greater impact on the computational resources available.

Small scale fabrication processes have led to the advent of very thin flexible devices such as RFID tags, flexible PCBs and smart clothing. In a geometrical sense, these present themselves as curved two dimensional surfaces embedded in a three dimensional domain. When simulating electromagnetic behaviour on these surfaces at low frequencies, a full 3D field model is not always necessary. Using 3D algorithms to solve these problems can result in a large portion of the computer memory and runtime being used to mesh and simulate areas of the domain that present little electromagnetic activity.

This stimulated an investigation into the use of mesh parameterisation in order to reduce the computational resources necessary to run UTLM simulations. By providing a one-to-one mapping of a surface embedded in a 3D domain, to a 2D flat plane, a 2D UTLM simulation can be preformed, as opposed to using the more computationally intensive 3D algorithms.

Chapter 4 presented an introduction to mesh parameterisation and the techniques implemented for this project. These techniques can roughly be categorised into linear and non-linear methods. The linear techniques discussed were Uniform Weight parameterisation [7.1], Discrete Harmonic Parameterisation (DHP) [7.2], Mean Value

Parameterisation [7.3], and Least Squares Conformal Maps (LSCM) [7.4]. Most Isometric ParameterisationS (MIPS) [7.5], and Angle Based Flattening (ABF) [7.6] encompass the non-linear methods.

Chapter 5 provided an experimental comparison of the parameterisation techniques discussed in Chapter 4. The purpose of this was to realise the effects of metric distortion each parameterisation technique imposes, in addition to the amount of time each takes to converge to a mapping solution. The investigation was restricted to open surfaces, topologically homeomorphic to a disk. Test cases were generated and meshed, providing geometries that conform to this criterion in the 3D domain. These test cases were a hollow hemispherical surface, a flexible metal sheet deformed into an arbitrary shape, and a flexible PCB, consisting of 3 metal wire tracks separated by a substrate.

Each test case was parameterised using each parameterisation method, and subsequently analysed in terms of the induced metric distortion. Linear methods typically require the boundary of the 2D domain to be predefined and fixed prior to the parameterisation. If bijectivity is to be guaranteed, these boundaries are required to be regular, convex shapes such as circles and squares, which were the boundary definitions used in this experimentation. It was seen that the fixed boundary techniques exhibited significant amounts of distortion at the parameter of the 2D domain for all of the test cases, with one exception. The exception to this was the parameterisation of the hemispherical surface to a circle in the 2D domain, a result attributed to the fact that the boundary of the hemisphere in the 3D domain is itself circular, which allowed the mesh to be injected into the predefined 2D domain with minimal authalic and angular distortion.

The free boundary techniques, which include the parameterisation of the boundary vertices as part of the solution, facilitated a parameterisation resulting in very little distortion at the perimeter of the 2D domain.

The flexible metallic sheet, and the PCB test cases were both deformed once more, changing the curvature of the geometry. These were subsequently parameterised using the free boundary parameterisation techniques to provide a comparison with the original curvature. In the case of the flexible sheet, little difference was observed in terms of authalic and angular distortion, however the distribution of distortion to the lengths of the link lines that constitute a UTLM network were seen to be focused at the highest points of curvature. This distortion did increase slightly and again was distributed at the points of high curvature when the geometry was flexed into a new shape. The effect was repeated in the case of the flexible PCB, showing that the distribution of error in angle and area, and consequently TLM link lengths, accumulated at the apex of the geometry.

The magnitude of distortion induced by the free boundary methods was seen to be a maximum of 4% for very coarse meshes, and this was reduced to less than 1% as the mesh became more refined. The exception to this was the MIPS method [7.5] which was less predictable in its results, and tended to induce a higher amount of distortion than ABF [7.7] and LSCM [7.4] throughout. The level of complexity involved in the implementation of the MIPS method, coupled with the higher magnitude of distortion of the resultant parameterisations, lead to a conclusion that ABF and LSCM are the best approaches for providing mesh parameterisations.

ABF performed consistently well as a parameterisation method, ensuring that the maximum magnitude of distortion inflicted on TLM link lines was generally significantly less than 1% for moderately fine meshes, with a median distortion of less than a tenth of a percent. LSCM also performed comparably well to the ABF method, and offers a much simpler and linear solution.

Distinctions can be made between the ABF and LSCM methods when the runtime is considered. Timing each parameterisation method showed that all of the linear boundary parameterisations were very fast and efficient, including the free boundary

implementation provided by LSCM. These were not affected by the curvature of the geometry. In the case of the flexible metallic sheet and flexible PCB test cases, each linear method maintained a similar runtime. Differences were certainly witnessed when considering the non-linear methods ABF and MIPS. MIPS was consistently and significantly slower than the linear methods. ABF was able to provide parameterisations of the test geometries in a similar amount of time to the linear methods for coarse and moderately fine meshes. However for very fine meshes, a comparatively significant amount of time is required to arrive at a solution using ABF. It was also seen that a geometry with a high amount of curvature results in a longer runtime for the non-linear methods, which was seen in the case of the deformed flexible PCB.

This chapter concluded that the ABF method, although a non-linear solution, produces a parameterisation with the least amount of distortion, and consequently the smallest deviation from the original lengths of the link lines constituting the UTLM network. This was consistent across all of the test cases, which provided geometries of different curvatures and dimensions. For meshes which have a mesh density of more than 40000 faces, the LSCM method provides a good alternative. LSCM induces slightly more distortion than ABF, however, because of the linearity of the solution, a parameterisation is achieved in a significantly shorter amount of time for meshes with a very high face density.

Chapter 6 took the test cases from Chapter 5 and applied 2D UTLM simulations to the surfaces. Each test case was parameterised using the ABF method, and a 2D UTLM simulation is then conducted in 2D space. The results are subsequently mapped back to the original surfaces in the 3D domain for visualisation. The results of each test case demonstrated that the material parameters applied to the surface in the 3D domain persist through the parameterisation, with the simulation results converging as the mesh was refined. In the case of the flexible PCB, running UTLM

on the parameterised surface successfully facilitated the modelling of coupling between wires in the plane. Further, it was shown that deforming the PCB in such away that changes to the electrical properties of the material were induced, perhaps through mechanical stress, or thermal fluctuations, this was still reflected in the simulation results.

A notable limitation of mesh parameterisation for the purposes of UTLM is that because the simulation is performed in two dimensions, any coupling in the 3D sense is not modelled. Hence, the method of using mesh parameterisation is restricted to low frequency applications, where the wavelength of the incident source is greater than the dimensional order of the problem space. For higher frequency applications, where greater inductive coupling is expected, a full field profile would be required, and meshing the entire 3D domain becomes necessary.

7.2 Future Work

If the surface in the 3D domain is a closed surface, such as a sphere, or when acceptable levels of shear or stretch are not attainable because a surface is too complex, the surface needs to be cut prior to being parameterised in order to achieve acceptable distortion. For this initial investigation, the geometries were restricted to those that are topologically homeomorphic to a disk, whereby no cuts in the mesh are necessary to flatten the surface to the two-dimensional plane. By cutting the mesh prior to a UTLM simulation, new boundaries are introduced to the problem space. It is essential that fields remain continuous across these new boundaries.

Meshing is often the most difficult task of the simulation. Ensuring the quality of a triangulation for a surface is good quality for 2D UTLM simulation is taxing enough. Ensuring the quality of meshes for a 3D tetrahedral mesh is even harder. Using mesh

parameterisation as a tool for remeshing a surface and mapping the triangulation back to the 3D domain is an interesting method for providing a good quality surface triangulation as an input for generating a 3D tetrahedral mesh, for the purpose of 3D UTLM simulations. An investigation into the use of surface parameterisation for this purpose may provide interesting research results.

References

[7.1] W. Tutte, "How To Draw a Graph," *Proc. London Math. Soc*, vol. 8, no. May 1962, pp. 743–767, 1963.

[7.2] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '95*, vol. 6, pp. 173–182, 1995.

[7.3] M. S. Floater and M. S. Floater, "Convex Combination Maps," *Algorithms Approx. IV*, vol. 0, no. 0, pp. 18–23, 2002.

[7.4] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Trans. Graph.*, vol. 21, no. 3, 2002.

[7.5] K. Hormann and G. Greiner, "MIPS : An Efficient Global Parametrization Method," *Curve Surf. Des.*, pp. 153–162, 2000.

[7.6] A. Sheffer and E. De Sturler, "Surface Parameterization for Meshing by Triangulation Flattening," *Proc. 9th Int. Meshing Roundtable*, pp. 161–172, 2000.

[7.7] A. Sheffer and E. de Sturler, "Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening," *Eng. Comput.*, vol. 17, no. 3, pp. 326–337, 2001.