City Research Online

City, University of London Institutional Repository

Citation: Kabir, S.M. Raiyan (2015). Finite element time domain method with a unique coupled mesh system for electromagnatics and photonics. (Unpublished Doctoral thesis, City University London)

This is the accepted version of the paper.

This version of the publication may differ from the final published version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/14523/

Link to published version:

Copyright: City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

Reuse: Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

City Research Online: http://openaccess.city.ac.uk/ publications@city.ac.uk

Finite Element Time Domain Method with a Unique Coupled Mesh System for Electromagnatics and Photonics

CITY UNIVERSITY
LONDON

S M Raiyan Kabir

School of Mathematics, Computer Science & Engineering

City University London

This dissertation is submitted for the degree of

Doctor of Philosophy

2015

I would like to dedicate this thesis to my loving wife Anita,
my parents and my lovely little boy Nihan

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures.

S M Raiyan Kabir
2015

Acknowledgements

First and foremost I would like to thank almighty Allah for giving me the courage and patience to undertake and complete this work. I would like to dedicate this thesis to my son, my wife and my patents for their love and support throughout my life and for constantly encouraging me to pursue excellence. I am immensely indebted to Prof. B.M.A. Rahman for being an excellent and patient supervisor and for providing continuous support and encouragement over the years. I would also like to thank Prof. K.T.V. Grattan for his valuable advice and support throughout this research. I would also like to sincerely thank all my colleagues at the Photonics Research Group for maintaining a friendly and joyful work atmosphere.

Abstract

The finite difference time domain (FDTD) method is a popular technique, being used successfully to analyse the electromagnetic properties of many structures, including a range of optical or photonic devices. This method offers several major advantages such as, a minimum level of calculation is required for each of the cells into which the structure is divided, as well as data parallelism and explicit and easy implementation. However, due to the use of the Finite Difference grid, this method suffers from higher numerical dispersion and inaccurate discretisation due to staircasing at slanted and curve edges. The rectangular computational domain in 2D and cuboid computational domain in 3D sometimes makes the method very resource intensive especially for large simulations.

Although the finite element (FE) approach is superior for the discretisation of both 2D and 3D structures, most of the FE-based time domain approaches reported so far suffer from limitations due to the implicit or iterative form or the mass matrix formulation, for example. Therefore, the speed of the simulation is much slower than the FDTD method. Time domain analysis of electromagnetic is a very resource intensive numerical technique. Due to the slow performance the FE based techniques are not as popular as the FDTD method.

In this research work a new FE based time domain technique has been proposed for both 2D and 3D problems which is similar to the FDTD method explicit and data parallel in nature. The method proposed does not requires any matrix formulation or iteration. It uses minimum possible CPU cycles among any FE-based techniques. The method also utilises a unique meshing scheme to reduce the number of calculation to at least half for 2D and one fifth for 3D compared to any full mesh FE based technique.

The method also shows very low numerical dispersion when used with equilateral elements in both 2D and 3D. Thus the proposed method effectively produces results with less numerical dispersion error with lower density mesh compared to the FDTD method. When the advantage in resolution is taken into consideration, calculation of each time-step using the proposed method is significantly faster than the FDTD method.

Contents

Contents v

List of Figures ix

List of Tables xi

Nomenclature xvii

1 Introduction 1

1.1 Numerical Methods for Electromagnetics 1

1.2 Maxwell's Equations 2

1.2.1 Integral Form 2

1.2.2 Differential Form 3

1.2.3 The Wave Equation 4

1.3 Numerical Analysis Techniques 5

1.3.1 Modal Analysis 5

1.3.2 Beam Propagation Method 6

1.3.3 Frequency Domain Analysis 7

1.3.4 Time Domain Analysis 7

1.4 Motivation for the Research 13

I Two-dimensional Formulation 15

2 Derivation of Governing Equations for Two-dimensions 16

2.1 Discretisation 19

2.1.1 Space Discretisation 19

2.1.2 Time Discretisation 20

3 The Two-dimensional Mesh 23
3.1 The Space Mesh System . . . 25
3.1.1 Completeness of the Mesh . . . 27
3.2 The Time Mesh System . . . 28

4 Perfectly Matched Layer Boundary for Two-dimensions 30
4.1 Large Computational Domain . . . 30
4.2 Absorbing Boundary Condition . . . 31
4.2.1 Methods Based on Oneway Wave Equations . . . 31
4.2.2 Perfectly Matched Layer Boundary . . . 32
4.3 X Axis PML . . . 35
4.4 Y Axis PML . . . 36
4.5 Corner PML . . . 37
4.6 Placing the PML in the Computational Domain . . . 39

5 Dispersive Materials 41
5.1 Maxwell's Equations with Drude Model . . . 43
5.2 Governing Equations for Metal . . . 44
5.3 Governing Equations for Metamaterial . . . 45

6 Results of Simulations in Two-dimensions 47
6.1 Free Space Propagation . . . 48
6.2 Planar Waveguide . . . 49
6.3 Metamaterial Flat Lens & Backward Propagation . . . 52

II Three-dimensional Formulation 57

7 Governing Equations for Three-dimensions 58
7.1 Space-Time discretisation . . . 59
7.1.1 Space discretisation . . . 59
7.1.2 Time discretisation . . . 65
7.1.3 Discretised Governing Equations . . . 65

8 The Three-dimensional Mesh 67
8.1 The Space Mesh . . . 67
8.2 The Time Mesh . . . 70

9 Perfectly Matched Layer Boundary for Three-dimensions 71
9.1 X PML . . . 72
9.2 Y PML . . . 73
9.3 Z PML . . . 74
9.4 XY PML . . . 75
9.5 YZ PML . . . 76
9.6 ZX PML . . . 77
9.7 XYZ PML . . . 78

10 Results of Simulations in Three-dimensions 80
10.1 Free Space Propagation . . . 80
10.2 Nanowires . . . 83
10.3 Nano Power Splitter . . . 85
10.4 Nano Directional Coupler . . . 89
10.4.1 Full Vectorial FEM Analysis . . . 90
10.4.2 Calculation of Coupling Length using the FETD 3D . . . 91

III Performance Analysis 94

11 Numerical Dispersion 95
11.1 Numerical Dispersion for 2D Formulation . . . 96
11.1.1 Calculation of Numerical Dispersion . . . 99
11.1.2 Calculating Resolution Reduction Factor . . . 101
11.1.3 Comparing Numerical Dispersion of Meshes by Simulation . . . 103
11.1.4 Comparison with the FDTD Method . . . 107
11.2 Numerical Dispersion for 3D Formulation . . . 109
11.2.1 Calculation of Numerical Dispersion . . . 113
11.2.2 Calculation of Resolution Reduction Factor . . . 115
11.2.3 Comparison with the FDTD Method . . . 117

12 Theoretical CPU Performance 121
12.1 CPU Performance for Two-dimensionals Formulation . . . 123
12.1.1 CPU Optimised Formulation of the FDTD in 2D . . . 123
12.1.2 CPU Optimised Formulation for Proposed FETD in 2D . . . 126
12.1.3 Comparing Proposed FETD and FDTD considering RRF in 2D . . . 130
12.2 CPU Performance for Three-dimensionals Formulation . . . 134

12.2.1 CPU Optimised Formulation of the FDTD in 3D 134
12.2.2 CPU Optimised Formulation for Proposed FETD in 3D 137
12.2.3 Comparing Proposed FETD and FDTD considering RRF in 3D . . 141

IV Future Plan and Conclusions 144

13 Future Works 145
13.1 Unstructured Mesh 145
13.2 Variable Time-stepping 147
13.3 Partial Mesh Solution for Pulse Propagation 148
13.4 Higher Order Implementation 148

14 Conclusion 149

References 151

List of Figures

1.1 Yee's Lattice 8

3.1 Space mesh system for 2D formulation 26
3.2 Coupled Time mesh system 29

4.1 Placement of PML in computational domain 40

6.1 Freespace propagation 47
6.2 Computational domain shown with PML. Expansion of the E_z field with time (a) after 1000 time steps, (b) 2000 time steps and (c) 3000 time steps . 49
6.3 Schematic diagram of the planner waveguide setup 50
6.4 Propagation and verification in a waveguide 51
6.5 Schematic setup and double focusing with a DNG flat lens 53
6.6 Start of EM wave propagation in metamaterial slab 54
6.7 EM wave propagation inside metamaterial slab 56

8.1 A basic tetrahedral element inside a 3D cubic cell 67
8.2 Tetrahedral Dual Mesh System 68

10.1 Free-space propagation of **E** and **H** components from a point source 81
10.2 Line plot over all three axis for free space propagation 82
10.3 Propagation in a *Si* nanowire 84
10.4 Propagation in a *Si* nano power splitter 86
10.5 H_y field profile at the input and output of the *Si* nano power splitter 88
10.6 Schematic diagram of the nano directional coupler 89
10.7 Odd and Even mode associated with the nano directional coupler 90
10.8 Modelling the nano directional coupler using the proposed 3D FETD with nanowire separation of $s = 0.1\mu m$ after 2450 time-steps 91

10.9 Comparison of coupling length obtained from FVFEM and the proposed 3D FETD method 92

11.1 Calculation of numerical dispersion in IRT and ET meshes 99
11.2 Comparison of phase velocity of the IRT and ET mesh 101
11.3 Comparison of the numerical dispersion of the ET and the IRT meshes . . . 102
11.4 Resolution Reduction Factor 103
11.5 Simulation results of the 2D FETD with the IRT and the ET meshes 105
11.6 E_z field profile after 2000 time steps with the FDTD method (in 2D) 108
11.7 Element arrange meant for both the IRT3D and ET3D mesh systems 113
11.8 Comparison of normalised v_p of the IRT3D and the ET3D meshes 114
11.9 Mean and Standard Deviation of the ET3D and the IRT3D meshes 116
11.10 Resolution Relation of the ET3D and the IRT3D meshes 117
11.11 Comparison of normalised v_p of the 3D FDTD and the 3D FETD 118
11.12 Mean and Standard Deviation of the 3D FDTD and the 3D FETD 120

12.1 CPU latency (non-SIMD) comparison between FDTD2D and FETD2D . . 131
12.2 CPU latency (SIMD) comparison between FDTD2D and FETD2D 132
12.3 CPU latency comparison between FDTD3D and FETD3D 142

List of Tables

12.1 Compute Operations and Latencies for 2D FDTD Method with General Purpose Instructions 125
12.2 Compute Operations and Latencies for 2D FDTD Method with General Purpose and SIMD Instructions 126
12.3 Compute Operations and Latencies for 2D FETD Method with General Purpose Instructions 127
12.4 Compute Operations and Latencies for 2D FETD Method with General Purpose and SIMD Instructions 129
12.5 Compute Operations and Latencies for 3D FDTD Method with General Purpose Instructions 136
12.6 Compute Operations and Latencies for 3D FDTD Method with General Purpose and SIMD Instructions 137
12.7 Compute Operations and Latencies for 3D FETD Method with General Purpose Instructions 138
12.8 Compute Operations and Latencies for 3D FETD Method with General Purpose and SIMD Instructions 140

Nomenclature

Roman Symbols

$\mathbf{B}$	Vector Magnetic Flux
c	Speed of Light in free-space
$\mathbf{D}$	Vector Electric Flux
$\mathbf{E}$	Vector Electric Field
$\mathscr{E}_{x[y]}$	Auxiliary field generated for X PML, XY PML, ZX PML and XYZ PML
$\mathscr{E}_{x[z]}$	Auxiliary field generated for X PML, XY PML, ZX PML and XYZ PML
$\mathscr{E}_{y[x]}$	Auxiliary field generated for Y PML, XY PML, YZ PML and XYZ PML
$\mathscr{E}_{y[z]}$	Auxiliary field generated for Y PML, XY PML, YZ PML and XYZ PML
$\mathscr{E}_{z[x]}$	Auxiliary field generated for Z PML, YZ PML, ZX PML and XYZ PML
$\mathscr{E}_{z[y]}$	Auxiliary field generated for Z PML, YZ PML, ZX PML and XYZ PML
E_x	x-directional component of the Electric Field
e_x	x-directional Electric Field component at a node point
E_y	y-directional component of the Electric Field
e_y	y-directional Electric Field component at a node point
E_z	z-directional component of the Electric Field
e_z	x-directional Electric Field component at a node point
$\mathbf{H}$	Vector Magnetic Field

$\mathscr{H}_{x[y]}$	Auxiliary field generated for X PML, XY PML, ZX PML and XYZ PML
$\mathscr{H}_{x[z]}$	Auxiliary field generated for X PML, XY PML, ZX PML and XYZ PML
$\mathscr{H}_{y[x]}$	Auxiliary field generated for Y PML, XY PML, YZ PML and XYZ PML
$\mathscr{H}_{y[z]}$	Auxiliary field generated for Y PML, XY PML, YZ PML and XYZ PML
$\mathscr{H}_{z[x]}$	Auxiliary field generated for Z PML, YZ PML, ZX PML and XYZ PML
$\mathscr{H}_{z[y]}$	Auxiliary field generated for Z PML, YZ PML, ZX PML and XYZ PML
H_x	x-directional component of the Magnetic Field
h_x	x-directional Magnetic Field component at a node point
H_y	y-directional component of the Magnetic Field
h_y	y-directional Magnetic Field component at a node point
H_z	z-directional component of the Magnetic Field
h_z	z-directional Magnetic Field component at a node point
$\mathbf{J}$	Vector Current Density
k_0	Wavenumber
$\mathbf{M}_e$	Auxiliary Vector Field for Drude Dispersion Model
M_{e_x}	x-directional component of $\mathbf{M}_e$
M_{e_y}	y-directional component of $\mathbf{M}_e$
M_{e_z}	z-directional component of $\mathbf{M}_e$
$\mathbf{M}_m$	Auxiliary Vector Field for Drude Dispersion Model
M_{m_x}	x-directional component of $\mathbf{M}_m$
M_{m_y}	y-directional component of $\mathbf{M}_m$
M_{m_z}	z-directional component of $\mathbf{M}_m$
n	Refractive index of the medium

n_{eff}	Effective Refractive Index
N_i	Shape function for spatial discretisation
Q_j	Shape function for time discretisation

Greek Symbols

β	Propagation constant of an electromagnetic wave
$\nabla\times$	Curl Operator
$\nabla\cdot$	Divergence Operator
ε	Permittivity of medium
ε_0	Permittivity of free-space
ε_r	Relative permittivity of the medium
γ_e	Electric Collision Frequency
γ_m	Magnetic Collision Frequency
μ	Permeability of medium
μ_0	Permeability of free-space
μ_r	Relative permeability of the medium
∇	Gradient Operator
$\tilde{\nabla}_x$	Modified Gradient operator for X PML (for 3D)
$\tilde{\nabla}_{xyz}$	Modified Gradient operator for XYZ PML (for 3D)
$\tilde{\nabla}_{yz}$	Modified Gradient operator for YZ PML (for 3D)
$\tilde{\nabla}_{zx}$	Modified Gradient operator for ZX PML (for 3D)
$\tilde{\nabla}^{(x)}$	Modified Gradient operator for X PML (for 2D)
$\tilde{\nabla}_{xy}$	Modified Gradient operator for XY PML (for 3D)
$\tilde{\nabla}_y$	Modified Gradient operator for Y PML (for 3D)

$\tilde{\nabla}^{(y)}$	Modified Gradient operator for Y PML (for 2D)
$\tilde{\nabla}_z$	Modified Gradient operator for Z PML (for 3D)
ω	Angular frequency of Electromagnetic Radiation
ω_{pe}	Electric Plasma Frequency
ω_{pm}	Magnetic Plasma Frequency
$\Omega_{z[xy]}$	Auxiliary field generated for Corner PML
Φ_x	Auxiliary field generated for YZ PML and XYZ PML
Φ_y	Auxiliary field generated for ZX PML and XYZ PML
Φ_z	Auxiliary field generated for XY PML and XYZ PML
$\Phi_{x[x]}$	Auxiliary field generated for X PML and Corner PML
$\Psi_{y[y]}$	Auxiliary field generated for Y PML and Corner PML
Ψ_x	Auxiliary field generated for YZ PML and XYZ PML
Ψ_y	Auxiliary field generated for ZX PML and XYZ PML
Ψ_z	Auxiliary field generated for XY PML and XYZ PML
$\Psi_{x[x]}$	Auxiliary field generated for X PML and Corner PML
$\Psi_{y[y]}$	Auxiliary field generated for Y PML and Corner PML
σ_x	Absorption coefficient function of PML in x direction
σ_y	Absorption coefficient function of PML in y direction
$\Theta_{z[xy]}$	Auxiliary field generated for Corner PML

Acronyms / Abbreviations

1D	One dimensions
2D	Two dimensions
3D	Three dimensions

Add	Addition
BPM	Beam Propagation Method
CPU	Central Processing Unit
DNG	Double Negative, meaning both ε and μ are negative
EM	Electromagnetic
ET3D	Equilateral Tetrahedral
ET	Equilateral Triangle
FD	Finite Difference
FDTD	Finite Difference Time Domain
FE	Finite Element
FEM	Finite Element Method
FETD	Finite Element Time Domain
FFT	Fast Fourier Transformation
Ins.	CPU Instructions
IRT3D	Isosceles Right Angled Tetrahedron
IRT	Isosceles Right-angled Triangle
Late.	Latency or required CPU Cycles
Mult	Multiplication
PEC	Perfect Electric Conductor
PML	Perfectly Matched Layer
RRF	Resolution Reduction Factor
SIMD	Single Instruction Multiple Data
Sub	Subtraction

TE	Transverse Electric
TM	Transverse Magnetic
VFEM	Full-vectorial Finite Element Method
VTK	Visualisation Tool Kit
X PML	PML Absorbing Radiation in x-direction (for 2D and 3D)
Corner PML	PML Absorbing Radiation both in x and y-direction (for 2D)
XY PML	PML Absorbing Radiation in x and y-directions (for 3D)
XYZ PML	PML Absorbing Radiation in x, y and z-directions (for 3D)
Y PML	PML Absorbing Radiation in y-direction (for 2D and 3D)
YZ PML	PML Absorbing Radiation in y and z-directions (for 3D)
Z PML	PML Absorbing Radiation in z-direction (for 3D)
ZX PML	PML Absorbing Radiation in z and x-directions (for 3D)

Chapter 1

Introduction

Numerical analysis and modelling are used in many branches of engineering and physics. Numerical techniques enable researchers, device designers and engineers to characterise a theoretical representation of a physical device under specific condition without even manufacturing it. This reduce both cost and time of development as numerical analysis of a physical device only require a capable enough computer system to run the software for sufficient time to produce the result which can be used for design and optimisation before going to physical production [1]. Moreover, numerical techniques allow engineers and system designers to make decision by simulation before purchasing any already manufactured products.

1.1 Numerical Methods for Electromagnetics

For electromagnetics device design and development, numerical modelling plays a big role. Many advanced and recently discovered phenomena are discovered numerically in conjunction with physical experiments [2, 3]. Use of commercial software such as, COMSOL Multi-Physics, HFSS, XFDTD, Lumerical, RSOFT, Photon Design etc. are widely used by numerical and experimental research groups, device designers and device manufacturers. Efficient and powerful numerical methods are a basic requirements for fast and low cost development and production of electromagnetic and photonics devices and systems.

With the advancement of the computer many numerical techniques have developed. Most of this techniques use the Maxwell's equations in different forms to solve the problem domain. Some of the more widely used methods are Galerkin and moment method [4–7], transfer matrix method [8], transmission line matrix method [9, 10], finite difference based methods [11–14], finite element based methods [15, 16] and finite volume method [17]. All these methods solve the Maxwell's equations to which is the starting point of all form of analytical or numerical analysis of electromagnetic structures.

1.2 Maxwell's Equations

The set of equations named after James Clerk Maxwell. These equations describe the relation between electric and magnetic fields in an electromagnetic radiation. Maxwell's equations can be presented in integral or differential form.

1.2.1 Integral Form

Integral or large scale form of Maxwell's equations are used in some finite difference algorithms [18, 19] and integral methods like finite integration method [20–22]. The integral form of Maxwell's equations corresponding to the equations from Eqs. 1.1 are as follows [23],

$$\oint_S \mathbf{D}\cdot\mathbf{dS} = \int_V \rho\, dV \tag{1.1a}$$

$$\oint_S \mathbf{B}\cdot\mathbf{dS} = 0 \tag{1.1b}$$

$$\oint \mathbf{E}\cdot\mathbf{dl} = -\frac{\partial}{\partial t}\int_S \mathbf{B}\cdot\mathbf{dS} \tag{1.1c}$$

$$\oint \mathbf{H}\cdot\mathbf{dl} = \int_S \mathbf{J}\cdot\mathbf{dS} + \frac{\partial}{\partial t}\int_S \mathbf{D}\cdot\mathbf{dS} \tag{1.1d}$$

where, **dS** is a vector and denotes the differential surface element S. Similarly **dl** is a vector denoting differential line element l.

1.2.2 Differential Form

The differential form of Maxwell's equations consist of four first order differential equations as follow [23],

$$\nabla \cdot \mathbf{D} = \rho \tag{1.2a}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.2b}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{1.2c}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} \tag{1.2d}$$

In these equations **E** is the vector electric field, **H** is the vector magnetic field, **D** is the vector electric flux density, **B** is the vector magnetic flux density, **J** is the current density of the medium and ρ is the charge density of the medium.

The relation between the electric and magnetic flux and the field intensity can be defined as,

$$\mathbf{B} = \mu \mathbf{H} \tag{1.3a}$$

$$\mathbf{D} = \varepsilon \mathbf{E} \tag{1.3b}$$

μ and ε are permeability and permittivity of the medium respectively. Both μ and ε can be constant, tensor or functional for isotropic, anisotropic or dispersive materials respectively.

This form of Maxwell's equations are more widely used compared to the integral form. Specially, the wave equation derived from the differential form of Maxwell's equations are more popular for different finite difference and finite element analysis [1, 4, 16, 17, 24, 25].

1.2.3 The Wave Equation

For many problems, the solution of the coupled equations presented in Eqs. 1.2 and Eqs. 1.1 are not very easy [26]. Sometimes it becomes difficult to implement as a computer program. Therefore, it is a common practice for many algorithms to decouple the Maxwell's first order partial differential equations presented in Eqs. 1.2 into a second order partial differential equation consisting of only one field (either **E** or **H**) [1].

To obtain the second order equation, at first the **B** from Eq. 2.3a could be substituted with the corresponding value in Eq. 1.3a.

$$\frac{1}{\mu_r}\nabla \times \mathbf{E} = -\mu_0 \frac{\partial \mathbf{H}}{\partial t} \tag{1.4}$$

where, μ_0 and μ_r are the permeability of vacuum and relative permeability of the medium respectively and $\mu = \mu_0\mu_r$.

After applying curl operator on both side of Eq. 1.4 the following equation is obtained,

$$\nabla \times \left(\frac{1}{\mu_r}\nabla \times \mathbf{E}\right) = -\mu_0 \frac{\partial(\nabla \times \mathbf{H})}{\partial t} \tag{1.5}$$

By substituting $\nabla \times \mathbf{H}$ in Eq. 1.5, with Eq. 2.3b and Eq.1.3b for charge free region (where both $\rho = 0$ and $\mathbf{J} = 0$) the wave equation with **E** field can be derived as,

$$\nabla \times \left(\frac{1}{\mu_r}\nabla \times \mathbf{E}\right) = -\left(\frac{\varepsilon_r}{c^2}\right)\frac{\partial^2 \mathbf{E}}{\partial t^2} \tag{1.6}$$

where, ε_0 and ε_r are the permittivity of the vacuum and the relative permittivity of the medium respectively and $\varepsilon = \varepsilon_0\varepsilon_r$. The speed of light in vacuum $c = 1/\sqrt{\mu_0\varepsilon_0}$. This is the wave equation with the electric field components.

Similarly, the wave equation for the magnetic field components can be obtained as,

$$\nabla \times \left(\frac{1}{\varepsilon_r}\nabla \times \mathbf{H}\right) = -\left(\frac{\mu_r}{c^2}\right)\frac{\partial^2 \mathbf{H}}{\partial t^2} \tag{1.7}$$

The wave equations presented in Eq. 1.6 and Eq. 1.7 are very popular among many numerical methods because the wave equation consist of only one field and it decouples

the **E** from **H** or *vice versa*. The amount of storage for field components are half of the coupled equations. As there are only one field associated with the equation, handling the boundary and discretisation of the computational domain is easier compared to the coupled equations [1].

1.3 Numerical Analysis Techniques

There are several techniques to evaluate different aspect of electromagnetic and photonic devices. Following are some of the widely used numerical analysis technique for device design, characterisation and discovery of new and novel photonic devices.

1.3.1 Modal Analysis

Modal analysis consists of a big part of numerical analysis of electromagnetic and photonic devices. This is a eigenvalue value problem. In modal analysis different shape of guide are analysed. The cross section perpendicular to the direction of propagation is taken into consideration. The structure is assumed to be uniform in the direction of propagation. Therefore, only waveguide in 2D (cross section will be a line) or 3D (cross section will be a plane surface) can be analysed using the mode solver. The method analyses one frequency at a time.

The solver can be used with both scalar and full-vectorial formulation. The scalar equations for **E** and **H** field components are [27],

$$L_\Phi = \int\int_\Omega \left[\left(\frac{\partial \Phi}{\partial x}\right)^2 + \left(\frac{\partial \Phi}{\partial y}\right)^2 - k_0^2 n^2 \Phi^2 + \beta^2 \Phi^2 \right] d\Omega \tag{1.8a}$$

$$L_\Psi = \int\int_\Omega \left[\frac{1}{n^2}\left(\frac{\partial \Psi}{\partial x}\right)^2 + \frac{1}{n^2}\left(\frac{\partial \Psi}{\partial y}\right)^2 - k_0^2 \Psi^2 + \frac{1}{n^2}\beta^2 \Psi^2 \right] d\Omega \tag{1.8b}$$

Here, Φ can be E_x, E_y or E_z and Ψ can be H_x, H_y or H_z, n is the refractive index of the material, β is the propagation constant and k_0 is the wavenumber.

One of the equation for full-vectorial formulation for **H** field is as follows [28, 29],

$$\omega^2 = \frac{\int (\nabla \times \mathbf{H})^* \cdot \frac{1}{\varepsilon} \cdot (\nabla \times \mathbf{H})\, d\Omega}{\int \mathbf{H}^* \cdot \mu \cdot \mathbf{H} d\Omega} \tag{1.9}$$

Here, ω is the angular frequency of the EM radiation.

When expanded in appropriate solution technique, the method produces an eigenvalue problem which could be solved by an eigenvalue solver.

The results produced by the method are the effective refractive index of the guide and the field distribution for different components of **E** and **H** fields. Although the outputs of the analysis are simple, they can be used to calculate the dispersion, loss, bending loss, coupling length of directional couplers. Also identify mode degeneration etc. Therefore, the modal analysis is a very important tool for design and optimisation of electromagnetic and photonic devices. Both the finite difference and finite element techniques are used to solve this type of problems. [11, 15, 24, 28].

1.3.2 Beam Propagation Method

The beam propagation method (BPM) simulates a given beam of light evolving through a device in the propagation direction. This is an initial value problem. The evolution of the field will depend both on the input field and the allowable guiding modes of the structure. The method allows variation in the propagation direction. Therefore, more devices can be analysed. The BPM technique can be used for non-linear analysis of photonic devices. This method considers one frequency at a time.

Initially in 80s the BPM was based on Fast Fourier Transformation (FFT). In 1996, Tsuji and Koshiba [30] presented a Finite Element (FE) based BPM. Then in 2000, Obayya *et al.* presented a full vectorial BPM for 3-D optical waveguides.

The wave equation presented in Eqs. 1.6 and 1.7 can be expressed as,

$$\frac{\partial^2}{\partial z^2} \mathbf{\Psi} = \Theta \mathbf{\Psi} \tag{1.10}$$

Here, vector $\mathbf{\Psi}$ is either electric or magnetic field and Θ is the operator containing the

transverse space derivative and the refractive index variation [1].

The BPM algorithms transforms the wave equation into a parabolic equation to approximate the propagating wave. It using the first derivative to construct a marching algorithms which takes the initial field provided at the input and march it through the device along the initial direction of propagation of the input field and produce the expected field at the output [31, 32].

Apart from the above mentioned method there are other methods to solve excitation-response problem *i.e.* the Scattering approach.

1.3.3 Frequency Domain Analysis

The frequency domain analysis uses the wave equation presented in Eqs. 1.6 and 1.7 and solved them to obtain the field distribution of a specific frequency in the computational domain. This is a boundary value problem. Therefore, the distribution of field inside the device is dependent on the boundary condition imposed at the boundary of the computational domain and the refractive index variation inside the domain [23]. This method is applicable for both two-dimensional and three-dimensional structures.

1.3.4 Time Domain Analysis

All methods discussed up to now produce a steady state field distribution at one specific frequency. Transient or time varying response of the photonic cannot be studied by the above mentioned methods. Study of broadband characteristics of is also difficult with above methods. As any broadband signal will contain more than one frequency and the shape of the signal may vary with time depending on the type of the signal used to excite the device. To tackle such problems, time domain analysis methods are necessary. The time domain analysis of EM and photonics started with Yee's algorithm [14] for Finite Difference analysis of time domain problems. Followed by Finite Integration [20–22], Finite Volume [17, 33] and Finite Element [27, 34–40] based method.

Finite Difference Time Domain Method

In 1966 Yee [14] proposed a finite difference based technique to solve Maxwell's equations over time to analyse time domain properties of Electromagnetics. The method he proposed is widely known as Finite Difference Time Domain or the FDTD method. The methods solves the Maxwell's equations (Eqs. 1.2) in their differential form on a rectangular gird for 2D and cuboid grid for 3D. The method uses a special staggered distribution of field components to solve the Maxwell's coupled equations. Figure 1.1 shows the distribution of field component in 3-dimensions in Yee's lattice.

The method calculates **E** and **H** field components at different space nodes and **E** and **H** fields at different time-steps.

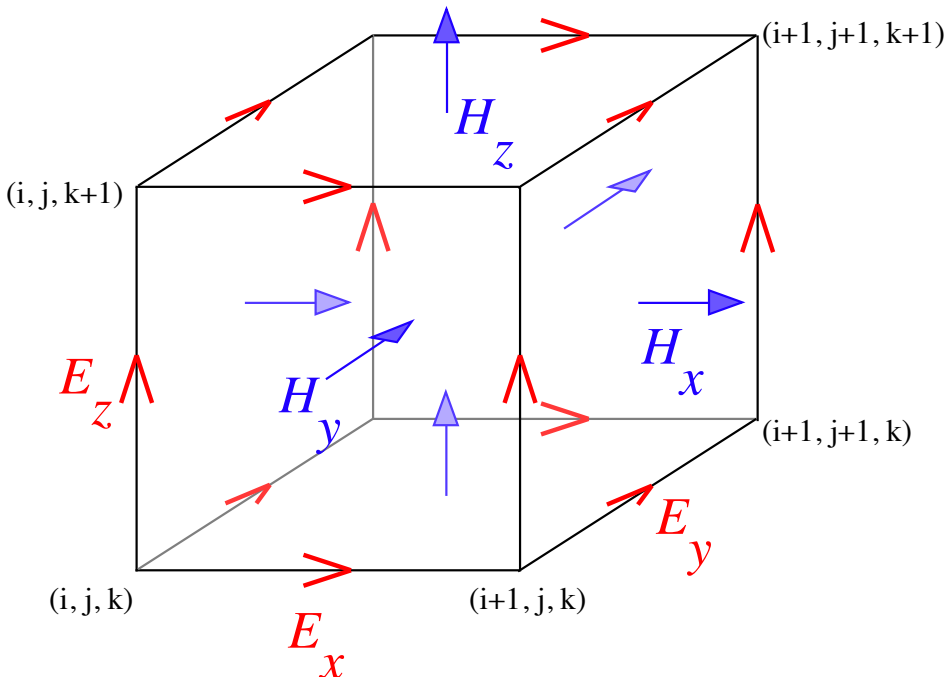


Fig. 1.1 Field component distribution in Yee's lattice in 3D (image taken from Wikipedia)

The equations the method solves for 3D are as follows,

$$H_x|_{i,j,k}^{n+1/2} = -\frac{\Delta t}{\mu_{i,j,k}}\left[\frac{E_z|_{i,j+1/2,k}^{n} - E_z|_{i,j-1/2,k}^{n}}{\Delta y} - \frac{E_y|_{i,j,k+1/2}^{n} - E_y|_{i,j,k-1/2}^{n}}{\Delta z}\right] + H_x|_{i,j,k}^{n-1/2} \tag{1.11a}$$

$$H_y|_{i,j,k}^{n+1/2} = \frac{\Delta t}{\mu_{i,j,k}}\left[\frac{E_z|_{i+1/2,j,k}^{n} - E_z|_{i-1/2,j,k}^{n}}{\Delta x} - \frac{E_x|_{i,j,k+1/2}^{n} - E_x|_{i,j,k-1/2}^{n}}{\Delta z}\right] + H_y|_{i,j,k}^{n-1/2} \tag{1.11b}$$

$$H_z|_{i,j,k}^{n+1/2} = -\frac{\Delta t}{\mu_{i,j,k}}\left[\frac{E_y|_{i+1/2,j,k}^{n} - E_y|_{i-1/2,j,k}^{n}}{\Delta x} - \frac{E_x|_{i,j+1/2,k}^{n} - E_x|_{i,j-1/2,k}^{n}}{\Delta y}\right] + H_z|_{i,j,k}^{n-1/2} \tag{1.11c}$$

$$E_x|_{i,j,k}^{n+1/2} = \frac{\Delta t}{\varepsilon_{i,j,k}}\left[\frac{H_z|_{i,j+1/2,k}^{n} - H_z|_{i,j-1/2,k}^{n}}{\Delta y} - \frac{H_y|_{i,j,k+1/2}^{n} - H_y|_{i,j,k-1/2}^{n}}{\Delta z}\right] + E_x|_{i,j,k}^{n-1/2} \tag{1.11d}$$

$$E_y|_{i,j,k}^{n+1/2} = -\frac{\Delta t}{\varepsilon_{i,j,k}}\left[\frac{H_z|_{i+1/2,j,k}^{n} - H_z|_{i-1/2,j,k}^{n}}{\Delta x} - \frac{H_x|_{i,j,k+1/2}^{n} - H_x|_{i,j,k-1/2}^{n}}{\Delta z}\right] + E_y|_{i,j,k}^{n-1/2} \tag{1.11e}$$

$$E_z|_{i,j,k}^{n+1/2} = \frac{\Delta t}{\varepsilon_{i,j,k}}\left[\frac{H_y|_{i+1/2,j,k}^{n} - H_y|_{i-1/2,j,k}^{n}}{\Delta x} - \frac{H_x|_{i,j+1/2,k}^{n} - H_x|_{i,j-1/2,k}^{n}}{\Delta y}\right] + E_z|_{i,j,k}^{n-1/2} \tag{1.11f}$$

Here, Δx, Δy and Δz are the space step size in x, y and z direction. Δt is the time step size. $n-1/2$, n and $n+1/2$ are half time step past, current time step and half time step future. i, j and k are the indexes used in Fig. 1.1.

Advantages:

Local Solution: The FDTD method does not create any global matrix to find the next time step. All calculations are done locally to each Yee's lattice. Therefore, the only memory it occupies are the memories for **E** and **H** field components and the properties of the material for each cell in the computational domain. Hence, the method's memory is requirement very low compared to methods forming global matrices.

Explicit Solution: Explicit methods computes the state of a system at a future time from the state of the system at the current time, while implicit methods find a solution by solving an equation involving both the current state of the system and the future state. Mathematically, if $S(t)$ is the current state of a system then an explicit method will calculate the future state $S(t+\Delta t)$ by solving a governing equation similar to the following,

$$S(t+\Delta t) = F(S(t)) \tag{1.12}$$

On the other hand a implicit method would solve an equation similar to the following,

$$G(S(t), S(t+\Delta t)) = 0 \tag{1.13}$$

It can be easily understood that, the implicit solver has to iterate multiple times over the system to find the solution as one of the input of the governing equation is unknown. However, the explicit solver produces the future state with a single step.

But the quick solution advantage of the explicit method comes with a stability condition which limits the size of the discretisation steps of the method. Implicit methods are unconditionally stable. Therefore, allows larger step sizes.

The FDTD equations are explicit in nature. As a result, the formulation can calculate the field distribution for next time-step accurately within its stability limits only by executing once. Therefore, the method is faster than iterative and implicit techniques for many problems.

Data Parallel: The FDTD method solves the evolution of field locally for each cell. The method takes **E** field components and calculate the **H** components from them. Calculation of **H** field components does not depend on any of the current **H** field values. In fact, all current values used in this algorithm are **E** field component values. The only **H** component necessary is the previous value of the same component. Calculation process of the **E** components are similar. Therefore, all governing equations are independent of each other. Which makes the method massively parallel in nature and suitable for any type of parallel computing. The method is also inherently domain discretised as only the boundaries are shared among subdomains if the computational domain is divided into several subdomain for distributed parallel computing on cluster computers and complex super computers.

Minimum Calculation: With the FDTD method all the governing equations are solved on a rectangular (2D) or cuboid (3D) grid. So the grid is known and does not change anywhere in the computational domain. Yee's lattice ensures minimum computation for every partial derivative by making them one dimensional by placing the field components on the same direction on the lattice. Therefore, the total computation is minimum for each cell.

Material Models: The formulation of the FDTD method allows easy addition of many broadband dispersive material including dielectric, metal and semi-conduction models.

Disadvantages:

Staircasing: The FDTD requires an uniform Cartesian grid. Therefore, the structure needs to be drawn with rectangles (2D) or cuboids (3D). Otherwise, the interfaces of the device are approximated with staircased interfaces which introduces error in the results. One way to reduce the error is to increase the resolution of discretisation. Therefore, a cuboid structure not aligned with the coordinate axes will require much higher resolution to have acceptable approximate representation compared to a cuboid aligned

with the axes. The situation become worse when the interface is curved.

Single Grid Resolution: The FD grid used in the FDTD method requires only one resolution throughout the computational domain, Therefore, when a small part of the structure requires higher resolution, the entire domain is needed to be discretised with the highest resolution. Hence, the discretisation of the FDTD method might cause high overall inefficiency, despite having most optimal and fastest performance for each Yee's cell. Recently techniques have been developed to produce finer grid in a sub block of the computational domain [41]. But it requires the time step to satisfy the smallest element due to stability condition.

Multiple Interface: The Yee's lattice places different field components into different nodes in space. Therefore, at the interface of two different materials there are 6 different boundaries for each field component. For high resolution grid the difference between these boundaries may be very small compared to the structure. But at low resolution the difference become more significant. Therefore, higher resolution become a requirement for accurate simulation when using FDTD approach.

Numerical Dispersion: The Cartesian grid used in the method causes difference in speed of light depending in the direction (angle) of the propagation. At 45° the speed of propagation is the highest and it gradually decreases to the lowest at 0° and 90°. Thereby, causing artificial phase delay in propagation depending on the direction. This numerical dispersion is higher for lower resolution. Therefore, higher resolution is required for phase matching of the wave. The resolution of the simulation also needs to be increased with the increase in the length of the structure. The low numerical dispersion is crucial for analysing phase matching devices.

Finite Element Based Techniques

As mentioned in the above section, the problems associated with the FDTD methods mostly arise from the grid. Therefore, the general idea is, a better grid or meshing technique could

improve the performance of the grid and improve overall performance of the time domain analysis.

The Finite Element (FE)-based approaches are better alternatives for the effective representation of an arbitrary shaped structure, such as one with slanted or curved interfaces because it uses an unstructured polygonal mesh to represent the structure. The FEM was introduced to the electromagnetic analysis during the 1970's [42] to solve frequency domain problems [15, 28, 43]. To represent the structure more accurately, researchers have considered the FEM for time domain analysis [34–36, 39, 44, 45]. Although these methods are sometime more accurate in structural representation, however some of them may require an implicit solution of the computational domain for each time step [46], some require the solution of large matrices [36] and some require higher order solutions of Maxwell's equations [34, 39, 45].

However, among all the FE methods reported, the point-matched method [34] has features which may make it the more suitable. Firstly, it solves Maxwell's equations directly in the same manner as the FDTD. Secondly, it does not generate mass matrices and all calculations are local to each element. Lastly, the formulation is data parallel and suitable for parallel computing implementation. However, the downsides of this method presented in the work of Cangellaris [34] are the use of the rectangular grid. So, there is no significant advantage offered in the numerical dispersion.

1.4 Motivation for the Research

Although many attempts were made to develop an alternative to the FDTD proposed by Yee in 1966, none of the method mentioned above are as popular as the FDTD.

A time domain simulation takes a long time as the entire domain has to be solved for each time step. Increasing the computation for each time-step will cost a significant delay in completing the simulation. Moreover, for time domain simulation in 2D and 3D space, the computational domain has to be discretised using 2D and 3D mesh(s)/grid(s) and more than one instance of the all component values of each field have to be stored in the computer

memory to successfully model a propagating wave both in time and space. Therefore, any time domain analysis requires significantly higher memory and CPU time compared to other methods. Even a single addition operation could cause millions/billions (or even more) of CPU cycles if the analysis runs for thousands of time steps.

The FE discretisation could solve the inefficiency and the inaccuracy of the FD grid but in most cases adds too much calculation per element which is the main reason for its slow adoption. In simple words, the alternative FE proposed before this research are too slow and in many cases higher in memory requirement than the FDTD method.

Any successful alternative to the FDTD method should contain most of its advantages and also solve some of the issues associated with the FDTD method. That is, the method has to be explicit, data parallel, easily extendable and minimise computation at element level. Also it has to discretise the computational domain efficiently, will have the flexibility to change the shape of the mesh/grid to produce better approximation of the structure at lower resolution. This should allow dense and coarse mesh in the same computational domain depending on local necessity. It should have better numerical dispersion to allow low resolution simulation for bigger computational domain. As discussed in the previous section, all of these have never been put together in a single FE based method.

The goal of this research is to develop a method with most of the features mentioned above.

Part I

Two-dimensional Formulation

Chapter 2

Derivation of Governing Equations for Two-dimensions

Maxwell's equations in differential form were chosen for the derivation of the governing equations. Initially the method was developed for simpler two-dimensional simulation. In this case the mesh will consists of triangles and Maxwell's equations can be simplified for 2D approximation. Later a three-dimensional implementation will be shown to simulate real world devices. To simplify the derivation only non isotropic, lossless and non dispersive material is considered at this stage in charge free region.

Let us consider a region of space Ω occupied by a medium with position dependent parameters ε and μ, amenable to Maxwellś curl equations Eqs. 1.2c and 1.2d, All electromagnetic interactions inside Ω must fulfil the well-known boundary conditions $\hat{n} \times \mathbf{E} = 0$ and $\hat{n} \times \mathbf{H} = 0$ on perfect electric and magnetic surfaces Λ_E and Λ_H, respectively, with $\hat{n}$ as their normal unit vector. Moreover, in the case of an unbounded domain, outgoing waves should satisfy an appropriate radiation or boundary condition, whereas for the accomplishment of a unique solution, all fields in Ω at t = 0 have to be known. In this context, two weak forms can be obtained through the Galerkin or weighted residual method [36, 47]. A set of vector functions $\mathbf{u}$ can be employed as the weight function, which are actually square-integrable quantities with finite energy. Hence, for every $\mathbf{u}$, one gets,

$$\int_{\Omega} \mathbf{u} \cdot \left(\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} \right) d\Omega = 0 \tag{2.1a}$$

$$\int_{\Omega} \mathbf{u} \cdot \left(\nabla \times \mathbf{H} - \frac{\partial \mathbf{D}}{\partial t} \right) d\Omega = 0 \tag{2.1b}$$

Now for point matched method the weight function in Eqs. 2.1a and 2.1b can be replace by $\delta(t-t_i)$ and $\delta(t-t_j)$ respectively [48]. After performing the integration one can obtain,

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0 \tag{2.2a}$$

$$\nabla \times \mathbf{H} - \frac{\partial \mathbf{D}}{\partial t} = 0 \tag{2.2b}$$

Applying Eqs. 1.3a and 1.3b on Eqs. 2.2a and 2.2b respective yields,

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} \tag{2.3a}$$

$$\nabla \times \mathbf{H} = \varepsilon \frac{\partial \mathbf{E}}{\partial t} \tag{2.3b}$$

The partial differential operator ∇ is given by,

$$\nabla = \hat{x}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{2.4}$$

For the 2D formulation propagation in $x-y$ plane is considered. Therefore, wave can propagate any any direction on $x-y$ plane. z direction is considered to be uniform. As a result, $\partial/\partial z = 0$ The materials involved are also assumed to be isotropic and non dispersive.

Therefore, Eq. 2.3a can be written as,

$$\frac{d\mathbf{H}}{dt} = -\frac{1}{\mu}\begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ E_x & E_y & E_z \end{vmatrix} = -\frac{1}{\mu}\begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & 0 \\ E_x & E_y & E_z \end{vmatrix} \tag{2.5}$$

Eq. 2.3b can be written as,

$$\frac{d\mathbf{E}}{dt} = \frac{1}{\varepsilon}\begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & 0 \\ H_x & H_y & H_z \end{vmatrix} \tag{2.6}$$

From Eqs. 2.5 and 2.6, six equations (Eqs. 2.7 and Eqs. 2.8) can be obtained by equating the coefficients of the unit vectors $\hat{x}$, $\hat{y}$ and $\hat{z}$. These six equations can be divided into two different sets for propagation of Transverse Electric (TE) and Transverse Magnetic (TM) modes, respectively.

TE Propagation

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y} \tag{2.7a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} \tag{2.7b}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) \tag{2.7c}$$

TM Propagation

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y} \tag{2.8a}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x} \tag{2.8b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}\right) \tag{2.8c}$$

2.1 Discretisation

To solve the governing equations of Eqs. 2.7 and Eqs. 2.8, the computational domain has to be discretised. All the field components in these equations are functions of both space (x, y) and time (t). For all equations the left hand side of the equations calculate only the time evolution of the field and the right hand side of the equations calculate evolution in space separately. Therefore, the time evolution can be calculated with a time shape function (t is the variable of the function) at a fixed space node and the space evolution can be calculated at a fixed time node with a spatial shape function (x and y are the variable of the function) if an explicit approach time stepping is used.

2.1.1 Space Discretisation

To discretise the computational domain both in space and time, nodal elements can be used. Linear shape functions can be used to describe the variation of the field inside an element. The spatial variation of all the field components can be written in terms of the shape function as

$$\Phi = \sum_{i=1}^{M} N_i \phi_i \tag{2.9}$$

where Φ can be any field component (any one of H_x, H_y, H_z, E_x, E_y, E_z) inside the element, ϕ_i is the field component at the i^{th} node of the element (any one of h_x, h_y, h_z, e_x, e_y,

e_z), M is the number of nodes in an element. For a linear element (*i.e.* three node triangular elements) $M = 3$ and N_i is the shape function for the i^{th} node. Linear shape functions can be expressed as

$$N_i = a_i + b_i x + c_i y \tag{2.10}$$

where, a_i, b_i and c_i are the coefficients of the equation of plane going through the nodes of the element [1]. It should be mentioned here that, by using Eq. 2.9, shape function of any order can be incorporated with the proposed method. However, to store each higher order element more computer memory space would be required than a simpler linear element. Each linear element also takes least computation time. So, linear elements were chosen for the space discretisation.

2.1.2 Time Discretisation

Similarly, the field components along the time axis can be discretised as

$$\Psi = \sum_{j=1}^{P} Q_j \psi^{(j)} \tag{2.11}$$

where Ψ can be any field component (any one of the H_x, H_y, H_z, E_x, E_y, E_z) inside the element, $\psi^{(j)}$ is the field component at j^{th} time node, P is the number of the time node in the time element and for linear elements, $P = 2$. Here, Q_j is the shape function for the j^{th} time node and for the linear shape function it can be expressed as

$$Q_j = p_j t + q_j \tag{2.12}$$

where p_j and q_j are the coefficients of the line passing through the nodes of the time element. Similar to the space discretisation, Eq. 2.11 allows higher order time elements, if needed.

Both Eq. 2.9 and Eq. 2.11 can be applied to Eqs. 2.7 and Eqs. 2.8. For Example, Eq. 2.7a can be written as

$$\frac{d}{dt}\sum_{j=1}^{2} Q_j h_x^{(j)} = -\frac{1}{\mu}\frac{\partial}{\partial y}\sum_{i=1}^{3} N_i e_{zi}$$

$$\Rightarrow \sum_{j=1}^{2} p_j h_x^{(j)} = -\frac{1}{\mu}\sum_{i=1}^{3}\frac{\partial N_i}{\partial y} e_{zi}, \quad \because \frac{dQ_j}{dt} = p_j$$

$$\Rightarrow h_x^{(2)} = \frac{1}{p_2}\left[-\frac{1}{\mu}\sum_{i=1}^{3}\frac{\partial N_i}{\partial y} e_{zi} - p_1 h_x^{(1)}\right]$$

Similarly, all the equations from Eqs. 2.7 and Eqs. 2.8 can be derived. Discretised versions of Eqs. 2.7 and Eqs. 2.8 are given in Eqs. 2.13 and Eqs. 2.14, respectively.

For TE Propagation

$$h_x^{(n+1)} = \frac{1}{p_2}\left[-\frac{1}{\mu}\sum_{i=1}^{3}\frac{\partial N_i}{\partial y} e_{zi}^{(n)} - p_1 h_x^{(n-1)}\right] \tag{2.13a}$$

$$h_y^{(n+1)} = \frac{1}{p_2}\left[\frac{1}{\mu}\sum_{i=1}^{3}\frac{\partial N_i}{\partial x} e_{zi}^{(n)} - p_1 h_y^{(n-1)}\right] \tag{2.13b}$$

$$e_z^{(n+1)} = \frac{1}{p_2}\left[\frac{1}{\varepsilon}\left(\sum_{i=1}^{3}\frac{\partial N_i}{\partial x} h_{yi}^{(n)} - \sum_{i=1}^{3}\frac{\partial N_i}{\partial y} h_{xi}^{(n)}\right) - p_1 e_z^{(n-1)}\right] \tag{2.13c}$$

For TM Propagation

$$e_x^{(n+1)} = \frac{1}{p_2}\left[\frac{1}{\varepsilon}\sum_{i=1}^{3}\frac{\partial N_i}{\partial y} h_{zi}^{(n)} - p_1 e_x^{(n-1)}\right] \tag{2.14a}$$

$$e_y^{(n+1)} = \frac{1}{p_2}\left[-\frac{1}{\varepsilon}\sum_{i=1}^{3}\frac{\partial N_i}{\partial x} h_{zi}^{(n)} - p_1 e_y^{(n-1)}\right] \tag{2.14b}$$

$$h_z^{(n+1)} = \frac{1}{p_2}\left[-\frac{1}{\mu}\left(\sum_{i=1}^{3}\frac{\partial N_i}{\partial y} e_{yi}^{(n)} - \sum_{i=1}^{3}\frac{\partial N_i}{\partial y} e_{xi}^{(n)}\right) - p_1 h_z^{(n-1)}\right] \tag{2.14c}$$

where the field components with the $(n+1)$, (n) and $(n-1)$ superscripts are the future, current and the past values, respectively.

These two sets of equations are the main governing equations of the 2D FETD. It should be noted that, although both Eqs. 2.13 and 2.14 are independent of each other, the equations

inside each set are not independent. Therefore, no single equations can produce time evolution of light without considering other equations in its set. Therefore, the equations in both sets are coupled and should be solved in a coupled manner.

Chapter 3

The Two-dimensional Mesh

The mesh is the most important part of all the FE-based methods. It allows discretisation of an irregular shaped structure in a more accurate and efficient manner. The speed of an FE-based code largely depends on how efficiently the mesh discretises the computational domain. Hence efficient meshing which reduces computational cells without sacrificing accuracy of the solution, is one of the key factors used to make a fast and efficient FE-based code.

The research on finite element mesh generation was formally started perhaps as early as the beginning of the 1970s [49], and a comprehensive review of the finite element mesh generation schemes developed before 1980 was presented by Thacker [50]. In line with the advance of the finite element method, the irregular computational grid became increasingly popular for two reasons:

1. they allow points to be situated on curved boundaries of irregularly shaped domains
2. they allow points to be distributed at the interior of the domain with variable nodal spacing

Coordinate transformation was an early attempt to map a regular reference domain onto a geometrically irregular computational physical domain with a possibility of smooth transition in element size [51].

Finite element interpolation as a means of mesh generation was presented in [52] in which a curved domain is represented by a super-element, which could be further divided into smaller elements following the element reference coordinates. The blending function interpolation developed for local refinements to minimise the energy of the system is related to the r-refinement procedure that we are using today.

Unstructured mesh generation thrived in the early 1980s mainly driven by the development of the three popular unstructured mesh generation schemes, namely, the Delaunay triangulation, AFT and Octree decomposition.

The theoretical basis of Delaunay triangulation was established a long time ago by Dirichlet [53], Voronoi [54] and Delaunay [55], and an ef cient and robust construction algorithm by point insertion was only developed in 1981 by Bowyer and Watson [56, 57]. Delaunay triangulation will only give the convex hull of the given point set, and for finite element mesh generation, geometrical and topological constraints on the boundary have to be enforced. However, the technique was first employed formally for 2D and 3D finite element mesh generation in 1970s [58, 59].

The AFT method divides the domain into two parts; meshed part and unmeshed part. There is a moving boundary between these parts and the technique focuses on the unmeshed part only and uses arbitrary shapes to generate elements to fill the unmeshed part. It was introduced in 1980s [60].

Octree decomposition was introduced in 1980s [61, 62]. The method considers boundary characteristics and nodal requirement first and generate mesh using *"marching cube"* method by projecting points to boundary and proper connection of points to form hexahedral and tetrahedral elements.

Although unstructured mesh generation became very advance and efficient, this thesis will only use uniform meshes. Primarily because it is much simpler to quantify and compare the performance of the method against its Finite Difference alternative, the FDTD.

3.1 The Space Mesh System

To explain the meshing scheme in a simpler manner, a uniform square grid was chosen to start with. This allows a simpler explanation of the meshing system. To use the linear shape functions, triangles with three nodes were considered to discretise the computational domain. This mesh can be termed the **"Main Mesh"**. For TE propagation, it maybe assumed that, all current e_z field components are stored at the nodes of the triangular mesh. Both h_x and h_y field components can be calculated from e_z using Eq. 2.13a and Eq. 2.13b. Equation 2.13a calculates one future value of h_x using the current values of the e_z field components at the nodes of the triangular element. As there is only one value for the whole element, it cannot be stored in any specific node of the triangle, but instead it can be stored at the centroid of the triangle, which is unique. As a result, no value of h_x field component will be available at the corner nodes of the elements of main mesh. Similarly, for Eq. 2.13b the calculated future value of h_y can be stored at the centroid. To obtain the next values of e_z from Eq. 2.13c, the current value of h_x and h_y are required which are placed at the centroids of the elements of main mesh. Hence, another triangular mesh is required which will be constructed using the centroids of the main mesh elements. This new mesh can be termed the **"Auxiliary Mesh"**. It should be mentioned here that the Voronoi mesh could be used as the auxiliary mesh. This is because the formulation supports higher order elements. But as it has been mentioned before that linear elements require less computational resources then higher order elements, the linear auxiliary mesh proposed in this section will be used throughout this thesis. As the h_x and h_y field components are stored at the auxiliary mesh, the elements of this mesh can be used to calculate the future e_z values which can be stored in the nodes of the main mesh, provided each element of the auxiliary mesh must surround one of the nodes of the main mesh. Similar arrangements can be made for TM propagation using Eqs. 2.14.

The method will allow any shape of elements. For simplicity and ease, a simple square grid was considered to illustrate the meshing process. This grid can be converted into a triangular mesh by dividing each cell with one diagonal line. Figure 3.1(a) shows a triangle mesh generated from a 4×4 square grid. As can be seen the resultant mesh is a **"Isosceles**

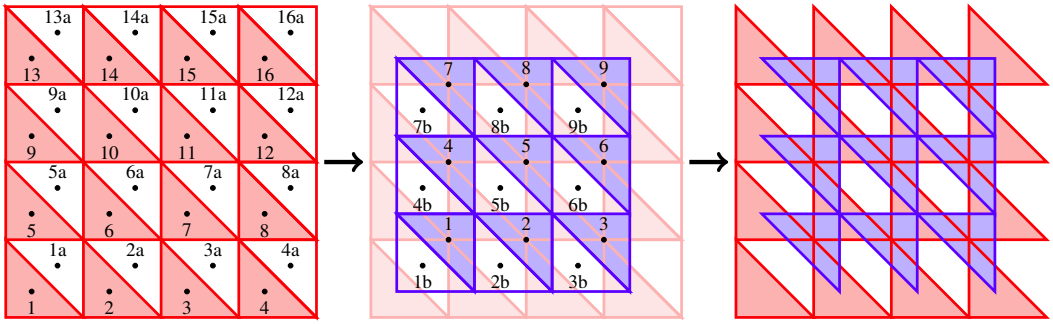


Fig. 3.1 (a) The generating the linear mesh by dividing a square grid by diagonal line, (b) Generating the auxiliary mesh by connecting the centroids of the main mesh, (c) Discarding the unwanted elements from both meshes

Right-angled Triangle (IRT) Mesh". The lower triangle of the square cell has been shaded in pink and the upper triangle is in white. The lower triangle is numbered with the cell index and the upper triangle is numbered with the cell index and additionally with a suffix "a". The black dot inside each triangle is the centroid of that triangle.

The auxiliary mesh has to be generated using the centroids of the main mesh. Each element of the auxiliary mesh has to surround one of the nodes of main mesh. The **Perfect Electric Conductor (PEC)** boundary condition will be applied at the boundaries of the main mesh to truncate the computational domain into a finite one. Therefore, calculation of field components of the boundary nodes of main mesh is not necessary. Centroids of the elements 2, 5 and 6 of the main mesh (Fig. 3.1(a)) are taken as the three nodes of the element number 1 of the auxiliary mesh (shaded in light blue) and the centroids of elements 1, 2 and 5 are taken for element 1b of the auxiliary mesh (white coloured). Similarly, all the other elements were generated from the centroids of the main mesh. Figure 3.1(b) shows the auxiliary mesh constructed from the centroids of the main mesh shown in Fig. 3.1(a). The centroids of the auxiliary mesh are shown as black dots in Fig 3.1(b).

It can be seen that the centroids of elements 1, 2, 3, $\cdots$, 9 of the auxiliary mesh (Fig. 3.1(b)) coincide with the node points of the main mesh, whereas the centroids of 1b, 2b, 3b, $\cdots$, 9b coincide with the centroids of 1a, 2a, 3a, $\cdots$, 11a of the main mesh. As a result, these elements cannot be used to calculate the field components at the nodes of the other mesh. Therefore, the interpolation functions (shape functions) of the elements 1a, 2a,

3a, $\cdots$ of the main mesh and 1b, 2b, 3b $\cdots$ of the auxiliary mesh will not be used in calculation of the governing equations. This is a requirement of the technique when used with a triangle mesh. By not using the shape functions of the unwanted elements, the number of elements in Fig. 3.1(c) will become half of Fig. 3.1(b). [63].

This is a unique mesh system introduced with the proposed method. This mesh system has a big advantage. It reduces the number of computational elements to less than half of the methods using all shape functions of the mesh. As a result, the method proposed in this paper is twice faster than any other FEM method using full mesh with the same computational need per element. The memory requirement for the method will be less than an FEM approach using the full mesh discretisation.

3.1.1 Completeness of the Mesh

It should be noted that at the beginning the discretisation of the computational domain was performed with two full meshes. Which covers the entire computational domain with piecewise interpolation functions called shape functions. Therefore, the mesh discretising the space is fully described by the interpolation functions. As the shape functions considered for the discretisation is not exactly the function of the field distribution, the approximation of the computational domain produces some residual error which can be minimised by reducing the size of the discretisation elements. Mathematically this can be expressed using completeness theorem [64],

$$R_m = \int_{\Omega} \left(F(x,y) - \sum_{k=1}^{m} \Phi_k(x,y) \right)^2 \dot{\mathrm{A}} \tag{3.1}$$

Here, $F(x,y)$ is the complete space and $\Phi_k(x,y)$ is the shape function (Eq. 2.9) and R_m is the residual error which converges to 0 with sufficiently large m (number of elements).

Due to the requirement of the governing equations the method uses less than half of the interpolation functions of one mesh to calculate the future field component in the other mesh. Proposed method also updates all nodes of both meshes. Therefore, the computational space discretised by the 2 meshes can be full explained by the shape functions of

elements used to describe the domain in the first step. Hence the method provide a complete solution.

As seen in Fig. 3.1(c), the main mesh elements surround their associated nodes on the auxiliary mesh and are used to update the value of the fields associated with the node and vice versa. Although the alternative elements (elements with suffix 'a' and 'b') have been removed from the mesh system, however, all of the three nodes associated with those triangle are still updated during the calculation of every time step. Therefore, the magnitude of the field components inside the skipped region can be calculated by using the space shape functions of that triangle (Eq. 2.10). The affect of neighbouring elements are transferred when calculating the field at a node on mesh 'a' using the element on mesh 'b'. Also the method does not rely on any mass matrix. As a result, the 2 staged solution presents a full solution of the space even though calculations are done on 2 meshes.

As this meshing is a new concept, present day meshing libraries may not be optimised for this type of coupled meshing.

3.2 The Time Mesh System

Along with the space domain, the time domain also needs to be discretised. This can be achieved by using linear elements with two nodes. In Eq. 2.13a and Eq. 2.13b, the future values of the h_x and the h_y field components may be calculated from the current e_z field components and the past h_x and h_y components respectively. Equation 2.13c calculates the future values of the e_z field components with the current h_x, h_y and the past e_z field components. Similar explanations can be given for Eqs. 2.14. For all the equations, the current values of the **E** field components with the past values of the **H** field components are needed to calculate the future values of **H** components and vice versa. Therefore, both the **E** and **H** field components cannot be calculated at the same time node.

For the TE propagation example given in Section 3.1, the simulation started with the calculation of the future h_x and h_y fields from the current e_z field components in the main mesh (with Eq. 2.13a and Eq. 2.13b). Therefore, the first time node belongs to the e_z component

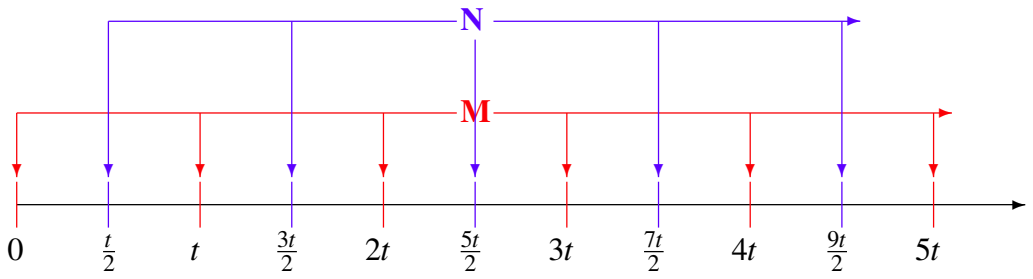


Fig. 3.2 Arrangement of time mesh system for equal time spacing

associated with the main mesh. The future e_z field was calculated from the h_x and h_y fields in the auxiliary mesh. So, the second time node belongs to the h_x and h_y field components associated with the auxiliary mesh. This way the time domain can be divided into two time meshes **M** and **N** which are associated with the field components of the main mesh and the auxiliary mesh respectively. Figure 3.2 shows the **M** and **N** meshes. In this example, the time step size for the calculation of the future e_z components from the previous e_z components is t. So, the e_z field was calculated at t, $2t$, $3t$, $4t$, $\cdots$ and hence these time nodes belong to mesh **M** along with the initial e_z at time 0. The time step size for the calculation of the h_x and h_y field components have to be of the same duration, t. To calculate the e_z components, the current h_x and h_y components are required. Therefore, the h_x and h_y field components were calculated at $t/2$, $3t/2$, $5t/2$, $7t/2$, $\cdots$. Thus, these time nodes belong to mesh **N**. Similar examples can be shown for TM propagation with Eqs. 2.14.

Chapter 4

Perfectly Matched Layer Boundary for Two-dimensions

As an EM wave approaches the boundary of the finite computational domain, a new type of problem starts to emerge. The boundary condition of the computational domain can be a perfect electric conductor (PEC) or a perfect magnetic conductor (PMC), depending on the formulation used. In both cases, the field incident on the boundary will reflect back into the computational domain, like open or short circuit termination and interference of the forward propagating and reflected waves would corrupt the results.

To avoid unwanted reflection from the computational boundary, two approaches can be under taken. Which are

1. Large computational domain
2. Absorbing boundary condition

4.1 Large Computational Domain

If a large enough computational domain is considered and the simulation is carried out for a limited number of steps, such that the field does not reach the domain boundaries and the

unphysical reflection can be avoided.

However, this would requires a large number of nodes to spread around the computational domain. As a result, the computational resource required would be huge compared to the problem area with a smaller computational domain. Thus, this is not a practically viable solution for the problem for most of the occasions.

4.2 Absorbing Boundary Condition

Another way to avoid reflections is to apply a suitable absorbing boundary condition/material that may absorb the wave before it hits the computational boundary without any reflection. To avoid reflection, the material should be impedance matched with the nearby medium.

There are different ways to absorb the outgoing wave. Among them only all different ways followings two ways are prominent,

1. those based on one-way wave equations
2. those based on surrounding the domain with a layer of absorbing material

4.2.1 Methods Based on Oneway Wave Equations

The first category of ABCs relies on the fact that the solution of Maxwell?s two coupled curl equations is equivalent to the solution of the second-order wave equation for any one of the field components. Although the wave equation naturally supports waves propagating in both forward and backward directions, it can be factored into two oneway wave equations, each of which supports waves in only one direction. This property provides the basis for an algorithmic method by which the fields can be “propagated out” of the domain minimising reflections back into the numerical space. A first-order scheme of this type is first discussed below, using the 1D wave equation, for simple one-way wave equations. This scheme is known as the first-order Mur boundary condition, and is quite effective in the removal of the

plane wave fields normally incident on an FDTD boundary, being particularly suitable for 1D problems.

First-order Mur boundary

The first-order Mur boundary condition is one of the simplest boundary conditions available, but it can be very effective for 1D simulations, as it relies on normal incidence of the wave on the boundary. This Mur ABC is based on the one-way wave equations [65].

$$\frac{\partial E_z}{\partial t} + v_p \frac{\partial E_z}{\partial x} = 0 \qquad \rightarrow E_z(x,t) = f(x - v_p t) \tag{4.1a}$$

$$\frac{\partial E_z}{\partial t} - v_p \frac{\partial E_z}{\partial x} = 0 \qquad \rightarrow E_z(x,t) = f(x + v_p t) \tag{4.1b}$$

The solutions of Equ. 4.1a and Equ. 4.1b are waves propagating respectively in the positive (+x) and negative (-x) directions. These equations can respectively be used to simulate open-region boundaries at the right and left sides of the 1D domain.

As the method relies on the direction of propagation, it is more difficult to implement and use in 2D and 3D as the wave can propagate in any direction [66].

Other mentionable ABC using oneway wave equation are Bayliss-Turkel operators [67] and Higdon operators [68].

4.2.2 Perfectly Matched Layer Boundary

The other type of boundary conditions surrounds the computational domain with absorbing medium and absorbs the outgoing wave it approaches boundary. The biggest advantage of this technique is that it does not assume any angel of incident and thus support absorption of wave from any direction.

Berenger, in 1994, where he proposed a boundary material for rectangular computational domain which would theoretically absorb all the incoming EM waves without reflec-

tion [69].

The PML can be implemented in several ways. For example, by split field, convolutional, uniaxial PML etc.

Split Field PML

In this type of PML the transverse fields are split into two directional components for 2D implementation. *i.e.* H_z field can be split into two directional components H_{zx} and H_{zy} in such a way that $H_z = H_{zx} + H_{zy}$. The 2D TM mode governing equations in Eqs. 2.8 for split field PML can be written as,

$$\frac{dE_x}{dt} + \sigma_y E_x = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y} \tag{4.2a}$$

$$\frac{dE_y}{dt} + \sigma_x E_y = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x} \tag{4.2b}$$

$$\frac{dH_{zx}}{dt} + \sigma_{m,x} H_{zx} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x}\right) \tag{4.2c}$$

$$\frac{dH_{zy}}{dt} + \sigma_{m,y} H_{zy} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y}\right) \tag{4.2d}$$

Here, σ_x, σ_y are the directional electric conductivity and $\sigma_{m,x}$, $\sigma_{m,y}$ are the magnetic conductivity.

Complex Frequency Shifted PML/Convolutional PML

This PML was introduced by Kuzuoglu and Mittra [70] to absorb the evanescent generated near the radiating sources. This is done by replacing the term $j\omega\varepsilon$ with $\alpha + j\omega\varepsilon$. This allows the PML to absorb the evanescent field.

The best implementation of this PML was proposoed by Roden and Gedney in 2000 [71] using a recursive convolution technique. Their technique is commonly referred to as the convolutional PML or CPML.

For this PML a more general stretched coordinate is introduced:

$$s_i = \left(\kappa_i + \frac{\sigma_i}{\alpha_i + j\omega\varepsilon_0} \right) \tag{4.3}$$

Now Equ. 4.2 can be written as,

$$\frac{dE_x}{dt} = s_y * \frac{1}{\varepsilon} \frac{\partial H_z}{\partial y} \tag{4.4a}$$

$$\frac{dE_y}{dt} = -s_x * \frac{1}{\varepsilon} \frac{\partial H_z}{\partial x} \tag{4.4b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu} \left(s_x * \frac{\partial E_y}{\partial x} - s_y * \frac{\partial E_x}{\partial y} \right) \tag{4.4c}$$

Uniaxial PML

This is an alternative way of implementing the PML which is simpler then the split field PML or the convolutional PML. Because, it does not require any complex calculation or splitting of field components. To perform some simulation with the proposed method ABC is necessary. For this purpose the uniaxial PML implementation was chosen because it can be implemented with only real numbers and it has wide band nature.

The implementation of various types of PML for Cartesian coordinate system can be classified into three different types, *i.e.*

1. X axis PML
2. Y axis PML
3. Corner PML

To implement any kind of PML the partial differential operator ∇ (Eq. 2.4) has to be modified [69].

4.3 X Axis PML

The X axis PML absorbs any wave that moves in the X direction and towards the boundary. The original partial differential operator is modified as follows [69],

$$\tilde{\nabla}^{(x)} = \hat{x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{4.5}$$

where σ_x is a function of x and ω is the angular frequency.

Thus the affected equations will be Eqs. 2.7b, 2.7c, 2.8b and 2.8c.

If $\tilde{\nabla}^{(x)}$ is used instead of ∇ in Eqs. 2.3, then Eq. 2.7b,

$$\begin{aligned}
&\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu}\left(\tilde{\nabla}^{(x)} \times \mathbf{E}\right) \\
\Rightarrow &\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu}\begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ H_x & H_y & H_z \end{vmatrix} \\
\Rightarrow &\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu}\begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1} & \frac{\partial}{\partial y} & 0 \\ H_x & H_y & H_z \end{vmatrix} \quad \text{(for 2D, similar to Equ. 2.5)}
\end{aligned} \tag{4.6}$$

Now, by equating the coefficient of $\hat{y}$ from Equ. 4.6,

$$\begin{aligned}
&\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1} \\
\Rightarrow &\left(1 + \frac{\sigma_x}{j\omega}\right)\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} \\
\Rightarrow &\frac{\partial H_y}{\partial t} + \sigma_x H_y = \frac{1}{\mu}\frac{\partial E_z}{\partial x} \quad \text{considering } j\omega = \frac{\partial}{\partial t} \\
\Rightarrow &\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} - \sigma_x H_y
\end{aligned} \tag{4.7}$$

Therefore, Equ. 2.7b is replaced by Equ. 4.7.

The implementation of the $1/\omega$ dependence in the time domain can be done using the auxiliary differential equations approach (ADE) [66]. which introduces auxiliary fields.

Eq. 2.7c will be replaced by following three equations,

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \Psi_{x[x]}\right) - \sigma_x E_z \tag{4.8a}$$

$$\frac{\partial \Psi_{x[x]}}{\partial t} = \sigma_x \frac{\partial H_x}{\partial y} \tag{4.8b}$$

Here, $\Psi_{x[x]}$ is the auxiliary field generated due to the use of X PML.

Eq. 2.8b will be replaced by following equation,

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x} - \sigma_x E_y \tag{4.9a}$$

And Eq. 2.8c will be replaced with,

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} - \Phi_{x[x]}\right) - \sigma_x H_z \tag{4.10a}$$

$$\frac{\partial \Phi_{x[x]}}{\partial t} = \sigma_x \frac{\partial E_x}{\partial y} \tag{4.10b}$$

Here, $\Phi_{x[x]}$ is the auxiliary field generated due to the use of X PML.

4.4 Y Axis PML

Similarly, for the Y axis PML, the differential operator is modified as follows,

$$\tilde{\nabla}^{(y)} = \hat{x}\frac{\partial}{\partial x} + \hat{y}\left(1 - j\frac{\sigma_y}{\omega}\right)^{-1}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{4.11}$$

As a result, Eq. 2.7a will be replaced with,

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y} - \sigma_y H_x \tag{4.12a}$$

Eq. 2.7c will become,

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} + \Psi_{y[y]}\right) - \sigma_y E_z \tag{4.13a}$$

$$\frac{\partial \Psi_{y[y]}}{\partial t} = \sigma_y \frac{\partial H_y}{\partial x} \tag{4.13b}$$

Here, $\Psi_{y[y]}$ is the auxiliary field generated due to the use of Y PML.

Eq. 2.8a can be replaced with,

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y} - \sigma_y E_x \tag{4.14a}$$

and finally Eq. 2.8c will become,

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + \Phi_{y[y]}\right) - \sigma_y H_z \tag{4.15a}$$

$$\frac{\partial \Phi_{y[y]}}{\partial t} = \sigma_y \frac{\partial E_y}{\partial x} \tag{4.15b}$$

Here, $\Phi_{y[y]}$ is the auxiliary field generated due to the use of Y PML.

4.5 Corner PML

This type of PML is used only at the corners of the computational domain. The reason is that it can absorb waves in both directions and due to the discrete implementation, the numerical reflection (the reflection due to the discrete implementation) from the PML would

be much higher with the change in each step towards the boundaries.

As was mentioned above, this PML absorbs electromagnetic waves in both transverse directions. The partial differential operator of Eq. 2.4 is replaced with the following modified operator given in the following equation,

$$\tilde{\nabla}^{(xy)} = \hat{x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1}\frac{\partial}{\partial x} + \hat{y}\left(1 - j\frac{\sigma_y}{\omega}\right)^{-1}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{4.16}$$

So, Eq. 2.7a will be replaced with,

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y} - \sigma_y H_x \tag{4.17a}$$

Eq. 2.7b will be replaced with,

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} - \sigma_x H_y \tag{4.18a}$$

Eq. 2.7c will be replaced with,

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} + \Psi_{y[y]} - \Psi_{x[x]}\right) - \sigma_x E_z - \sigma_y E_z - \Theta_{z[xy]} \tag{4.19a}$$

$$\frac{\partial \Psi_{x[x]}}{\partial t} = \sigma_x \frac{\partial H_x}{\partial y} \tag{4.19b}$$

$$\frac{\partial \Psi_{y[y]}}{\partial t} = \sigma_y \frac{\partial H_y}{\partial x} \tag{4.19c}$$

$$\frac{\partial \Theta_{z[xy]}}{\partial t} = \sigma_x \sigma_y E_z \tag{4.19d}$$

Here, $\Psi_{x[x]}$, $\Psi_{y[y]}$ and $\Theta_{z[xy]}$ are the auxiliary fields generated due to the use of Corner PML.

Eq. 2.8a will be replaced with,

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y} - \sigma_y E_x \tag{4.20a}$$

Eq. 2.8b will be replaced with,

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x} - \sigma_x E_y \tag{4.21a}$$

Eq. 2.8c will be replaced with,

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + \Phi_{y[y]} - \Phi_{x[x]}\right) - \sigma_x H_z - \sigma_y H_z - \Omega_{z[xy]} \tag{4.22a}$$

$$\frac{\partial \Phi_{x[x]}}{\partial t} = \sigma_x \frac{\partial E_x}{\partial y} \tag{4.22b}$$

$$\frac{\partial \Phi_{y[y]}}{\partial t} = \sigma_y \frac{\partial E_y}{\partial x} \tag{4.22c}$$

$$\frac{\partial \Omega_{z[xy]}}{\partial t} = \sigma_x \sigma_y H_z \tag{4.22d}$$

Here, $\Phi_{x[x]}$, $\Phi_{y[y]}$ and $\Omega_{z[xy]}$ are the auxiliary fields generated due to the use of Corner PML.

4.6 Placing the PML in the Computational Domain

The PML is a non physical medium and as a result, improper placement in the computational domain can make the solution unstable. As reported in [69], the PML must be placed around the boundary, as shown in Fig. 4.1.

As can be seen, the X Axis PMLs should be placed on the left and right hand sides of the computational domain such that the value of σ_x increases with the increase in distance from the boundary along the x axis inside the PML.

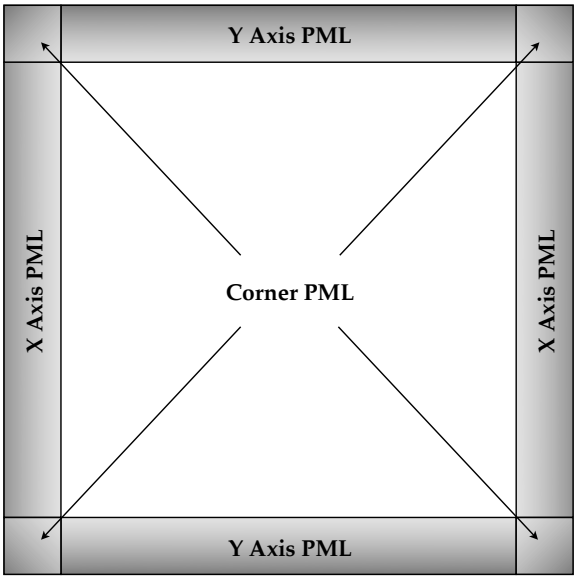


Fig. 4.1 Placement of different type of PML in computational domain

Similarly, the Y PML should be placed at top and bottom of the computational domain and σ_y follows the same profile as σ_x. However, this time the distance is measured along y axis.

The corner PML has to be placed at the four corners of the computational domain. The profile σ_x and σ_y is similar to the σ_x and σ_y used in the X PML and Y PML respectively.

Chapter 5

Dispersive Materials

Although the initial derivation of the proposed method in in Chapter 2 only considers isotropic and non dispersive material, the capability of the method presented in Chapters 2 and 3 are far beyond it. The method is equally good at simulating dispersive materials such as metals.

Metals: Metals are highly dispersive materials in the optical range. The dispersive characteristics of the permittivity of a metal can be modelled by the Drude model [72].

Metamaterials: Metalmaterials are man made materials designed to produce characteristics which are not found in natural materials. Metamaterials are often characterised in terms of their effective material parameters, such as electric permittivity and magnetic permeability. These constituent parameters can either be both negative, or only one of them may be negative, while the other is positive. The former is often referred to as LHM, DNG, or negative refractive index material (NRIM) [73–76]. The latter is called single negative material (SNG).

The concept of LHMs was first theorised by the Russian physicist Veselago in 1968 [73]. In this paper, Veselago speculated on the possible existence of LHMs and anticipated their unique electromagnetic properties such as the reversal of Snell's law, the Doppler effect, and the Vavilov Cherenkov effect. Veselago showed that the electric field, magnetic field, and

wave vector of an electromagnetic wave in an LHM form an LH triad. As a result, LHMs support electromagnetic waves with group velocity and phase velocities that are antiparallel, known as backward waves. Consequently, while the energy still travels away from the source, so as to satisfy causality, wavefronts travel backward toward the source in an LHM, a phenomenon that is associated with negative refractive index of refraction.

Every material is a composite in some sense, even if the individual ingredients consist of atoms and molecules. For periodic structures, defined by a unit cell whose characteristic dimension is a, the following criterion must be satisfied in order for the structure to be viewed as a homogeneous medium:

$$a \ll \lambda = \frac{2\pi c}{\omega} \tag{5.1}$$

Here, λ is the wavelength, c is speed of light and ω is the angular frequency of the light.

Should the above condition be violated, the possibility would exist that the internal structure of the medium would diffract as well as refract radiation, and thus invalidate the homogeneous medium assumption [75].

Although Veselago presented the idea of DNG material in 1968, the research in this area was largely discontinued due to the absence of naturally occurring materials with negative μ. New discoveries of LH media was not made until recently, when a composite medium was demonstrated in which, both the effective ε and μ were purportedly shown to be simultaneously less than zero [74], over a finite frequency band.

The composite medium used in [74] made use of an array of metallic posts to create a frequency region with $\varepsilon_{eff} < 0$, interspersed with an array of split-ring resonators (SRR) for which $\mu_{eff} < 0$ was supposed in the frequency range of interest. The SRR and wire medium, both revisited by Pendry [75], have been extensively studied by a number of researchers.

Due to the dispersive nature of the DNG metamaterial characteristics both the permeability and the permittivity can be modelled using the Drude dispersion model [72].

One way to show the extensibility of the proposed method is by incorporating Drude mode and perform some benchmark examples using the extended method to prove its efficacy.

To extend the capability of the method to highly dispersive material like metal and metamaterial the Drude dispersion model [72] can be incorporated with the governing equations of the method.

5.1 Maxwell's Equations with Drude Model

The Drude model for permeability and permittivity can be written as follows,

$$\varepsilon(\omega) = \varepsilon_0 \left(1 - \frac{\omega_{pe}^2}{\omega(\omega + j\gamma_e)}\right) \tag{5.2a}$$

$$\mu(\omega) = \mu_0 \left(1 - \frac{\omega_{pm}^2}{\omega(\omega + j\gamma_m)}\right) \tag{5.2b}$$

where, ω_{pe} and ω_{pm} are the electric and magnetic plasma frequencies and γ_e and γ_m are the electric and magnetic collision frequencies respectively [72].

Replacing ε of Eq. 1.3b with the new dispersive equation from Eq. 5.2a,

$$\mathbf{D} = \varepsilon(\omega)\mathbf{E} = \varepsilon_0 \left(1 - \frac{\omega_{pe}^2}{\omega(\omega + j\gamma_e)}\right)\mathbf{E} \tag{5.3}$$

By taking the time derivative of Eq. 5.3 can be divided into two partial differential equations as follows,

$$\frac{\partial \mathbf{D}}{\partial t} = \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mathbf{M}_e \tag{5.4a}$$

$$\frac{\partial \mathbf{M}_e}{\partial t} = \varepsilon_0 \omega_{pe}^2 \mathbf{E} - \gamma_e \mathbf{M}_e \tag{5.4b}$$

where, $-j\omega = \frac{\partial}{\partial t}$ and $\mathbf{M}_e = -\frac{\varepsilon_0 \omega_{pe}^2}{j\omega - \gamma_e}\mathbf{E}$.

$\frac{\partial \mathbf{D}}{\partial t}$ of Eq. 2.3b can be replaced with the dispersive version from Eq. 5.4a and by taking

$\mathbf{J} = 0$ for source free region,

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\varepsilon_0}(\nabla \times \mathbf{H} - \mathbf{M}_e) \tag{5.5}$$

Eqs. 5.5 and 5.4b can be used to calculate time variation of the **E** field within metals and DNG materials. Here, $\mathbf{M}_e$ is an auxiliary field generated to incorporate the effect of the Drude dispersion into the governing equation. The value of $\mathbf{M}_e$ at a given time and space can be calculated using Eq. 5.4b.

Similarly, a set of equations can be derived for the **H** field calculation by using Eq. 5.2b as the dispersion model for $\mu(\omega)$ in Eq. 2.3a.

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu_0}(\nabla \times \mathbf{E} + \mathbf{M}_m) \tag{5.6a}$$

$$\frac{\partial \mathbf{M}_m}{\partial t} = \mu_0 \omega_{pm}^2 \mathbf{H} - \gamma_m \mathbf{M}_m \tag{5.6b}$$

where, $\mathbf{M}_m$ is an auxiliary field.

5.2 Governing Equations for Metal

For metals, the Drude dispersion model only applies for the permittivity. Therefore, Eqs. 2.3a and 5.5 along with Eq. 5.4b can be used to generate the governing equations for propagation in metals.

TE Propagation

When the $E_z \neq 0$ and the plane of propagation is $x-y$ the $E_x = E_y = 0$. The governing equations are,

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y} \tag{5.7a}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} \tag{5.7b}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon_0}\left(\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) - M_{e_z}\right) \tag{5.7c}$$

$$\frac{\partial M_{e_z}}{\partial t} = \varepsilon_0 \omega_{pe}^2 E_z - \gamma_e M_{e_z} \tag{5.7d}$$

TM Propagation

When the $H_z \neq 0$ and the plane of propagation is $x-y$ the $H_x = H_y = 0$. The governing equations are,

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon_0}\left(\frac{\partial H_z}{\partial y} - M_{e_x}\right) \tag{5.8a}$$

$$\frac{\partial M_{e_x}}{\partial t} = \varepsilon_0 \omega_{pe}^2 E_x - \gamma_e M_{e_x} \tag{5.8b}$$

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon_0}\left(\frac{\partial H_z}{\partial x} + M_{e_y}\right) \tag{5.8c}$$

$$\frac{\partial M_{e_y}}{\partial t} = \varepsilon_0 \omega_{pe}^2 E_y - \gamma_e M_{e_y} \tag{5.8d}$$

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}\right) \tag{5.8e}$$

5.3 Governing Equations for Metamaterial

For metal material Drude dispersion model applies on both Eqs. 2.3a and 2.3b. Therefore, to derive the governing equations for metamaterials, Eqs. 5.5 and 5.6a should be utilised along with Eqs. 5.4b and 5.6b. By equating the coefficients of $\hat{x}$, $\hat{y}$ and $\hat{z}$, two sets of coupled equations for the TE and TM modes of propagation can be derived.

TE Propagation

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu_0}\left(\frac{\partial E_z}{\partial y} + M_{m_x}\right) \tag{5.9a}$$

$$\frac{\partial M_{m_x}}{\partial t} = \mu_0 \omega_{pm}^2 H_x - \gamma_m M_{m_x} \tag{5.9b}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu_0}\left(\frac{\partial E_z}{\partial x} - M_{m_y}\right) \tag{5.9c}$$

$$\frac{\partial M_{m_y}}{\partial t} = \mu_0 \omega_{pm}^2 H_y - \gamma_m M_{m_y} \tag{5.9d}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon_0}\left(\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) - M_{e_z}\right) \tag{5.9e}$$

$$\frac{\partial M_{e_z}}{\partial t} = \varepsilon_0 \omega_{pe}^2 E_z - \gamma_e M_{e_z} \tag{5.9f}$$

TM Propagation

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon_0}\left(\frac{\partial H_z}{\partial y} - M_{e_x}\right) \tag{5.10a}$$

$$\frac{\partial M_{e_x}}{\partial t} = \varepsilon_0 \omega_{pe}^2 E_x - \gamma_e M_{e_x} \tag{5.10b}$$

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon_0}\left(\frac{\partial H_z}{\partial x} + M_{e_y}\right) \tag{5.10c}$$

$$\frac{\partial M_{e_y}}{\partial t} = \varepsilon_0 \omega_{pe}^2 E_y - \gamma_e M_{e_y} \tag{5.10d}$$

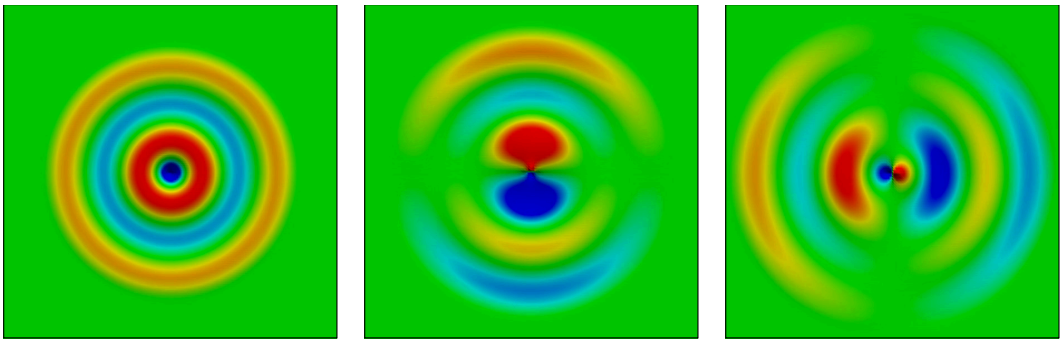
$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu_0}\left(\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}\right) + M_{m_z}\right) \tag{5.10e}$$

$$\frac{\partial M_{m_z}}{\partial t} = \mu_0 \omega_{pm}^2 H_z - \gamma_m M_{m_z} \tag{5.10f}$$

Chapter 6

Results of Simulations in Two-dimensions

A C++ code was developed to perform the numerical simulations. To make the code multi-threaded OpenMP technology was used. The implementation was made dimensionless or scale invariant [24] by taking the speed of light as $c = 1$. As a result permeability and permittivity of vacuum $\mu_0 = 1$ and $\varepsilon_0 = 1$, respectively. This made the implementation dimensionless, scalable and for many problem reduces the effect of floating point errors. The outputs of the program were stored in the VTK file format to visualize with Paraview software.



(a) (b) (c)

Fig. 6.1 (a) E_z field in free space, (b) H_x field in free space, (c) H_y field in free space.

6.1 Free Space Propagation

In this section, different components of the EM wave radiating from a point source in free space will be observed. This will allow us to see the propagation and field distribution of different components of EM wave.

Setup: The computational domain chosen for the simulation is square shaped. The length of a side of the domain is 100. The resolution of the mesh is 10 per unit length ($\Delta = 0.1$) The boundaries of the computational domain is surrounded with PML layers similar Fig. 4.1. The source has been placed at the centre of the computational domain. The time step considered was 0.05 ($\Delta t = \Delta/2c$).

Material: The material chosen for the computational domain was free space ($\mu = 1$ and $\varepsilon = 1$). The system has been normalised with the speed of light. Therefore, $c = 1$ and $\mu_0 = 1$ and $\varepsilon_0 = 1$.

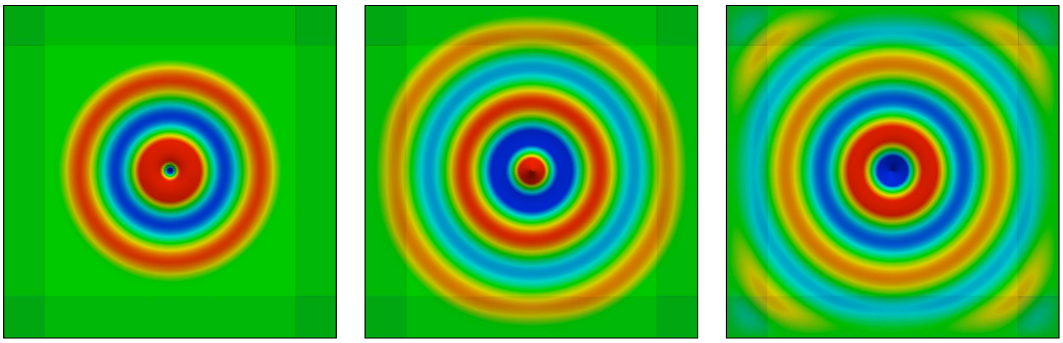
Source: Point source at the centre of the computational domain emits continuous sine wave of E_z field component (Transverse Electric). The frequency of the source is 0.15 (normalised as the speed of light is $c = 1$.).

PML: The PML thickness was 10. The PML profile used can be defined using the following equation

$$\sigma_x(r) = \sigma_y(r) = \begin{cases} 0 & r < r_0 \\ r^2 & r >= r_0 \end{cases} \tag{6.1}$$

Here, r is the distance from the centre of the computational domain in x or y direction. r_0 is the distance of the PML interface from in x or y direction.

Result: Figure 6.1(a) shows the E_z field expands uniformly with time in all directions with successive positive (red) and negative (blue) peaks in the radial direction. This field is radially symmetric and the amplitude of the field dropping with the distance from the point



(a) (b) (c)

Fig. 6.2 Computational domain shown with PML. Expansion of the E_z field with time (a) after 1000 time steps, (b) 2000 time steps and (c) 3000 time steps

source. The progressive expansion of the H_x and H_y fields are shown in Figs. 6.1(b) and (c) respectively; however their extrema are in the y and x directions, respectively with zero value along the x and y axes, respectively. In the figures, the red and blue colours correspond to the positive and negative half cycles respectively and the green colour corresponds to zero amplitude. The field components shown in Figs. 6.1 are taken after 800 time steps. At this stage the wave is well within the free space area at the middle. In fact Figs. 6.1 exclude the PML layers from the views.

Figures 6.2 show the expansion of E_z component with time. It also highlights the PML layers and Figs. 6.2(b) and (c) show the affect of PML when propagate inside the PML layer.

The wavelength calculated from the expansion of the E_z field matches the given wavelength of the point source, which allows the validation of the effectiveness of the solver.

6.2 Planar Waveguide

The goal of this example is to simulate a planer waveguide and compare the field profile obtained using the proposed method with the mode profile obtained using 1D FEM method [15, 28].

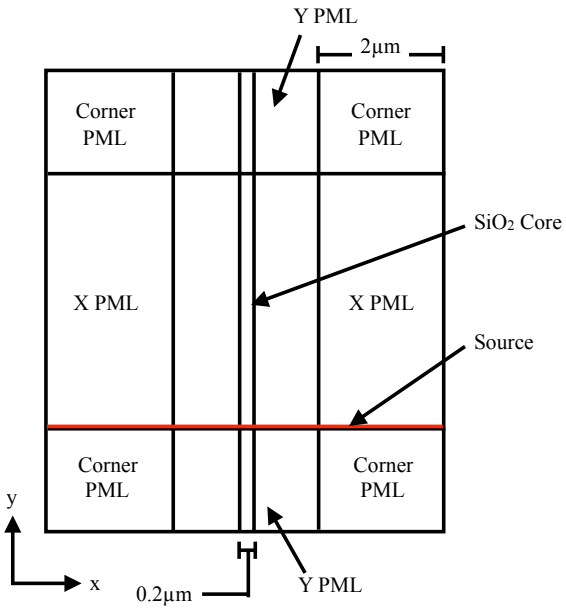


Fig. 6.3 Schematic diagram of the planner waveguide setup

Setup: The planer waveguide considered in this section is a planer waveguide with a Silicon core. The cladding of the waveguide was air. The thickness of the waveguide was $0.2\mu m$. As this is 2D both sides of the guide were surrounded by air. The width of the computational domain was $6.2\mu m$ and the height was $10\mu m$. The resolution chosen for the simulation was $\Delta = 50$ per unit length. Time step size is like the previous example is $\Delta t = \Delta/2c$. Figure 6.3 shows the schematic setup for this section.

Source: In all occasions the wavelength of this example is $1.55\mu m$. In the first case, a point source was placed at the centre the guide was used. In the second case, a line mode source which injected the field profile obtained using the FEM method into the source location. In this case a line source with the mode profile was used to emit wave into the guide.

Observer: An line observer is at the very end of the waveguide in the opposite direction of the source. The observer was with the PML layer.

PML: The PML was chosen to be $2\mu m$. The profile of the PML is the same as previous example.

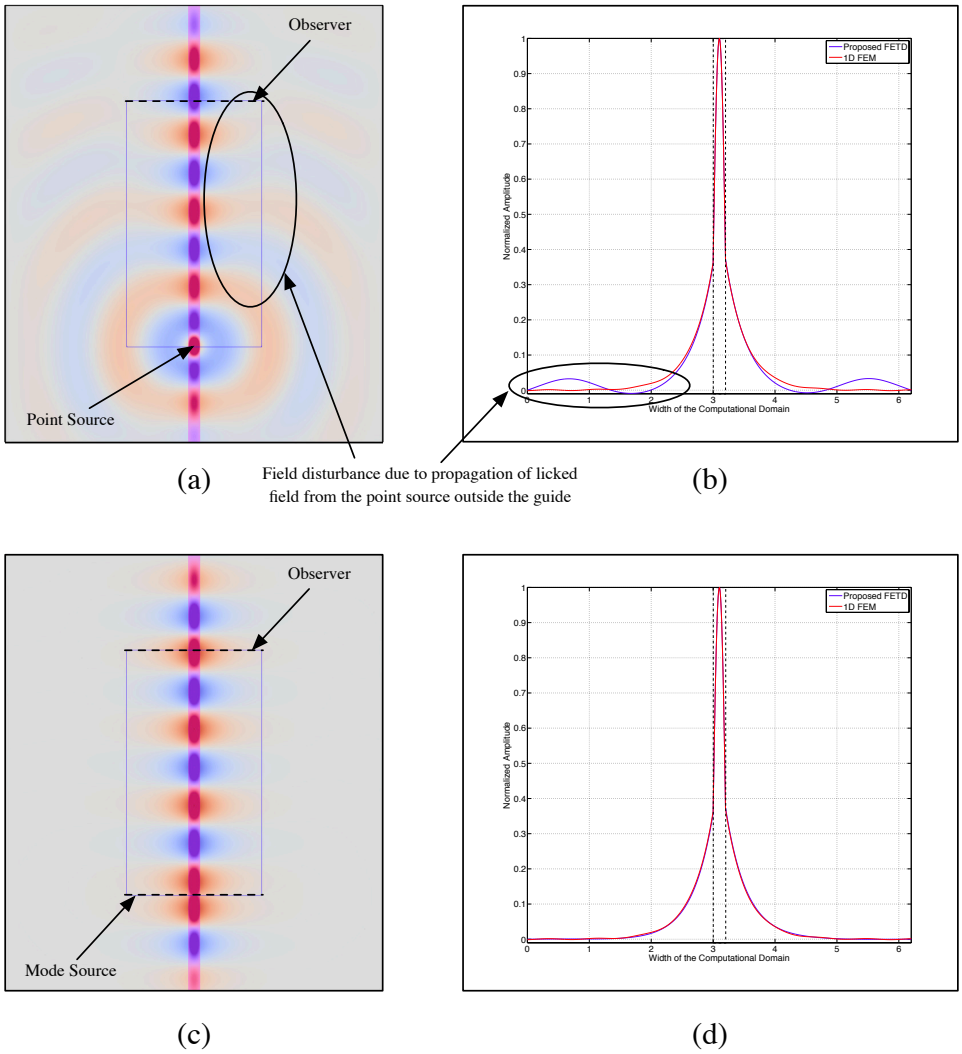


Fig. 6.4 (a) E_z field profile for a dielectric planar waveguide with a E_z point source (red and blue parts are the positive and negative half cycles of the propagating wave), (b) Comparison of E_z field profile from the proposed FETD with the point source at the observer point with the mode profile, (c) E_z field profile for a dielectric planar waveguide excited with the E_z mode profile, (d) Comparison of E_z field profile from the proposed FETD and the mode profile

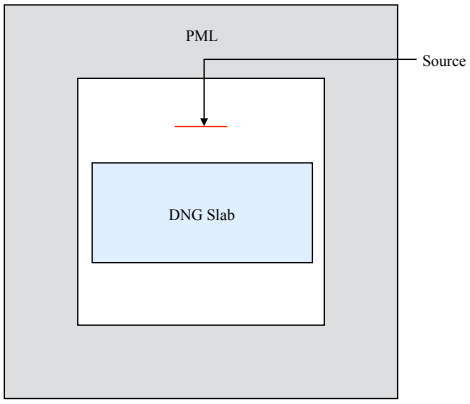
Result: Figure 6.4(a) shows the E_z field profile after 5000 time steps. It can be observed, the E_z field is mostly confined inside the Silicon core. Some of the field outside the guide is radiating away from the guide. As in this case, a E_z point source was used to initiate the propagation. So besides its evolution to a propagating mode other higher order modes were also generated and radiated subsequently. As shown in Fig. 6.4(a), an observing line was placed at the end of the guide before the PML boundary layer. The E_z field mode profile along the line was observed during the simulation. This was carried out to extract E_z field profile of the planar waveguide away from the point source. Figure 6.4(b) shows comparison of the actual mode profile shown by red line and the observed field profile evolved from the point source. The field profile from the proposed method perfectly matches the mode profile in the core region, but shows some ripple in the air cladding. This ripple is due to the radiating field as shown earlier in Fig 6.4(a).

Another simulation was performed by replacing the point source with a line source representing the mode profile. The results are presented in Fig. 6.4(c) and Fig. 6.4(d). As can be seen in Fig. 6.4(c), the structure is supporting the mode without any leakage and Fig. 6.4(d) shows the exact match of the mode profiles obtained from both the proposed FETD and the FEM mode solver. This also validates the accuracy of the newly developed approach using proposed mesh system.

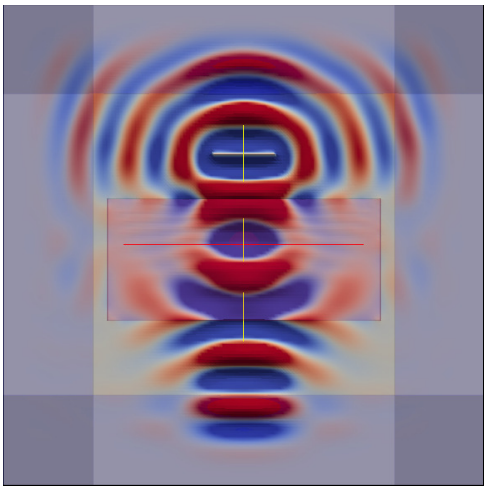
6.3 Metamaterial Flat Lens & Backward Propagation

The goal of this simulation was to show the flexibility and adaptability of the formulation. Here a dispersive Double Negative (DNG) metamaterial slab was simulated to show backward propagation of the wavefront inside and double focusing of the wave [73]. For this simulation, the governing equations for Drude model [72] derived in Chapter 5 was incorporated into the code and used inside the DNG slab. Backward propagation is also a property of the metamaterial that cannot be analysed without a time domain method. This is because it is a time domain effect.

For the simulation, a rectangular computational domain surrounded with PML boundary



(a) Schematic setup of the DNG flat lens simulation



(b) Double focusing of EM wave with a DNG flat lens

Fig. 6.5 Schematic setup and double focusing with a DNG flat lens

layer was taken. The total dimension chosen was 30×30 (including PML layers). Resolution for the simulation was 10 per unit length. The width of the PML layer was 6. A rectangular metamaterial slab was placed at the centre of the domain. The dimension of the DNG slab was 8×17. The material model for both ε and μ chosen was Drude model described in Chapter 5. For this simulation following parameter values were chosen. $\omega_{pe} = \omega_{pm} = 5.732$ and $\gamma_e = \gamma_m = 0$. An E_z line source with a length of 4 and a wavelength of 1.55 was placed 10 unit distance away from the top of the computational domain above the slab parallel to its x-axis. The time step size chosen for the simulation was 0.05 sec (normalised assuming $c = 1$). Figure 6.5a shows the schematic setup of the simulation.

Figure 6.5b shows the result of the simulation after 400 timestep or 20 secs. As shown, the wave generated from the line source approaches the metamaterial slab perpendicularly. Inside the slab the field becomes curved and focuses inside the slab. When the field comes out of the slab, again it focuses outside. The wavefront inside the guide moves in opposite direction to the direction of movement outside the guide. The wavelength used was 1.55 μm. The index of the metamaterial slab was taken as -1. Because, both the μ_r and ε_r are negative at this frequency and the index could be calculated as $n = -\sqrt{\varepsilon_r \mu_r}$.

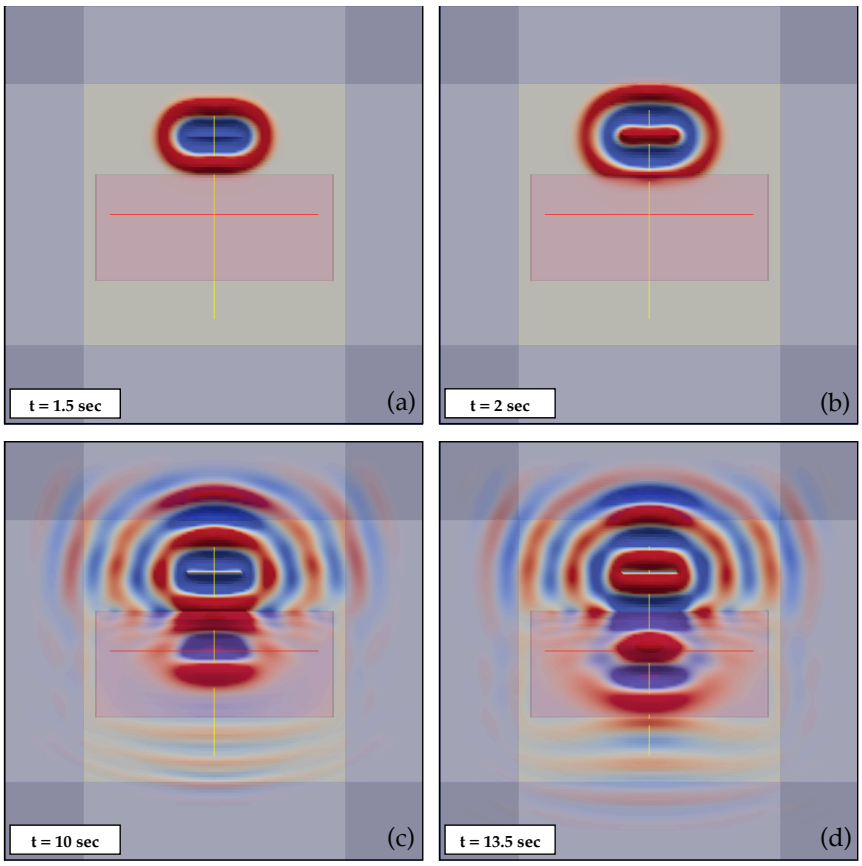


Fig. 6.6 (a) Line source radiating E_z field before incident on the DNG slab, (b) interaction of first half cycle with the DNG slab, (c) after a while when the wave start to propagation inside the DNG slab, (d) when the wave start to come out of the DNG slab on the other side

Figure 6.6 (a) shows the evolution of field before incident on metamaterial slab. As can be seen, the wave is progressively moving away from the line source. Figure 6.6 (b) the first interaction of the wave with the DNG material. The wave is decaying inside the metamaterial slab. Figure 6.6 (c) illustrates the initial forward propagation of wave through the metamaterial slab. The speed of propagation is slower than the surrounding material. Figure 6.6 (d) shows the backward wave inside the metamaterial slab and forward wave coming out the other side of the slab.

Figure 6.7 compares three successive time steps to show the backward propagation. As can be seen, the positive half cycle indicated inside the slab is moving towards the source (upward). Whereas, negative half cycle indicated of the source side is moving away from the source (downward). Positive half cycle indicated at the bottom is also moving away

from the source. Due to the negative permittivity and permeability the wave inside the slab is moving backward. This simulation takes a longer time to settle. So, the PML is required for this type of simulations. This result also agrees with results presented in [72, 77, 78].

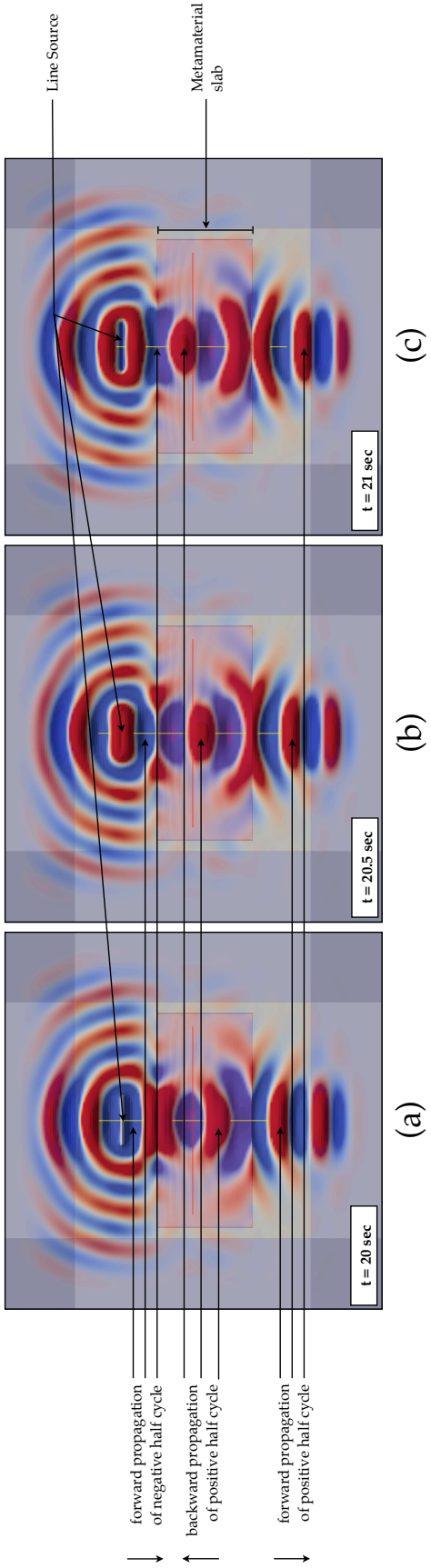


Fig. 6.7 Forward wave outside the DNG slab and backward wave inside the slab

Part II

Three-dimensional Formulation

Chapter 7

Governing Equations for Three-dimensions

To analyse the total effect of EM propagation, the proposed method has to be developed for three-dimensional geometry. The governing equations arise from Maxwell's equations (Eqs. 2.3). Both the equations in Eqs. 2.3 are intrinsically 3D in nature, as they are applied on 3D **E** and **H** vector fields. The curl operator ($\nabla\times$) is also 3D in nature. In Chapter 2 the assumption $\frac{\partial}{\partial z} = 0$, forces the Maxwell's equations to generate two sets of two-dimensional governing equations for TE (Eqs. 2.7) and TM (Eqs. 2.8) propagation respectively. If the assumption is lifted and the x, y and z directional components from the Eqs. 2.3 are separated, the following equations can be obtained.

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y}E_z - \frac{\partial}{\partial z}E_y\right) \tag{7.1a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x}E_z - \frac{\partial}{\partial z}E_x\right) \tag{7.1b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x}E_y - \frac{\partial}{\partial y}E_x\right) \tag{7.1c}$$

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial y}H_z - \frac{\partial}{\partial z}H_y\right) \tag{7.1d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_z - \frac{\partial}{\partial z}H_x\right) \tag{7.1e}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_y - \frac{\partial}{\partial y}H_x\right) \tag{7.1f}$$

Equations 7.1 are coupled equations and these six equations can be solved to calculate the time evolution of the EM wave. As can be seen in all six equations, the **E** components can always be calculated using only the **H** field components and *vice versa*.

7.1 Space-Time discretisation

To solve Eqs. 7.1 using the finite element technique, the space-time domain has to be discretised by using the finite elements. In Eqs. 7.1, all the time calculations are on the left hand side and the space calculations are on the right hand side. This split in the calculation enables avoiding the four dimensional form and makes the calculation easier by allowing the discretisation of the domains separately using two 3D meshes for the spatial dimensions and two one-dimensional (1D) meshes for the time.

7.1.1 Space discretisation

To discretise a 3D structure, linear tetrahedrons are chosen. This approach is implemented to reduce both the computational load and the memory requirement of each element to a minimum. The shape function for a tetrahedron is given by

$$N_i = a_i x + b_i y + c_i z + d_i \tag{7.2}$$

where, N is the shape function, i is the index of local node for an element and a, b, c and d are constants.

The parameters a_i, b_i, c_i and d_i can be calculated in a similar manner to the parameters calculated for Eq. 2.10.

As there are 4 nodes in every tetrahedral element, There will be 4 shape functions, N_0,

N_1, N_2 and N_3. Where, coordinate position for the nodes are,

$$\begin{aligned} \texttt{node}_0 &: (x_0, y_0) \\ \texttt{node}_1 &: (x_1, y_1) \\ \texttt{node}_2 &: (x_2, y_2) \\ \texttt{node}_3 &: (x_3, y_3) \end{aligned} \tag{7.3}$$

For N_i following system of linear equations can be obtained,

$$\begin{bmatrix} 1 & x_0 & y_0 & z_0 \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}$$

$$\Rightarrow \mathscr{A}^{-1}\mathscr{A} \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \mathscr{A}^{-1} \begin{bmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} \quad \text{where, } \mathscr{A} = \begin{bmatrix} 1 & x_0 & y_0 & z_0 \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{bmatrix}$$

$$\Rightarrow \mathscr{I}_4 \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \mathscr{A}^{-1} \begin{bmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} \quad \text{where, } \mathscr{I}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \mathscr{A}^{-1} \begin{bmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \\ \mathbf{c}_0 & \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \\ \mathbf{d}_0 & \mathbf{d}_1 & \mathbf{d}_2 & \mathbf{d}_3 \end{bmatrix} \begin{bmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}$$

$$\text{where, } \mathscr{A}^{-1} = \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \\ \mathbf{c}_0 & \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \\ \mathbf{d}_0 & \mathbf{d}_1 & \mathbf{d}_2 & \mathbf{d}_3 \end{bmatrix} \tag{7.4a}$$

The elements of the matrix can be expressed as,

$$\mathbf{a}_0 = -\frac{x_1y_2z_3 - x_1y_3z_2 - x_2y_1z_3 + x_2y_3z_1 + x_3y_1z_2 - x_3y_2z_1}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{A}_0}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{a}_1 = \frac{x_0y_2z_3 - x_0y_3z_2 - x_2y_0z_3 + x_2y_3z_0 + x_3y_0z_2 - x_3y_2z_0}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{A}_1}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{a}_2 = -\frac{x_0y_1z_3 - x_0y_3z_1 - x_1y_0z_3 + x_1y_3z_0 + x_3y_0z_1 - x_3y_1z_0}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{A}_2}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{a}_3 = \frac{x_0y_1z_2 - x_0y_2z_1 - x_1y_0z_2 + x_1y_2z_0 + x_2y_0z_1 - x_2y_1z_0}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{A}_3}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{b}_0 = \frac{y_1z_2 - y_2z_1 - y_1z_3 + y_3z_1 + y_2z_3 - y_3z_2}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{B}_0}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{b}_1 = -\frac{y_0z_2 - y_2z_0 - y_0z_3 + y_3z_0 + y_2z_3 - y_3z_2}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{B}_1}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{b}_2 = \frac{y_0z_1 - y_1z_0 - y_0z_3 + y_3z_0 + y_1z_3 - y_3z_1}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{B}_2}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{b}_3 = -\frac{y_0z_1 - y_1z_0 - y_0z_2 + y_2z_0 + y_1z_2 - y_2z_1}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{B}_3}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{c}_0 = -\frac{x_1z_2 - x_2z_1 - x_1z_3 + x_3z_1 + x_2z_3 - x_3z_2}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{C}_0}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{c}_1 = \frac{x_0z_2 - x_2z_0 - x_0z_3 + x_3z_0 + x_2z_3 - x_3z_2}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{C}_1}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{c}_2 = -\frac{x_0z_1 - x_1z_0 - x_0z_3 + x_3z_0 + x_1z_3 - x_3z_1}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{C}_2}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{c}_3 = \frac{x_0z_1 - x_1z_0 - x_0z_2 + x_2z_0 + x_1z_2 - x_2z_1}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{C}_3}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{d}_0 = \frac{x_1y_2 - x_2y_1 - x_1y_3 + x_3y_1 + x_2y_3 - x_3y_2}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{D}_0}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{d}_1 = -\frac{x_0y_2 - x_2y_0 - x_0y_3 + x_3y_0 + x_2y_3 - x_3y_2}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{D}_1}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{d}_2 = \frac{x_0y_1 - x_1y_0 - x_0y_3 + x_3y_0 + x_1y_3 - x_3y_1}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{D}_2}{\mathscr{D}\mathscr{D}}$$

$$\mathbf{d}_3 = -\frac{x_0y_1 - x_1y_0 - x_0y_2 + x_2y_0 + x_1y_2 - x_2y_1}{\mathscr{D}\mathscr{D}} = \frac{\mathbf{D}_3}{\mathscr{D}\mathscr{D}}$$

where,

$$\begin{aligned}
\mathbf{A}_0 &= -(x_1y_2z_3 - x_1y_3z_2 - x_2y_1z_3 + x_2y_3z_1 + x_3y_1z_2 - x_3y_2z_1)\\
\mathbf{A}_1 &= x_0y_2z_3 - x_0y_3z_2 - x_2y_0z_3 + x_2y_3z_0 + x_3y_0z_2 - x_3y_2z_0\\
\mathbf{A}_2 &= -(x_0y_1z_3 - x_0y_3z_1 - x_1y_0z_3 + x_1y_3z_0 + x_3y_0z_1 - x_3y_1z_0)\\
\mathbf{A}_3 &= x_0y_1z_2 - x_0y_2z_1 - x_1y_0z_2 + x_1y_2z_0 + x_2y_0z_1 - x_2y_1z_0\\
\mathbf{B}_0 &= y_1z_2 - y_2z_1 - y_1z_3 + y_3z_1 + y_2z_3 - y_3z_2\\
\mathbf{B}_1 &= -(y_0z_2 - y_2z_0 - y_0z_3 + y_3z_0 + y_2z_3 - y_3z_2)
\end{aligned}$$

$$
\begin{aligned}
\mathbf{B}_2 &= y_0z_1 - y_1z_0 - y_0z_3 + y_3z_0 + y_1z_3 - y_3z_1 \\
\mathbf{B}_3 &= -(y_0z_1 - y_1z_0 - y_0z_2 + y_2z_0 + y_1z_2 - y_2z_1) \\
\mathbf{C}_0 &= -(x_1z_2 - x_2z_1 - x_1z_3 + x_3z_1 + x_2z_3 - x_3z_2) \\
\mathbf{C}_1 &= x_0z_2 - x_2z_0 - x_0z_3 + x_3z_0 + x_2z_3 - x_3z_2 \\
\mathbf{C}_2 &= -(x_0z_1 - x_1z_0 - x_0z_3 + x_3z_0 + x_1z_3 - x_3z_1) \\
\mathbf{C}_3 &= x_0z_1 - x_1z_0 - x_0z_2 + x_2z_0 + x_1z_2 - x_2z_1 \\
\mathbf{D}_0 &= x_1y_2 - x_2y_1 - x_1y_3 + x_3y_1 + x_2y_3 - x_3y_2 \\
\mathbf{D}_1 &= -(x_0y_2 - x_2y_0 - x_0y_3 + x_3y_0 + x_2y_3 - x_3y_2) \\
\mathbf{D}_2 &= x_0y_1 - x_1y_0 - x_0y_3 + x_3y_0 + x_1y_3 - x_3y_1 \\
\mathbf{D}_3 &= -(x_0y_1 - x_1y_0 - x_0y_2 + x_2y_0 + x_1y_2 - x_2y_1)
\end{aligned}
$$

$$
\begin{aligned}
\mathscr{D}\mathscr{D} = {} & x_0y_1z_2 - x_0y_2z_1 - x_1y_0z_2 + x_1y_2z_0 + x_2y_0z_1 - x_2y_1z_0 \\
& - x_0y_1z_3 + x_0y_3z_1 + x_1y_0z_3 - x_1y_3z_0 - x_3y_0z_1 + x_3y_1z_0 \\
& + x_0y_2z_3 - x_0y_3z_2 - x_2y_0z_3 + x_2y_3z_0 + x_3y_0z_2 - x_3y_2z_0 \\
& - x_1y_2z_3 + x_1y_3z_2 + x_2y_1z_3 - x_2y_3z_1 - x_3y_1z_2 + x_3y_2z_1 \\
= {} & \mathbf{A}_3 + \mathbf{A}_2 + \mathbf{A}_1 + \mathbf{A}_0
\end{aligned}
$$

and for shape function N_i,

$$
\zeta_j = \begin{cases} 1 & i = j \\ \\ 0 & \text{Otherwise} \end{cases}
$$

For shape function N_0,

$$\begin{bmatrix} a_0 \\ b_0 \\ c_0 \\ d_0 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \\ \mathbf{c}_0 & \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \\ \mathbf{d}_0 & \mathbf{d}_1 & \mathbf{d}_2 & \mathbf{d}_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_0 \\ b_0 \\ c_0 \\ d_0 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{b}_0 \\ \mathbf{c}_0 \\ \mathbf{d}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_0/\mathscr{D}\mathscr{D} \\ \mathbf{B}_0/\mathscr{D}\mathscr{D} \\ \mathbf{C}_0/\mathscr{D}\mathscr{D} \\ \mathbf{D}_0/\mathscr{D}\mathscr{D} \end{bmatrix} \tag{7.5}$$

So, parameters of N_0 are $a_0 = \frac{\mathbf{A}_0}{\mathscr{D}\mathscr{D}}$, $b_0 = \frac{\mathbf{B}_0}{\mathscr{D}\mathscr{D}}$, $c_0 = \frac{\mathbf{C}_0}{\mathscr{D}\mathscr{D}}$ and $d_0 = \frac{\mathbf{D}_0}{\mathscr{D}\mathscr{D}}$.

Similarly, N_1 parameters are $a_1 = \frac{\mathbf{A}_1}{\mathscr{D}\mathscr{D}}$, $b_1 = \frac{\mathbf{B}_1}{\mathscr{D}\mathscr{D}}$, $c_1 = \frac{\mathbf{C}_1}{\mathscr{D}\mathscr{D}}$ and $d_1 = \frac{\mathbf{D}_1}{\mathscr{D}\mathscr{D}}$; N_2 parameters are $a_2 = \frac{\mathbf{A}_2}{\mathscr{D}\mathscr{D}}$, $b_2 = \frac{\mathbf{B}_2}{\mathscr{D}\mathscr{D}}$, $c_2 = \frac{\mathbf{C}_2}{\mathscr{D}\mathscr{D}}$ and $d_2 = \frac{\mathbf{D}_2}{\mathscr{D}\mathscr{D}}$; N_3 parameters are $a_3 = \frac{\mathbf{A}_3}{\mathscr{D}\mathscr{D}}$, $b_3 = \frac{\mathbf{B}_3}{\mathscr{D}\mathscr{D}}$, $c_3 = \frac{\mathbf{C}_3}{\mathscr{D}\mathscr{D}}$ and $d_3 = \frac{\mathbf{D}_3}{\mathscr{D}\mathscr{D}}$;

The variation of the field inside each element can be expressed by,

$$\Phi(x,y,z) = \sum_{i=1}^{M} N_i \phi_i \tag{7.6}$$

where Φ can be any field component (any one of H_x, H_y, H_z, E_x, E_y and E_z components) inside the element, ϕ_i is the field component at the i^{th} node of the element (any one of h_x, h_y, h_z, e_x, e_y, e_z) and M is total number of nodes associated with the element. For a linear element (*i.e.* four node tetrahedral elements) $M = 4$.

7.1.2 Time discretisation

In a way similar to what was shown in the literature [63], the time axis can be discretised with 1D finite elements. For a linear element the shape function can be written as shown,

$$Q_j = p_j t + q_j \tag{7.7}$$

where p_j and q_j are the coefficients of the line passing through the nodes of the time element.

The variation of field between two time nodes can be expressed as,

$$\Psi(t) = \sum_{j=1}^{P} Q_j \psi^{(j)} \tag{7.8}$$

where Ψ can be any field component (any one of the H_x, H_y, H_z, E_x, E_y, E_z components) inside the element, $\psi^{(j)}$ is the field component at j^{th} time node, P is the number of the time nodes in the time element and for linear elements, $P = 2$.

7.1.3 Discretised Governing Equations

Applying both the discretisations on Eqs. 7.1, the discretised form of the governing equations may be obtained as follows

$$h_x^{(n+1)} = -\frac{1}{p_2}\left[\frac{1}{\mu}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial y}e_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}e_{yi}^{(n)}\right) + p_1 h_x^{(n-1)}\right] \tag{7.9a}$$

$$h_y^{(n+1)} = \frac{1}{p_2}\left[\frac{1}{\mu}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}e_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}e_{xi}^{(n)}\right) - p_1 h_y^{(n-1)}\right] \tag{7.9b}$$

$$h_z^{(n+1)} = -\frac{1}{p_2}\left[\frac{1}{\mu}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}e_{yi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial y}e_{xi}^{(n)}\right) + p_1 h_z^{(n-1)}\right] \tag{7.9c}$$

$$e_x^{(n+1)} = \frac{1}{p_2}\left[\frac{1}{\varepsilon}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial y}h_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}h_{yi}^{(n)}\right) - p_1 e_x^{(n-1)}\right] \tag{7.9d}$$

$$e_y^{(n+1)} = -\frac{1}{p_2}\left[\frac{1}{\varepsilon}\left[\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}h_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}h_{xi}^{(n)}\right) + p_1 e_y^{(n-1)}\right]\right] \tag{7.9e}$$

$$e_z^{(n+1)} = \frac{1}{p_2}\left[\frac{1}{\varepsilon}\left[\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}h_{yi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial y}h_{xi}^{(n)}\right) - p_1 e_z^{(n-1)}\right]\right] \tag{7.9f}$$

where the field components with the $(n+1)$, (n) and $(n-1)$ superscripts are the future, current and the past values, respectively. It can be noted that each of the equations, when applied on one element, produces only one future value of the field. For this reason, this one value of the field cannot be placed on any of the corner nodes of the element. Therefore, the future field calculated will be stored at the centroid of each element, which is unique. It can be observed from Eqs. 7.9 that the formulation is explicit and data parallel for the calculation of the field components for each time step.

Chapter 8

The Three-dimensional Mesh

For the 3D implementation of the proposed FETD method the space and time can be dealt with a 3D space mesh system and a 1D time mesh system similar to section 3.2.

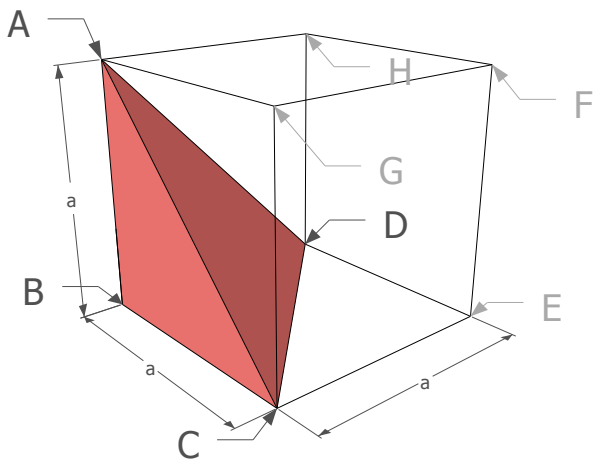


Fig. 8.1 A basic tetrahedral element inside a 3D cubic cell

8.1 The Space Mesh

For 3D space discretisation, polyhedral elements can be used. Here linear elements take the minimum memory and the minimum computation time and the 4-node tetrahedron is the linear element in a 3D FE discretisation. Therefore, a tetrahedron can be chosen as the basic

element type for space discretisation. A basic cube, given by *ABCGFHDE* in Fig. 8.1 can be taken as the initial building block and thus a tetrahedron can be obtained from this cube by drawing a plane through the points *A*, *B* and *C*. The tetrahedron *ABCD* thus generated is shaded (in red) in the figure while it should be note that the unshaded part of the cube will be left unused in the calculation. It should be noted that this is an **"Isosceles Right Angled Tetrahedron (IRT3D)"**.

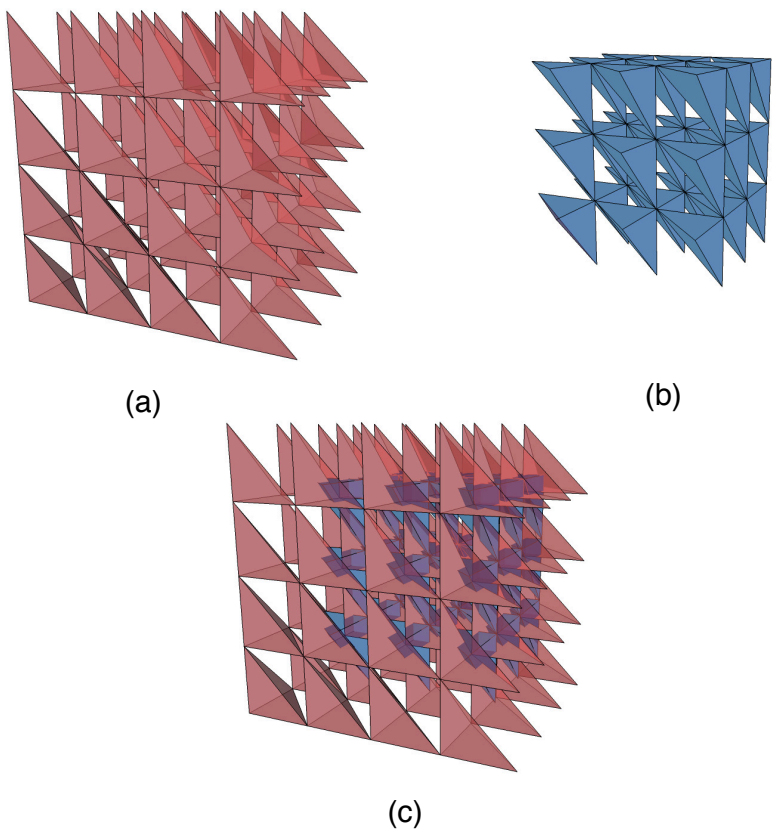


Fig. 8.2 (a) $4 \times 4 \times 4$ main mesh generated by using the basic element of Fig. 8.1, (b) the $3 \times 3 \times 3$ auxiliary mesh generated using the centroid of the main mesh elements in Fig. 8.2(a), (c) both meshes together

A mesh can then be generated using the basic tetrahedron by adding further tetrahedra in all directions. Figure 8.2(a) shows a $4 \times 4 \times 4$ mesh generated using the basic element presented in Fig. 8.1. As can be seen, there are hollow spaces inside the mesh, giving an overall mesh discretisation, in a way similar to the mesh presented in previous work [63] presented in Section 3.1.

At this stage if all the **E** components are placed in the corner nodes of all the elements of the mesh, Eqs. 7.9a, 7.9b and 7.9c will produce one value for all the **H** field components at the centroid of each of the elements. To calculate the **E** field components from these **H** field components stored at the centroid, an auxiliary mesh is required which contains only the centroids of the main mesh as the corner nodes and the corner nodes of the main mesh as centroids. Figure 8.2(b) shows the auxiliary mesh for the main mesh presented in Fig. 8.2(a). It can be observed that the mesh in Fig. 8.2(b) is similar in nature and Fig. 8.2(c) shows both the meshes together.

It should be noted that, unlike the situation in the FDTD method where the components of both fields are all staggered in space at different points, the meshing system proposed puts all components of the same field at the same node. This implies, if all components of the **E** field are stored in the corner nodes of the main mesh, that all components of the **H** field will be stored at the corner nodes of the auxiliary mesh.

This proposed meshing technique will allow a more accurate representation of structures consisting of non-magnetic materials. The reason for this is that in the EM time domain and non-magnetic materials, the device structure may be generated with the appropriate permittivity and the interface between the materials. As all the **E** field components are at the same point, only one material interface on the mesh exists to represent the physical boundary of the device. By contrast, with the FDTD method all the field components are staggered at different points in space, so for any 3D structure with non-magnetic materials, there must be 3 different interfaces for each physical boundary. As a result, the FDTD always makes a representation of the physical boundaries that is inaccurate.

Although the mesh presented in this section is uniform, a more accurate approximation of the structure can be obtained by moving the nodes on the interfaces. This can be performed if necessary. To make better approximation this technique does not require increased resolution. For greater efficiency and accuracy, an advance meshing scheme can be developed. For the FDTD method, moving a node is more difficult as it is a grid based method. Therefore, a finer grid is needed to increase the accuracy of the representation of the structure, causing much higher memory requirement and increased CPU time.

Another major advantage of the proposed technique is the use of the unique mesh system. To discretise a cube with tetrahedral elements, a minimum of 5 tetrahedra are required. The method discussed considers only one tetrahedron and therefore, compared to a full mesh finite element approach, this method could be up to 5 times faster. This makes the method better suited to 3D calculations when compared to any other finite element time domain method so far reported. As the number of elements that needs to be calculated is very large for 3D structures and for time domain analysis all the elements have to be calculated for each time step, these advantages play a vital role in the adoption of the 3D FETD as a more appropriate approach.

8.2 The Time Mesh

As for time discretisation there is no difference between Eqs. 2.13, 2.14 and Eqs. 7.9. Therefore, the mesh system presented in section 3.2 can be used without any alteration for the 3D implementation of the proposed method.

Chapter 9

Perfectly Matched Layer Boundary for Three-dimensions

To eliminate unwanted reflection from the boundary of the cuboid computational domain, the Perfectly Matched Layer (PML) boundary has to be derived and implemented of the proposed 3D FETD implementation [79]. For three dimensional cuboid computational domain, 7 types of PML has to be derived to avoid any non-physical reflection from the boundary. They are,

1. X PML
2. Y PML
3. Z PML
4. XY PML
5. YZ PML
6. ZX PML
7. XYZ PML

Similar to Chapter 4, the Auxiliary Differential Equation (ADE) approach was used for the derivation of the governing equations for different PML layers. To derive different set of governing equations for different PML layers modified form ∇ operators were used. The modified ∇s are as follows,

$$\tilde{\nabla}_x = \hat{x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{9.1a}$$

$$\tilde{\nabla}_y = \hat{x}\frac{\partial}{\partial x} + \hat{y}\left(1 - j\frac{\sigma_y}{\omega}\right)^{-1}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{9.1b}$$

$$\tilde{\nabla}_z = \hat{x}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\left(1 - j\frac{\sigma_z}{\omega}\right)^{-1}\frac{\partial}{\partial z} \tag{9.1c}$$

$$\tilde{\nabla}_{xy} = \hat{x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1}\frac{\partial}{\partial x} + \hat{y}\left(1 - j\frac{\sigma_y}{\omega}\right)^{-1}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{9.1d}$$

$$\tilde{\nabla}_{yz} = \hat{x}\frac{\partial}{\partial x} + \hat{y}\left(1 - j\frac{\sigma_y}{\omega}\right)^{-1}\frac{\partial}{\partial y} + \hat{z}\left(1 - j\frac{\sigma_z}{\omega}\right)^{-1}\frac{\partial}{\partial z} \tag{9.1e}$$

$$\tilde{\nabla}_{zx} = \hat{x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\left(1 - j\frac{\sigma_z}{\omega}\right)^{-1}\frac{\partial}{\partial z} \tag{9.1f}$$

$$\tilde{\nabla}_{xyz} = \hat{x}\left(1 - j\frac{\sigma_x}{\omega}\right)^{-1}\frac{\partial}{\partial x} + \hat{y}\left(1 - j\frac{\sigma_y}{\omega}\right)^{-1}\frac{\partial}{\partial y} + \hat{z}\left(1 - j\frac{\sigma_z}{\omega}\right)^{-1}\frac{\partial}{\partial z} \tag{9.1g}$$

9.1 X PML

The X PML is introduced to absorb any field component propagating along x-axis. To derive governing equations for the X PML layer, the ordinary ∇s of Eqs. 2.3 were replace with the modified ∇_x from Eqs. 9.1a. The coefficients of $\hat{x}$, $\hat{y}$ and $\hat{z}$ were separated to obtain the governing equations of the X PML layer for 3D FETD. The governing equations are as follows,

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y}E_z - \frac{\partial}{\partial z}E_y\right) \tag{9.2a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x}E_z - \frac{\partial}{\partial z}E_x - \mathscr{E}_{x[z]}\right) - \sigma_x H_y \tag{9.2b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x}E_y - \frac{\partial}{\partial y}E_x - \mathscr{E}_{x[y]}\right) - \sigma_x H_z \tag{9.2c}$$

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial y}H_z - \frac{\partial}{\partial z}H_y\right) \tag{9.2d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_z - \frac{\partial}{\partial z}H_x - \mathscr{H}_{x[z]}\right) - \sigma_x E_y \tag{9.2e}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_y - \frac{\partial}{\partial y}H_x - \mathscr{H}_{x[y]}\right) - \sigma_x E_z \tag{9.2f}$$

$$\frac{d\mathscr{E}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z} E_x \tag{9.2g}$$

$$\frac{d\mathscr{E}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y} E_x \tag{9.2h}$$

$$\frac{d\mathscr{H}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z} H_x \tag{9.2i}$$

$$\frac{d\mathscr{H}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y} H_x \tag{9.2j}$$

Here, $\mathscr{E}_{x[z]}$, $\mathscr{E}_{x[y]}$, $\mathscr{H}_{x[z]}$ and $\mathscr{H}_{x[y]}$ are auxiliary field for the X PML layer which are calculated using Eqs. 9.2g, 9.2h, 9.2i and 9.2j, respectively.

9.2 Y PML

Similarly, the governing equations for the Y PML can be derived using ∇_y from Eq. 9.1b.

The governing equations for Y PML are,

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y} E_z - \frac{\partial}{\partial z} E_y - \mathscr{E}_{y[z]}\right) - \sigma_y H_x \tag{9.3a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x} E_z - \frac{\partial}{\partial z} E_x\right) \tag{9.3b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x} E_y + \mathscr{E}_{y[x]} - \frac{\partial}{\partial y} E_x\right) - \sigma_y H_z \tag{9.3c}$$

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial y} H_z - \frac{\partial}{\partial z} H_y - \mathscr{H}_{y[z]}\right) - \sigma_y E_x \tag{9.3d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\left(\frac{\partial}{\partial x} H_z - \frac{\partial}{\partial z} H_x\right) \tag{9.3e}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial x} H_y + \mathscr{H}_{y[x]} - \frac{\partial}{\partial y} H_x\right) - \sigma_y E_z \tag{9.3f}$$

$$\frac{d\mathscr{E}_{y[z]}}{dt} = \sigma_y \frac{\partial}{\partial z} E_y \tag{9.3g}$$

$$\frac{d\mathscr{E}_{y[x]}}{dt} = \sigma_y \frac{\partial}{\partial x} E_y \tag{9.3h}$$

$$\frac{d\mathscr{H}_{y[z]}}{dt} = \sigma_y \frac{\partial}{\partial z} H_y \tag{9.3i}$$

$$\frac{d\mathscr{H}_{y[x]}}{dt} = \sigma_y \frac{\partial}{\partial x} H_y \tag{9.3j}$$

Here, $\mathscr{E}_{y[z]}$, $\mathscr{E}_{y[x]}$, $\mathscr{H}_{y[z]}$ and $\mathscr{H}_{y[x]}$ are the associated auxiliary fields.

9.3 Z PML

Governing equations for Z PML are,

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y} E_z + \mathscr{E}_{z[y]} - \frac{\partial}{\partial z} E_y\right) - \sigma_z H_x \tag{9.4a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x} E_z + \mathscr{E}_{z[x]} - \frac{\partial}{\partial z} E_x\right) - \sigma_z H_y \tag{9.4b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x} E_y - \frac{\partial}{\partial y} E_x\right) \tag{9.4c}$$

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial y} H_z + \mathscr{H}_{z[y]} - \frac{\partial}{\partial z} H_y\right) - \sigma_z E_x \tag{9.4d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\left(\frac{\partial}{\partial x} H_z + \mathscr{H}_{z[x]} - \frac{\partial}{\partial z} H_x\right) - \sigma_z E_y \tag{9.4e}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial x} H_y - \frac{\partial}{\partial y} H_x\right) \tag{9.4f}$$

$$\frac{d\mathscr{E}_{z[y]}}{dt} = \sigma_z \frac{\partial}{\partial y} E_z \tag{9.4g}$$

$$\frac{d\mathscr{E}_{z[x]}}{dt} = \sigma_z \frac{\partial}{\partial x} E_z \tag{9.4h}$$

$$\frac{d\mathscr{H}_{z[y]}}{dt} = \sigma_z \frac{\partial}{\partial y} H_z \tag{9.4i}$$

$$\frac{d\mathscr{H}_{z[x]}}{dt} = \sigma_z \frac{\partial}{\partial x} H_z \tag{9.4j}$$

Here, $\mathscr{E}_{z[y]}$, $\mathscr{E}_{z[x]}$, $\mathscr{H}_{z[y]}$ and $\mathscr{H}_{z[x]}$ are the associated auxiliary fields.

9.4 XY PML

Governing equations for XY PML are,

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y}E_z - \frac{\partial}{\partial z}E_y - \mathscr{E}_{y[z]}\right) - \sigma_y H_x \tag{9.5a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x}E_z - \frac{\partial}{\partial z}E_x - \mathscr{E}_{x[z]}\right) - \sigma_x H_y \tag{9.5b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x}E_y + \mathscr{E}_{y[x]} - \frac{\partial}{\partial y}E_x - \mathscr{E}_{x[y]}\right) - \sigma_x H_z - \sigma_y H_z - \Phi_z \tag{9.5c}$$

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial y}H_z - \frac{\partial}{\partial z}H_y - \mathscr{H}_{y[z]}\right) - \sigma_y E_x \tag{9.5d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_z - \frac{\partial}{\partial z}H_x - \mathscr{H}_{x[z]}\right) - \sigma_x E_y \tag{9.5e}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_y + \mathscr{H}_{y[x]} - \frac{\partial}{\partial y}H_x - \mathscr{H}_{x[y]}\right) - \sigma_x E_z - \sigma_y E_z - \Psi_z \tag{9.5f}$$

$$\frac{d\mathscr{E}_{y[z]}}{dt} = \sigma_y \frac{\partial}{\partial z}E_y \tag{9.5g}$$

$$\frac{d\mathscr{E}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z}E_x \tag{9.5h}$$

$$\frac{d\mathscr{E}_{y[x]}}{dt} = \sigma_y \frac{\partial}{\partial x}E_y \tag{9.5i}$$

$$\frac{d\mathscr{E}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y}E_x \tag{9.5j}$$

$$\frac{d\mathscr{H}_{y[z]}}{dt} = \sigma_y \frac{\partial}{\partial z}H_y \tag{9.5k}$$

$$\frac{d\mathscr{H}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z}H_x \tag{9.5l}$$

$$\frac{d\mathscr{H}_{y[x]}}{dt} = \sigma_y \frac{\partial}{\partial x}H_y \tag{9.5m}$$

$$\frac{d\mathscr{H}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y}H_x \tag{9.5n}$$

$$\frac{d\Phi_z}{dt} = \sigma_x \sigma_y H_z \tag{9.5o}$$

$$\frac{d\Psi_z}{dt} = \sigma_x \sigma_y E_z \tag{9.5p}$$

Here, $\mathscr{E}_{y[z]}$, $\mathscr{E}_{x[z]}$, $\mathscr{E}_{y[x]}$, $\mathscr{E}_{x[y]}$, $\mathscr{H}_{y[z]}$, $\mathscr{H}_{x[z]}$, $\mathscr{H}_{y[x]}$, $\mathscr{H}_{x[y]}$, Φ_z and Ψ_z are the associated

auxiliary fields.

9.5 YZ PML

Governing equations for YZ PML are,

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y}E_z + \mathscr{E}_{z[y]} - \frac{\partial}{\partial z}E_y - \mathscr{E}_{y[z]}\right) - \sigma_y H_x - \sigma_z H_x - \Phi_x \tag{9.6a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x}E_z + \mathscr{E}_{z[x]} - \frac{\partial}{\partial z}E_x\right) - \sigma_z H_y \tag{9.6b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x}E_y + \mathscr{E}_{y[x]} - \frac{\partial}{\partial y}E_x\right) - \sigma_y H_z \tag{9.6c}$$

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial y}H_z + \mathscr{H}_{z[y]} - \frac{\partial}{\partial z}H_y - \mathscr{H}_{y[z]}\right) - \sigma_y E_x - \sigma_z E_x - \Psi_x \tag{9.6d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_z + \mathscr{H}_{z[x]} - \frac{\partial}{\partial z}H_x\right) - \sigma_z E_y \tag{9.6e}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_y + \mathscr{H}_{y[x]} - \frac{\partial}{\partial y}H_x\right) - \sigma_y E_z \tag{9.6f}$$

$$\frac{d\mathscr{E}_{z[y]}}{dt} = \sigma_z \frac{\partial}{\partial y}E_z \tag{9.6g}$$

$$\frac{d\mathscr{E}_{y[z]}}{dt} = \sigma_y \frac{\partial}{\partial z}E_y \tag{9.6h}$$

$$\frac{d\mathscr{E}_{z[x]}}{dt} = \sigma_z \frac{\partial}{\partial x}E_z \tag{9.6i}$$

$$\frac{d\mathscr{E}_{y[x]}}{dt} = \sigma_y \frac{\partial}{\partial x}E_y \tag{9.6j}$$

$$\frac{d\mathscr{H}_{z[y]}}{dt} = \sigma_z \frac{\partial}{\partial y}H_z \tag{9.6k}$$

$$\frac{d\mathscr{H}_{y[z]}}{dt} = \sigma_y \frac{\partial}{\partial z}H_y \tag{9.6l}$$

$$\frac{d\mathscr{H}_{z[x]}}{dt} = \sigma_z \frac{\partial}{\partial x}H_z \tag{9.6m}$$

$$\frac{d\mathscr{H}_{y[x]}}{dt} = \sigma_y \frac{\partial}{\partial x}H_y \tag{9.6n}$$

$$\frac{d\Phi_x}{dt} = \sigma_y \sigma_z H_x \tag{9.6o}$$
$$\frac{d\Psi_x}{dt} = \sigma_y \sigma_z E_x \tag{9.6p}$$

Here, $\mathscr{E}_{z[y]}$, $\mathscr{E}_{y[z]}$, $\mathscr{E}_{z[x]}$, $\mathscr{E}_{y[x]}$, $\mathscr{H}_{z[y]}$, $\mathscr{H}_{y[z]}$, $\mathscr{H}_{z[x]}$, $\mathscr{H}_{y[x]}$, Φ_x and Ψ_x are the auxiliary variables.

9.6 ZX PML

Governing equations for ZX PML are,

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y}E_z + \mathscr{E}_{z[y]} - \frac{\partial}{\partial z}E_y\right) - \sigma_z H_x \tag{9.7a}$$
$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x}E_z + \mathscr{E}_{z[x]} - \frac{\partial}{\partial z}E_x - \mathscr{E}_{x[z]}\right) - \sigma_z H_y - \sigma_x H_y - \Phi_y \tag{9.7b}$$
$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x}E_y - \frac{\partial}{\partial y}E_x - \mathscr{E}_{x[y]}\right) - \sigma_x H_z \tag{9.7c}$$
$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial y}H_z + \mathscr{H}_{z[y]} - \frac{\partial}{\partial z}H_y\right) - \sigma_z E_x \tag{9.7d}$$
$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_z + \mathscr{H}_{z[x]} - \frac{\partial}{\partial z}H_x - \mathscr{H}_{x[z]}\right) - \sigma_z E_y - \sigma_x E_y - \Psi_y \tag{9.7e}$$
$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial x}H_y - \frac{\partial}{\partial y}H_x - \mathscr{H}_{x[y]}\right) - \sigma_x E_z \tag{9.7f}$$
$$\frac{d\mathscr{E}_{z[x]}}{dt} = \sigma_z \frac{\partial}{\partial x}E_z \tag{9.7g}$$
$$\frac{d\mathscr{E}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z}E_x \tag{9.7h}$$
$$\frac{d\mathscr{E}_{z[y]}}{dt} = \sigma_z \frac{\partial}{\partial y}E_z \tag{9.7i}$$
$$\frac{d\mathscr{E}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y}E_x \tag{9.7j}$$
$$\frac{d\mathscr{H}_{z[y]}}{dt} = \sigma_z \frac{\partial}{\partial y}H_z \tag{9.7k}$$
$$\frac{d\mathscr{H}_{z[x]}}{dt} = \sigma_z \frac{\partial}{\partial x}H_z \tag{9.7l}$$

$$\frac{d\mathscr{H}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z} H_x \tag{9.7m}$$

$$\frac{d\mathscr{H}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y} H_x \tag{9.7n}$$

$$\frac{d\Phi_y}{dt} = \sigma_z \sigma_x H_y \tag{9.7o}$$

$$\frac{d\Psi_y}{dt} = \sigma_z \sigma_x E_y \tag{9.7p}$$

Here, $\mathscr{E}_{z[x]}$, $\mathscr{E}_{x[z]}$, $\mathscr{E}_{z[y]}$, $\mathscr{E}_{x[y]}$, $\mathscr{H}_{z[y]}$, $\mathscr{H}_{z[x]}$, $\mathscr{H}_{x[z]}$, $\mathscr{H}_{x[y]}$, Φ_y and Ψ_y are the auxiliary variables.

9.7 XYZ PML

Governing equations for XYZ PML are,

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y} E_z + \mathscr{E}_{z[y]} - \frac{\partial}{\partial z} E_y - \mathscr{E}_{y[z]}\right) - \sigma_y H_x - \sigma_z H_x - \Phi_x \tag{9.8a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x} E_z + \mathscr{E}_{z[x]} - \frac{\partial}{\partial z} E_x - \mathscr{E}_{x[z]}\right) - \sigma_z H_y - \sigma_x H_y - \Phi_y \tag{9.8b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x} E_y + \mathscr{E}_{y[x]} - \frac{\partial}{\partial y} E_x - \mathscr{E}_{x[y]}\right) - \sigma_x H_z - \sigma_y H_z - \Phi_z \tag{9.8c}$$

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial y} H_z + \mathscr{H}_{z[y]} - \frac{\partial}{\partial z} H_y - \mathscr{H}_{y[z]}\right) - \sigma_y E_x - \sigma_z E_x - \Psi_x \tag{9.8d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\left(\frac{\partial}{\partial x} H_z + \mathscr{H}_{z[x]} - \frac{\partial}{\partial z} H_x - \mathscr{H}_{x[z]}\right) - \sigma_z E_y - \sigma_x E_y - \Psi_y \tag{9.8e}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial}{\partial x} H_y + \mathscr{H}_{y[x]} - \frac{\partial}{\partial y} H_x - \mathscr{H}_{x[y]}\right) - \sigma_x E_z - \sigma_y E_z - \Psi_z \tag{9.8f}$$

$$\frac{d\mathscr{E}_{z[y]}}{dt} = \sigma_z \frac{\partial}{\partial y} E_z \tag{9.8g}$$

$$\frac{d\mathscr{E}_{y[z]}}{dt} = \sigma_y \frac{\partial}{\partial z} E_y \tag{9.8h}$$

$$\frac{d\mathscr{E}_{z[x]}}{dt} = \sigma_z \frac{\partial}{\partial x} E_z \tag{9.8i}$$

$$\frac{d\mathscr{E}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z} E_x \tag{9.8j}$$

$$\frac{d\mathscr{E}_{y[x]}}{dt} = \sigma_y \frac{\partial}{\partial x} E_y \tag{9.8k}$$

$$\frac{d\mathscr{E}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y} E_x \tag{9.8l}$$

$$\frac{d\mathscr{H}_{z[y]}}{dt} = \sigma_z \frac{\partial}{\partial y} H_z \tag{9.8m}$$

$$\frac{d\mathscr{H}_{y[z]}}{dt} = \sigma_y \frac{\partial}{\partial z} H_y \tag{9.8n}$$

$$\frac{d\mathscr{H}_{z[x]}}{dt} = \sigma_z \frac{\partial}{\partial x} H_z \tag{9.8o}$$

$$\frac{d\mathscr{H}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z} H_x \tag{9.8p}$$

$$\frac{d\mathscr{H}_{y[x]}}{dt} = \sigma_y \frac{\partial}{\partial x} H_y \tag{9.8q}$$

$$\frac{d\mathscr{H}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y} H_x \tag{9.8r}$$

$$\frac{d\Phi_x}{dt} = \sigma_y \sigma_z H_x \tag{9.8s}$$

$$\frac{d\Phi_y}{dt} = \sigma_z \sigma_x H_y \tag{9.8t}$$

$$\frac{d\Phi_z}{dt} = \sigma_x \sigma_y H_z \tag{9.8u}$$

$$\frac{d\Psi_x}{dt} = \sigma_y \sigma_z E_x \tag{9.8v}$$

$$\frac{d\Psi_y}{dt} = \sigma_z \sigma_x E_y \tag{9.8w}$$

$$\frac{d\Psi_z}{dt} = \sigma_x \sigma_y E_z \tag{9.8x}$$

Here, $\mathscr{E}_{z[y]}$, $\mathscr{E}_{y[z]}$, $\mathscr{E}_{z[x]}$, $\mathscr{E}_{x[z]}$, $\mathscr{E}_{y[x]}$, $\mathscr{E}_{x[y]}$, $\mathscr{H}_{z[y]}$, $\mathscr{H}_{y[z]}$, $\mathscr{H}_{z[x]}$, $\mathscr{H}_{x[z]}$, $\mathscr{H}_{y[x]}$, $\mathscr{H}_{x[y]}$, Φ_x, Φ_y, Φ_z, Ψ_x, Ψ_y and Ψ_z are the auxiliary variables.

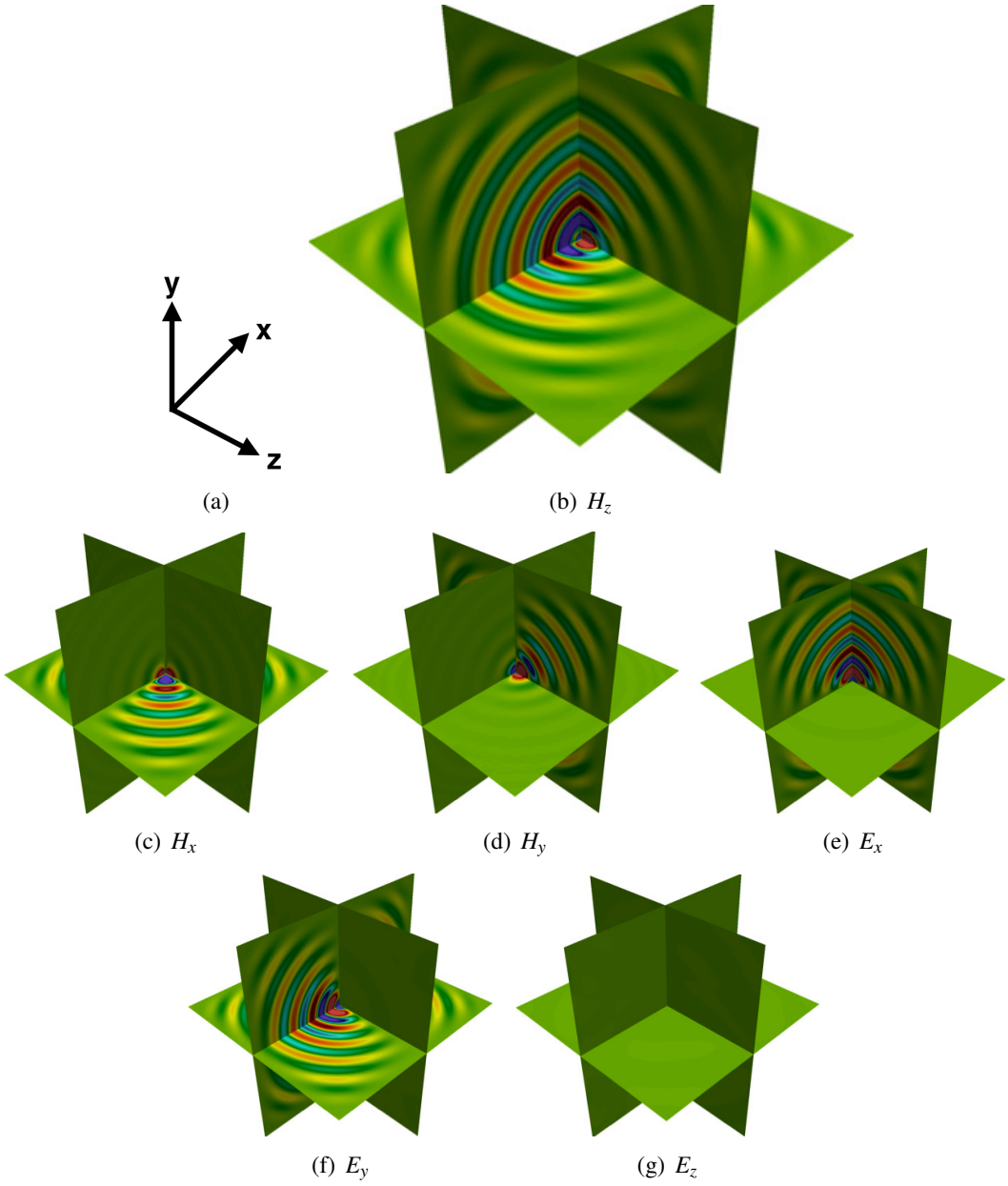
Chapter 10

Results of Simulations in Three-dimensions

A C++ code was developed to perform the numerical simulations. Using the threading mechanism in C++11 the code was made multi-threaded. The overall implementation was made dimensionless or scale invariant [24] by taking the speed of light as $c = 1$. Similar to the 2D implementation described in Chapter 6, the permeability and permittivity of vacuum are taken as $\mu_0 = 1$ and $\varepsilon_0 = 1$, respectively. The output of the program is stored in the VTK file format to visualize with Paraview software.

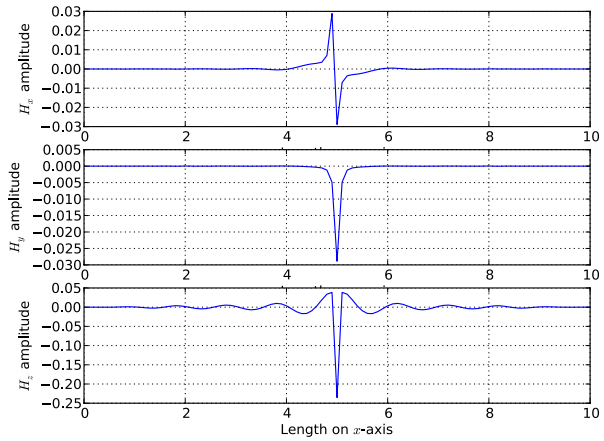
10.1 Free Space Propagation

The simplest possible test which can be performed using the code is the free space propagation from a point source. To carry out this test, an H_z continuous sine wave point source was placed at the centre of a cubic computational domain and the wavelength chosen was $1\mu m$. The dimension of the cubic computational domain wss $10 \times 10 \times 10$. The resolution of the mesh was chosen to be 10 per unit length. The time step size chosen was 0.01 sec (normalised considering light speed $c = 1$). The PML was placed around the computational domain to truncate it and absorb all the reflections from the boundary of the computational domain. The depth of the PML was 2 unit length. To illustrate the output of the simulation,

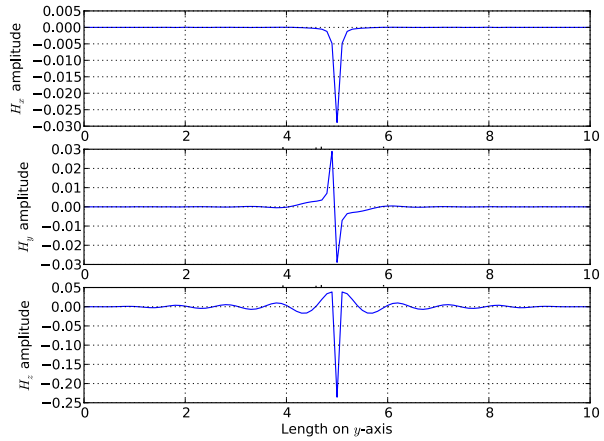


(a) (b) H_z (c) H_x (d) H_y (e) E_x (f) E_y (g) E_z

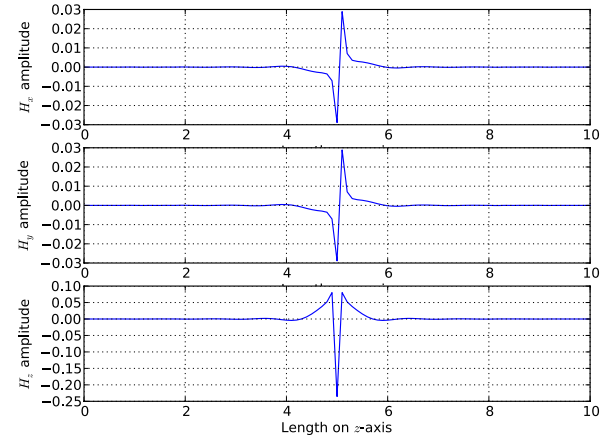
Fig. 10.1 (a) Axes of all plots are shown in this figure, (b) H_z, (c) H_x, (d) H_y, (e) E_x, (f) E_y and (g) E_z field distribution in space after 1000 time steps for a H_z point source at the middle of the computational domain.



(a) Propagation of **H** field components along x-axis



(b) Propagation of **H** field components along y-axis



(c) Propagation of **H** field components along z-axis

Fig. 10.2 Line plot over all three axes through the centre of the computational domain for the free space propagation

three slices were taken from the volume after 1000 time steps where the centre of each slice was placed at the centre of the computational domain. The 'slicing planes' considered were parallel to the *xy*, *yz* and *zx* planes of the computational domain and Fig. 10.1 shows the field distribution in these planes.

As can be seen, all the fields radiating from the centre of the computational domain have been absorbed at the outer boundary and there is no sign of reflection visible inside the domain. Figure 10.1 (b), shows the distribution of the H_z field and although the field is inserted directly by the source at the centre, the field propagates uniformly in *xy*-plane. The field intensity decreases gradually when the propagation direction changes from the *xy*-plane towards the *z* axis and there is no propagation of the H_z field in the *z* direction.

The propagations of the H_x and the H_y fields are mostly confined in the *zx* and the *yz*-planes, respectively. The propagation of the field It should be noted that these H_x and H_y fields were evolved from the H_z point source. However, the E_x and the E_y fields do not propagate in the *zx* and the *yz*, planes respectively. For this setup with the H_z source, the E_z field does not develop and Fig. 10.1 (g) shows no presence of a E_z field in that domain.

It should also be noted that H_x and H_y fields do not evolve on any of the axises. They expand in 45° with the axises of *zx* and *zy* planes respectively. Figure 10.2 shows the field expansion on the three axis in space after 1000 time steps. It confirms that the H_x and H_y fields do not evolve on any of the axises. But H_z field with a decaying sinusoidal variation on *x* and *y* axis.

10.2 Nanowires

Next, to illustrate the guiding of an EM wave in a waveguide, a silicon nano-guide is simulated and the cross section of the nanowire can be seen in Fig. 10.3 (a). The *Si* core of the waveguide was fabricated on a SiO_2 buffer layer where the waveguide considered here is $240nm$ in height and $500nm$ in width. An H_y sine wave point source at a wavelength of $1.55\mu m$ wavelength was placed at the centre of the guide cross section at $z = 1.1\mu m$. The resolution for the simulation was 50 nodes$/\mu m$ where the time-step size was taken as $\Delta/2c$

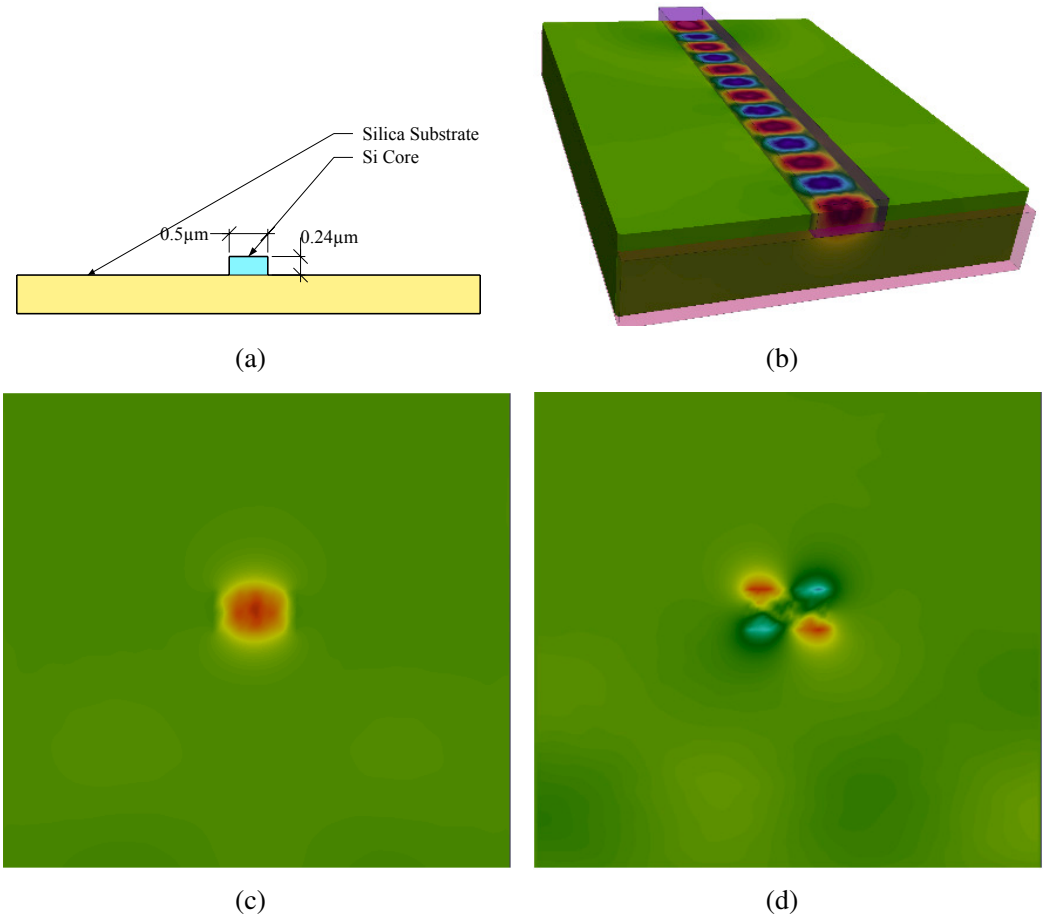


Fig. 10.3 (a) Cross section of the nanowire, (b) Propagation of H_y field inside the guide, (c) Mode profile of the dominant H_y field, (d) Mode profile of the non-dominant H_x field.

and Δ is the size of a cell in μm with c being the speed of light in $\mu m/s$.

As the radiation from the point source propagates along the guide, the mode profile of the guide then develops. Figure 10.3 (b) shows the propagation of the H_y field injected inside the waveguide after 5000 time steps. Figures 10.3 (c) and 10.3 (d) show the mode profiles for the H_y and H_x fields, respectively. These mode profiles were captured at a distance of $3\mu m$ from the source on a plane parallel to xy-plane. It is encouraging that this result agrees well with the mode profile of the same silicon nanowire, obtained by using a full-vectorial finite element method (VFEM) [80]. The effective index of the guide was also calculated by using Eq. 10.1 [81], as shown below

$$n_{\text{eff}} = \frac{1}{k_0 d}\left(\phi|_b - \phi|_a\right) \tag{10.1}$$

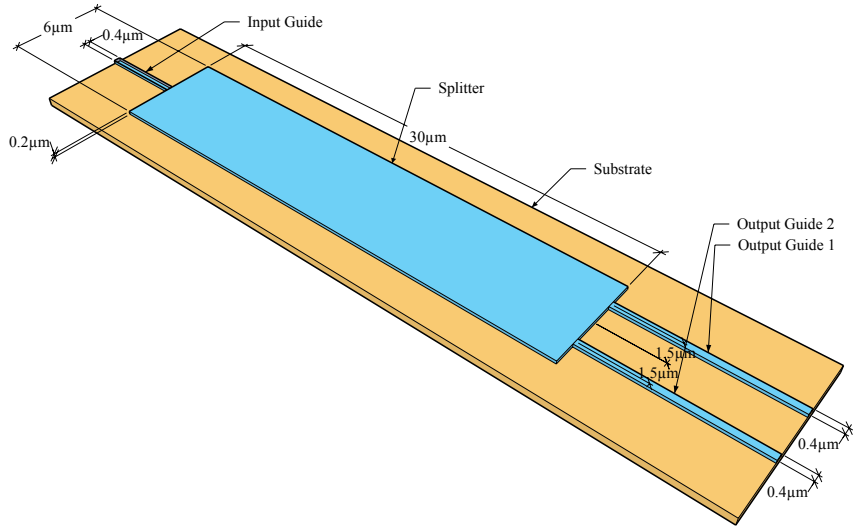
where, k_0 is the wave vector for free space propagation in a vacuum; $\phi|_a$ and $\phi|_b$ are the phases of the wave at two different observation points a and b at the same time and d is the distance between these two points.

To calculate the effective index, the field distribution along the central axis of the nanowire was determined after 5000 time-steps. The phases were recorded at distances of $3\mu m$ and $4\mu m$ away from the source on the central axis and from these two fields the effective index of the structure was calculated to be 2.4721.

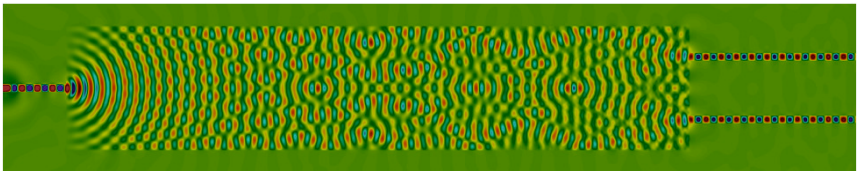
To benchmark this result obtained for the effective index, the same nanowire design was simulated using the VFEM [15, 28]. Similar H_x and H_y field patterns were obtained for the fundamental mode from the VFEM and as a result, the effective index was calculated to be equal to 2.4751. It is pleasing that the results obtained are very close, considering that the proposed method uses discrete time and space axes, whereas for the VFEM method is a steady state solver and the propagation axis considered to be continuous.

10.3 Nano Power Splitter

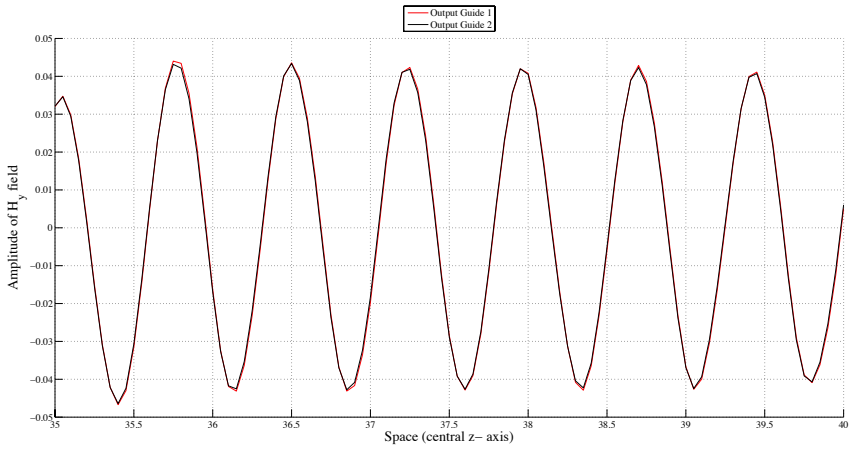
To evaluate further the code developed in this work, a structure which is non-uniform along the axial direction was considered and its characteristics simulated. For this purpose, a



(a)



(b)



(c)

Fig. 10.4 (a) Schematic 3D diagram of the nano power splitter (sky blue part is the *Si* core and orange coloured part is SiO_2 substrate), (b) H_y field profile at the central zx-plane of the power splitter after 9000 time-steps, (c) Comparison of the H_y field of the output waves in output guide 1 and 2 at the central z-axis after 9000 time-steps

compact MMI-based power splitter was considered [82] where the core of the power splitter was considered to be fabricated from *Si* and the buffer layer was SiO_2. The structure consists of three parts, as discussed below

1. The input guide
2. The splitter section
3. The two output guides

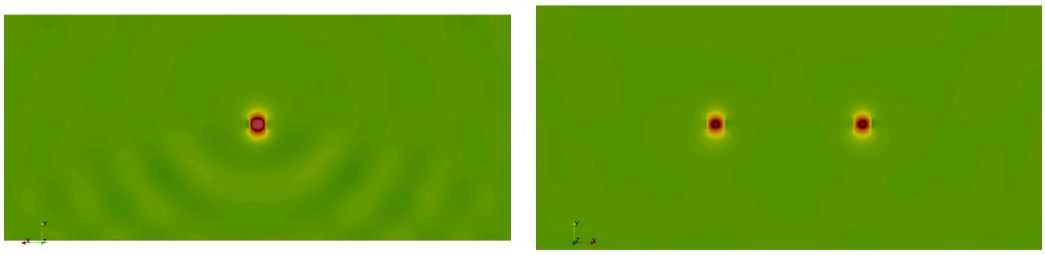
The height of the *Si* structure was taken as $200\,nm$ in all the sections. The widths of the input and output guides were taken to be $400\,nm$. The multimoded splitter section was considered to be $6\,\mu m$ in width and $30\,\mu m$ in length. These type of power splitters are called "multimode interference power splitter". When the light enters the multimoded splitter section, it spreads out into the larger core and wave reflected from the boundary of the splitter section interfere and produce multiple modes. The design methodology for this type of device was presented in [83]. The input power into the splitter section is coupled only to the even modes due to the symmetrical design. These modes propagate with different phase velocities, shaping different power profiles. The propagation constant of these even modes can be approximated by,

$$\beta_m = \frac{2\pi}{\lambda}\sqrt{N^2 - \frac{(2m+1)^2\lambda^2}{4W_{eff}^2}} \simeq \beta_0 - \frac{(m^2+m)\lambda\pi}{W_{eff}^2 N} \tag{10.2}$$

Here, N is the effective index of the guiding layer, λ the vacuum wavelength, $m(0,1,2)$ the even mode number, and W_{eff} is the modified width of the multimode rectangular section, assuming the mode is perfectly confined inside the section. This modified width is slightly larger than the real width. It is defined as,

$$W_{eff} = \frac{\lambda}{2\sqrt{N^2 - n_f^2}} \tag{10.3}$$

Here, n_f is the effective index of the fundamental mode of the multimode section. So, all the even modes constructively interfere at a distance along the propagation direction defined



(a) (b)

Fig. 10.5 (a) H_y field profile propagating in the nano input wire, (b) H_y field profile after splitting the input power into two at the output nanowires

by

$$d = \frac{NW_{eff}^2}{\lambda} \tag{10.4}$$

where, the input image is reproduced. At distances defined by d/n, where n is an arbitrary positive integer number, the modes interfere shaping a pattern with n images equally spaced.

In the simulation an H_y field point source at a $1.55\,\mu m$ wavelength was placed at the center of the input guide where the length of the input guide was chosen to be $5\,\mu m$ to form a mode before entering the splitter section. These output waveguides were placed $1.5\mu m$ away from the center of the splitter and the resolution taken for the simulation was 20 nodes$/\mu m$, with the time-step size being $\Delta/2.5c$. Figure 10.4 (a) shows a 3D schematic diagram of the structure where Figure 10.4 (b), shows the H_y field distribution after 9000 time-steps (180 secs considering light speed $c = 1$). It can clearly be seen from Fig. 10.4 (c) that the output waves are both equal in amplitude and phase. Therefore, the structure divides the input into two coherent and equal amplitude parts at the output waveguides and the mode profiles of the H_y field both at the input and output of the nanowires are shown in Figs. 10.5 (a) and 10.5 (b), respectively.

10.4 Nano Directional Coupler

For a rigorous benchmark with the full-vectorial FEM mode solver [15, 28] a nano directional coupler was simulated with variable spacing between the two nanowires. A schematic diagram is presented in Fig. 10.6. The orange part of the diagram is the buffer layer or substrate. The two blue parts are the two nanowares of the directional coupler. The width and height of the cross-section of the two nanowires are taken as $0.5\,\mu m$ and $0.2\,\mu m$, respectively. The distance between the two guides are taken as s which was varied between $0.1-0.5\,\mu m$. The core material of the nanowires is Si which is placed on top of SiO_2 buffer layer. The top of the guide is surrounded by air. The source of excitation was considered to be a monochromatic source of wavelength $1.55\,\mu m$. Hence, the refractive indices of Si, SiO_2 and air are taken as 3.5, 1.455 and 1.0, respectively.

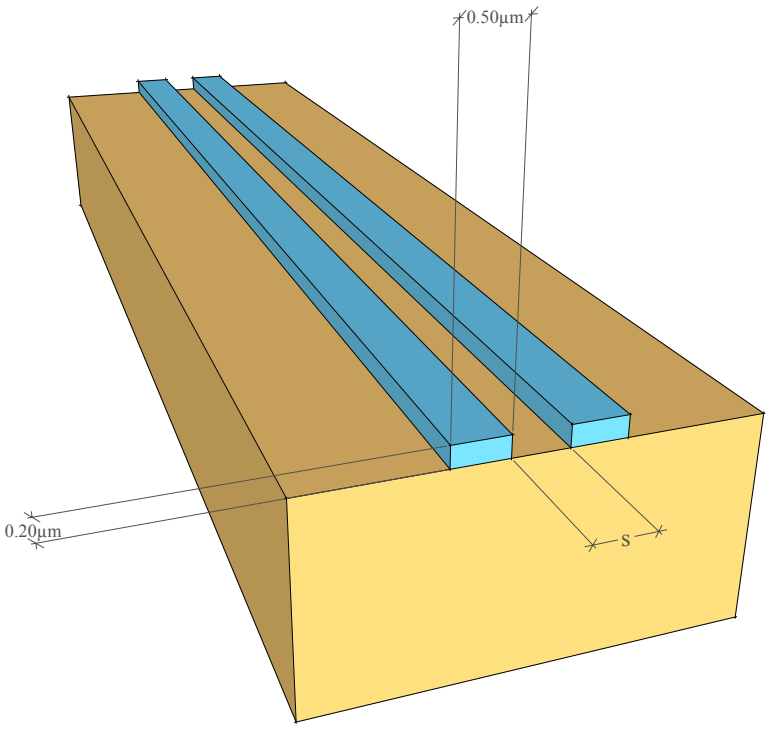
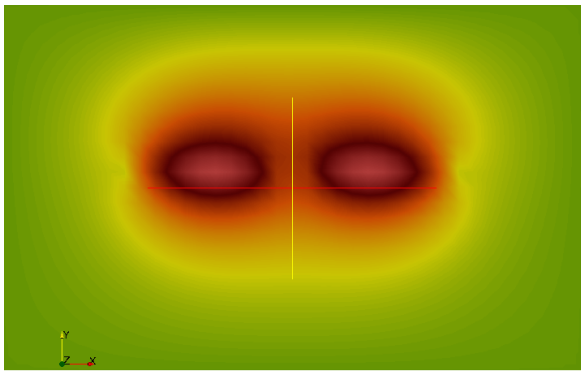
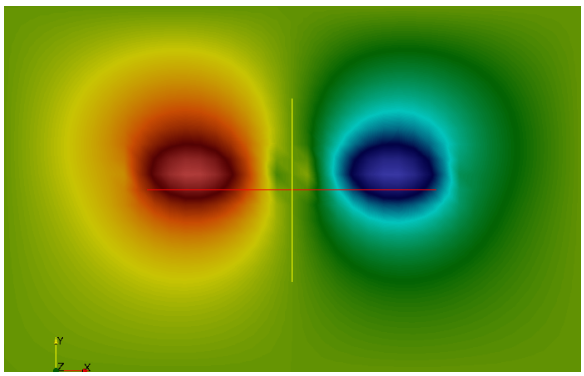


Fig. 10.6 Schematic diagram of the nano directional coupler



(a) Even mode of the nano directional coupler



(b) Odd mode of the nano directional coupler

Fig. 10.7 Odd and Even mode associated with the nano directional coupler

10.4.1 Full Vectorial FEM Analysis

To obtain the coupling distance of the nano directional coupler the odd and even supermodes of the structure was obtained and their corresponding propagation constant were taken. The coupling distance was calculated using the following formula,

$$L_c = \frac{\pi}{|\beta_{\text{even}} - \beta_{\text{odd}}|} \tag{10.5}$$

Here, L_c is the coupling length of the directional coupler, β_{even} and β_{odd} are the propagation constants of even and odd supermodes respectively. The odd and even modes for the guide presented in Fig. 10.6 when the separation of the nanowires is $0.1\mu m$ is shown in Fig. 10.7.

10.4.2 Calculation of Coupling Length using the FETD 3D

To obtain the the coupling length of a nano directional coupler, the 3D structure of Fig. 10.6 was created with 3D mesh system. The computational domain was surrounded with PML layers and a continuous point source of $1.55\mu m$ was placed into left nanowire. As the nanowire structure only support fundamental mode, the propagating light inside the guide quickly form mode and started to couple from the left nanowire to the right nanowire and at one point in the propagation direction the entire wave moves from the the left nanowire to the right nanowire.

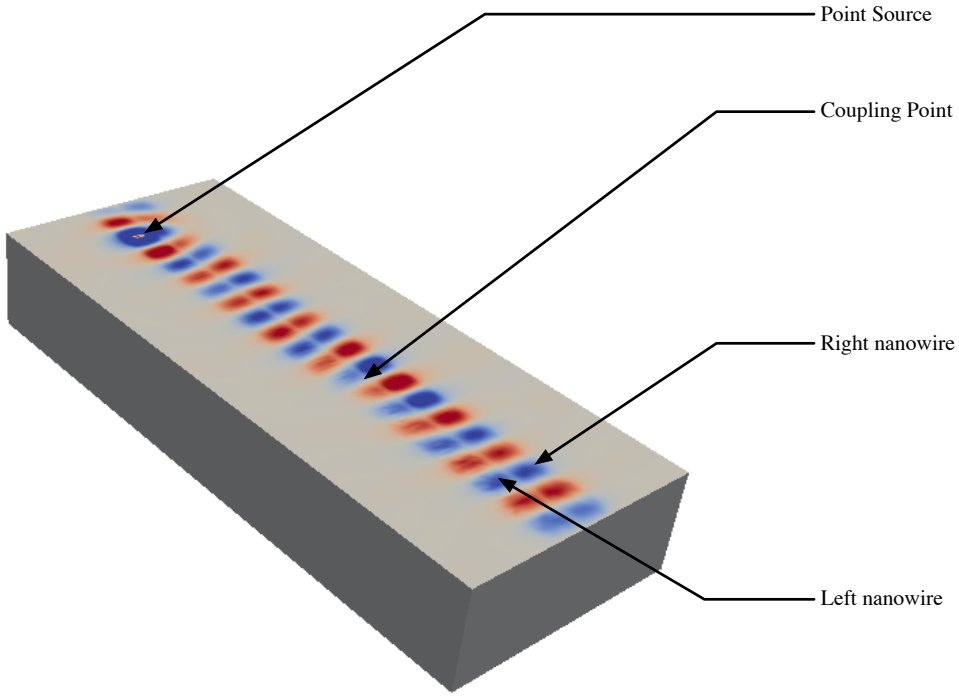


Fig. 10.8 Modelling the nano directional coupler using the proposed 3D FETD with nanowire separation of $s = 0.1\mu m$ after 2450 time-steps

The computational domain considered was $4 \times 3 \times 15\mu m^3$. The resolution chosen was 20 per unit length. The time step size chosen was $0.025\,sec$ (normalised). The depth of the PML layer was $1\mu m$. A H_z field point sine wave point source with $1.55\mu m$ wavelength was placed at the point $(1.65, 1.6, 1.1)$ at the centre of one waveguide.

Simulations were performed by the increasing the nanowire separation s from $0.1\,\mu m$ to $0.5\,\mu m$. A slice of the computational domain parallel to the $z-x$ plain at $y = 1.6\,\mu m$

has been shown in Fig. 10.8. The H_x field profile shown in the figure was taken after 2450 time-steps. The structure taken into consideration has a separation of $0.1\mu m$ between the guides. The field profile presented clearly shows the transfer of propagating wave from one guide to the other.

To obtain the exact coupling length the minimum amplitude point on the propagation axis has to be found in the left nanowire. To obtain that all the **H** field profiles of the computational domain for all time-steps were saved and only the field distribution along the central line of the left guide was extracted from all the field profile using a Python macro in Paraview. Using the extracted data the magnitude profile of the **H** field at each times-step can be obtained,

$$H_m = \sqrt{{H_x}^2 + {H_y}^2 + {H_z}^2} \tag{10.6}$$

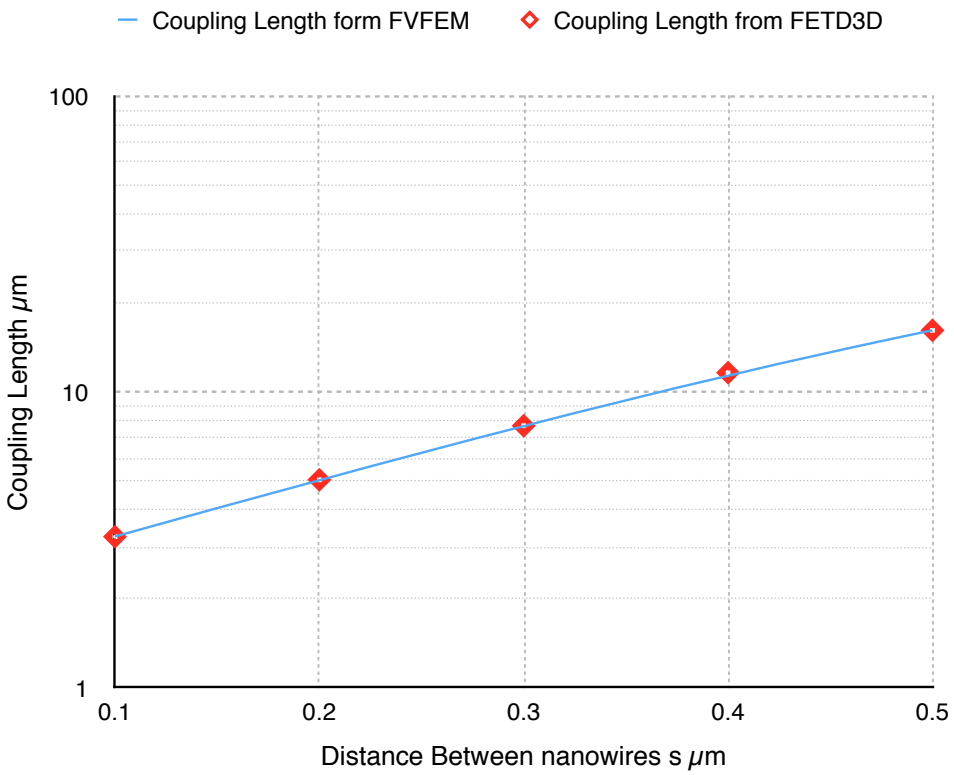


Fig. 10.9 Comparison of coupling length obtained from FVFEM and the proposed 3D FETD method

If the source is stationary at the same point the maximum and minimum magnitude point on the structure will appear at the same point along the guide. Therefore, the average magnitude profile over time can show the maximum and minimum magnitude points on the structure.

The average magnitude profile $H_{m(avg)}$ at can be calculated using Eq. 10.7

$$H_{m(avg)} = \frac{\sum_{i=0}^{N} H_{m(i)}}{N} \tag{10.7}$$

Here, N is the total number of time-steps calculated with the 3D FETD.

The coupling length can be obtained by finding the position of the minima and maxima in $H_{m(avg)}$ and calculating the distance between these two successive maximum and minimum points. To obtain the precise coupling length a MATLAB code was developed. Figure 10.9 shows the comparison between calculation of coupling length using the Full-vectorial FEM method and the proposed 3D FETD method. As can be seen the results obtained from the 3D FETD are very close to the results obtained from the full-vectorial FEM method.

Part III

Performance Analysis

Chapter 11

Numerical Dispersion

For the numerical simulation, the computational domain has to be discretised. However, due to the discretisation, phase error can be introduced into the propagating plane wave. The reason for the numerical dispersion can be both the shape function and and the position of the nodes of an element. In most occasions a fixed shape function is determines the distribution of the field components inside an element. In most situations the shape function do not represent the distribution function of the propagating wave and thus introduces error. The position of the nodes of an element in space also can introduce error. This is because the nodes directly effect the shape and behaviour of the shape function inside the domain. *i.e.* A uniform triangle might introduce less error then a very narrow counterpart. The order of the shape function also play a vital role in numerical dispersion. Because a higher order function most of the time produces a better approximation then its lower order counterpart. In this respect a mesh with more uniform and higher order elements might be better than a mesh with lower order very narrow shaped elements. As a result of the numerical dispersion, the speed of propagation may be slower than the actual speed of the wave. The phase lag with the actual wave increases with the length of its propagation in the computational domain. The issue with the phase error gets worse when the error varies with the direction of propagation. The result of this is different speeds of propagation in different directions, which is equivalent to an artificial anisotropy imposed on the wave by the discretisation even when the material is actually isotropic. This phenomenon is known as **"Numerical**

Dispersion" [36] or **"Numeric Anisotropy"** [84]. This dispersion can be minimised by increasing the resolution of the discretisation [12], but to do this would require more memory and more computations.

The method proposed here can be very efficient with the discretisation and can represent the structure accurately with fewer elements. However, even if the structure was accurately discretised with an efficient meshing algorithm, if the numerical dispersion of the output mesh remains high, an erroneous solution can be obtained for a longer propagation distance. To reduce the error, the resolution of the mesh needs to be increased. As a result, there will be little benefit in terms of memory usage and computational load. Therefore, to achieve maximum memory and computational efficiency, a mesh with a minimal numerical dispersion need to be considered.

11.1 Numerical Dispersion for Two-dimensional Formulation

For convenience of calculation, the matrix form of the equations of Eqs. 2.13 and Eqs. 2.14 will be used. For TE propagation, equations Eqs. 2.13 can be written as,

$$\left\{h_{x\langle l\rangle}^{(n)}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^{T} = -\frac{1}{\mu}\left\{e_{zk}^{[m]}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^{T} \tag{11.1a}$$

$$\left\{h_{y\langle l\rangle}^{(n)}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^{T} = \frac{1}{\mu}\left\{e_{zk}^{[m]}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^{T} \tag{11.1b}$$

$$\left\{e_{z\langle k\rangle}^{(m)}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^{T} = \frac{1}{\varepsilon}\left(\left\{h_{yl}^{[n]}\right\}\left\{\frac{\partial N_l}{\partial x}\right\}^{T} - \left\{h_{xl}^{[n]}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}^{T}\right) \tag{11.1c}$$

where, superscript T is the transpose operator to convert a row matrix into a column matrix and vice versa.

Section 3.1 introduced one main and one auxiliary mesh to hold the **E** and **H** field components. In Section 3.2, the time nodes were divided into sets **M** and **N** for the **E** and **H**

fields, respectively. In Eqs. 11.1, the field components for the space node of the main and the auxiliary meshes are denoted with k and l subscripts, respectively. For the time nodes, the field components for the members of $\mathbf{M}$ and $\mathbf{N}$ are denoted with (m) and (n) superscripts, respectively. The angle brackets $\langle\ \rangle$ are used to denote the centroid of the current element and the square brackets $[\ ]$ are used to denote the current time.

Following steps were previously been used to derive the numerical dispersion relation [25]. To calculate the numerical dispersion of the proposed method similar steps were used.

To study the numerical dispersion relation, a monochromatic source was assumed for the TE mode of propagation where E_z, H_x and H_y can be expressed as

$$h_{xl}^{(n)} = H_{x0} e^{j(\omega t^{(n)} - \tilde{\kappa}_x x_l - \tilde{\kappa}_y y_l)} \tag{11.2a}$$

$$h_{yl}^{(n)} = H_{y0} e^{j(\omega t^{(n)} - \tilde{\kappa}_x x_l - \tilde{\kappa}_y y_l)} \tag{11.2b}$$

$$e_{zk}^{(m)} = E_{z0} e^{j(\omega t^{(m)} - \tilde{\kappa}_x x_k - \tilde{\kappa}_y y_k)} \tag{11.2c}$$

where, $\bar{\kappa} = \hat{x}\tilde{\kappa}_x + \hat{y}\tilde{\kappa}_y$ is the numerical wave vector, ω is the frequency of the source and H_{x0}, H_{y0} and E_{z0} are the amplitudes of the H_x, H_y and E_z field components, respectively.

Applying Eqs. 11.2 to Eq. 11.1a and Eq. 11.1b, the expressions for H_{x0} and H_{y0} (in terms of E_{z0}) can be obtained.

$$H_{x0} = -\frac{E_{z0}}{\mu} \cdot \frac{\left\{ e^{-j(\tilde{\kappa}_x \Delta x_k + \tilde{\kappa}_y \Delta y_k)} \right\} \left\{ \frac{\partial N_k}{\partial y} \right\}^T}{\left\{ e^{j\omega \Delta t^{(n)}} \right\} \left\{ \frac{dQ^{(n)}}{dt} \right\}^T} \tag{11.3a}$$

$$H_{y0} = \frac{E_{z0}}{\mu} \cdot \frac{\left\{ e^{-j(\tilde{\kappa}_x \Delta x_k + \tilde{\kappa}_y \Delta y_k)} \right\} \left\{ \frac{\partial N_k}{\partial x} \right\}^T}{\left\{ e^{j\omega \Delta t^{(n)}} \right\} \left\{ \frac{dQ^{(n)}}{dt} \right\}^T} \tag{11.3b}$$

where, $\Delta x_{k(i)} = x_{k(i)} - x_{\langle l \rangle}$, $\Delta y_{k(i)} = y_{k(i)} - y_{\langle l \rangle}$, $\Delta t_\tau^{(n)} = t_\tau^{(n)} - t^{[m]}$, i is the local index of a node in space element and τ is the local index of a node in time element.

Applying Eqs. 11.2 and Eqs. 11.3 on Eq. 11.1c and dividing both sides of the equation by E_{z0}, the numerical dispersion relation can be obtained

$$\begin{aligned}
&\left\{e^{j\omega\Delta t^{(n)}}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^T\cdot\left\{e^{j\omega\Delta t^{(m)}}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^T = \\
&v_p^2\cdot\left(\left\{e^{-j(\tilde{\kappa}_x\Delta x_k+\tilde{\kappa}_y\Delta y_k)}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^T\cdot\left\{e^{-j(\tilde{\kappa}_x\Delta x_l+\tilde{\kappa}_y\Delta y_l)}\right\}\left\{\frac{\partial N_l}{\partial x}\right\}^T\right. \\
&\left.+\left\{e^{-j(\tilde{\kappa}_x\Delta x_k+\tilde{\kappa}_y\Delta y_k)}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^T\cdot\left\{e^{-j(\tilde{\kappa}_x\Delta x_l+\tilde{\kappa}_y\Delta y_l)}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}^T\right)
\end{aligned} \tag{11.4}$$

where, $v_p=\frac{1}{\sqrt{\mu\varepsilon}}$, $\Delta x_{l(i)}=x_{l(i)}-x_{\langle k\rangle}$, $\Delta y_{l(i)}=y_{l(i)}-y_{\langle k\rangle}$, $\Delta t_\tau^{(m)}=t_\tau^{(m)}-t^{[n]}$, i is the local index of a node in space element and τ is the local index of a node in time element.

For omnidirectional propagation in an isotropic medium, Eq. 11.4 can be written as

$$\begin{aligned}
&\left\{e^{j\omega\Delta t^{(n)}}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^T\cdot\left\{e^{j\omega\Delta t^{(m)}}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^T = \\
&v_p^2\cdot\left(\left\{e^{-j\tilde{\kappa}(\Delta x_k\cos\phi+\Delta y_k\sin\phi)}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^T\cdot\left\{e^{-j\tilde{\kappa}(\Delta x_l\cos\phi+\Delta y_l\sin\phi)}\right\}\left\{\frac{\partial N_l}{\partial x}\right\}^T\right. \\
&\left.+\left\{e^{-j\tilde{\kappa}(\Delta x_k\cos\phi+\Delta y_k\sin\phi)}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^T\cdot\left\{e^{-j\tilde{\kappa}(\Delta x_l\cos\phi+\Delta y_l\sin\phi)}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}^T\right)
\end{aligned} \tag{11.5}$$

where, $\tilde{\kappa}_x=\tilde{\kappa}\cos\phi$, $\tilde{\kappa}_y=\tilde{\kappa}\sin\phi$ and ϕ is the direction angle of propagation with respect to the x-axis. Equation 11.5 does not assume any specific shape for the mesh. As a result, this relationship holds for all types nodal meshes including all types of linear triangular meshes. The following sections will only compare 2 types of linear triangle meshes. This is because of the low resource usage for the linear triangle meshes.

Equation 11.5 can be used with Newton's iterative method to obtain the numerical wave vector $\tilde{\kappa}$. Newton's method, also called the "Newton-Raphson" method, is a root-finding algorithm that uses the first few terms of the Taylor series of a function in the vicinity of a suspected root. In the work of Taflove and Hagness [12] a similar technique was

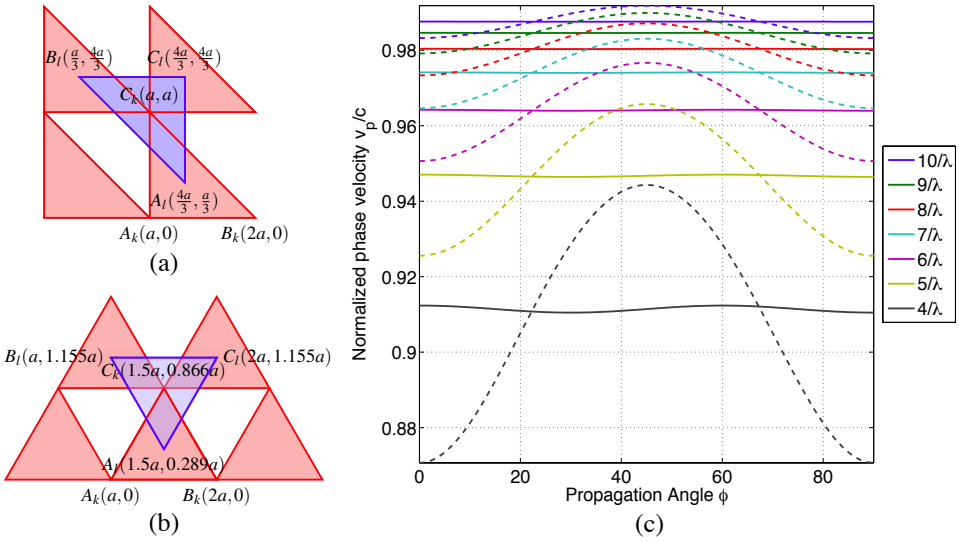


Fig. 11.1 Calculation of phase velocity using (a) IRT mesh and (b) ET mesh, (c) Comparison of phase velocities in different directions for different resolutions in IRT and ET meshes. Dashed and solid lines are normalised phase velocity curves for the IRT and the ET meshes, respectively

used to calculate the wave vector in all directions. The normalized propagation velocity, $v_p/c = 2\pi/\tilde{\kappa}_{final}$ can be calculated using the final converged value for a specific angle of propagation.

11.1.1 Calculation of Numerical Dispersion

To calculate the numerical dispersion of the mesh presented in Section 3.1, Newton's iterative method was implemented using Eq. 11.5 in MATLAB. This code was used to calculate the phase velocity of the EM wave in different directions using two different meshes: first, the IRT mesh used in Section 3.1 (shown in Fig. 11.1(a)) and the second, an "Equilateral Triangle (ET) Mesh" shown in Fig. 11.1(b). The calculation of the numerical dispersion was performed for different resolutions. For simplicity, the resolution of a mesh is expressed in the form m/λ where, the resolution is m points per wavelength. This convention will be followed throughout this thesis.

Figure 11.1(c) shows the phase velocity variation for resolutions from $4/\lambda$ to $10/\lambda$, with the propagation angle ϕ in degrees. As it can be seen, the phase velocities of the IRT

mesh (dashed lines) show a high variation with the angle, ϕ. On the other hand, phase velocities of the ET mesh (solid lines) are almost constant. A slight ripple can be seen for more coarse resolutions of $4/\lambda$ and $5/\lambda$. Resolutions higher than $5/\lambda$ show no variation in phase velocity for the ET mesh. However, for the IRT mesh, a high phase velocity variation is visible for all resolutions, as shown in Fig. 11.1(c).

A more precise measurement of the numerical dispersion can be obtained from the standard deviation of the phase velocities for different propagation angles. Figure 11.3 shows the relationship between the standard deviation of the normalized phase velocity with the resolution for both type of meshes. The horizontal green dashed line in Fig. 11.3a denotes the standard deviation of the phase velocity of $30/\lambda$ resolution with the IRT mesh (3.25×10^{-4}). As can be seen, the numerical dispersion of $5/\lambda$ resolution for the ET mesh is below the dashed line and the standard deviation of the phase velocity is lower (2.132×10^{-4}) than that of the $30/\lambda$ IRT mesh. So, the numerical dispersion is less for the ET mesh with $5/\lambda$ resolution than the IRT mesh with a much finer resolution of $30/\lambda$. This allows a reduction of resolution by a factor of 6 which can be used for the ET mesh to obtain a similar numerical dispersion. This factor can be called the "Resolution Reduction Factor (RRF)".

When the resolution of the ET mesh is increased, the RRF also increases. Figure 11.3b shows a comparison of the standard deviation of the phase velocity of the $200/\lambda$ IRT mesh with that of the ET mesh. As can be seen, a slightly lower standard deviation can be obtained using a resolution of only $11/\lambda$ in the ET mesh with the RRF value of more than 18. To make a more precise measurement of the RRF, the following mathematical procedure is used.

Although the standard deviation of the normalised velocity of the wave for the IRT and the ET meshes behave in a different manner the mean of the velocity does not show significant variation. Figure 11.2 shows the comparison. As can be seen there is little difference between them.

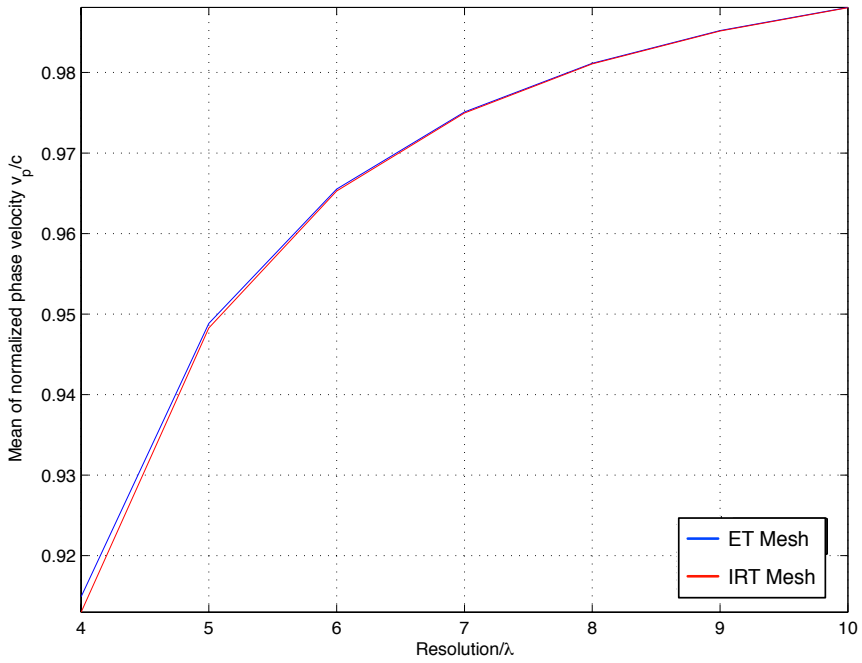


Fig. 11.2 Comparison of phase velocity of the IRT and ET mesh

11.1.2 Calculating Resolution Reduction Factor

Since the calculations of normalised phase velocities for the ET and the IRT mesh have to be carried out separately using procedure described in Section 11.1, two different sets of resolutions $\mathbf{RES_{ET}}$ and $\mathbf{RES_{IRT}}$ are taken for the ET and the IRT meshes respectively.

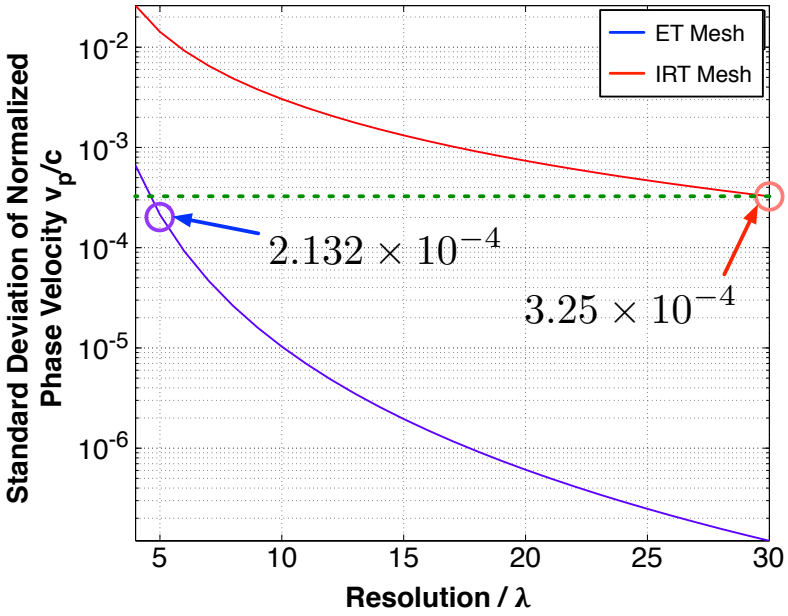
The standard deviation of the normalised phase velocity can be expressed as function of resolution r can by any real number. For IRT and ET mesh the standard deviation function can be $\mathbf{STD}_{IRT}(r)$ and $\mathbf{STD}_{ET}(r)$ respectively.

To calculate the RRF calculated by finding the resolutions r_{IRT} and r_{ET} for IRT and ET meshes with produces almost the same value with $\mathbf{STD}_{IRT}(r)$ and $\mathbf{STD}_{ET}(r)$ respectively.

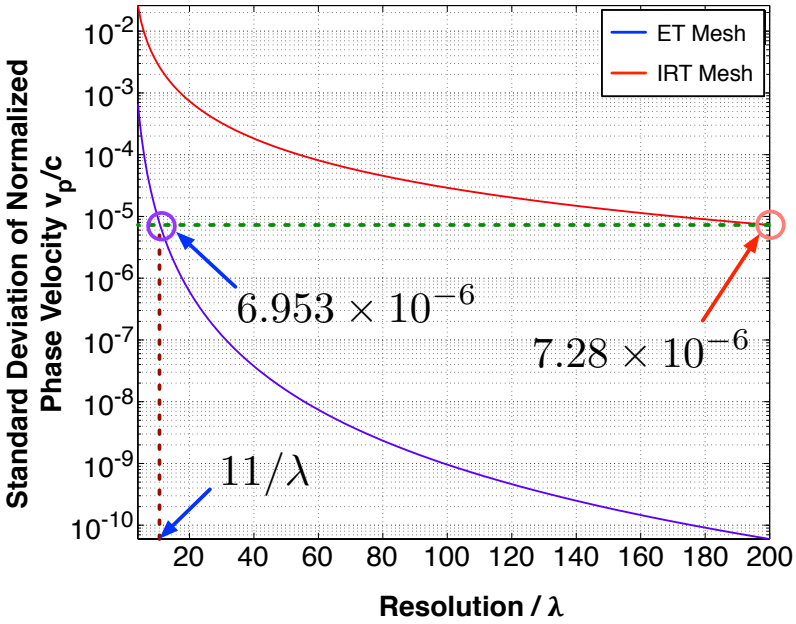
Now the RRF can be calculated by dividing r_{IRT} by r_{ET},

$$RRF(r_{IRT}, r_{ET}) = \frac{r_{IRT}}{r_{ET}} \tag{11.6}$$

An inverse relation can be defined as follows between the RRF and the r_{ET}



(a) Standard deviation of normalised phase velocity from $4/\lambda$ to $30/\lambda$ resolution



(b) Standard deviation of normalised phase velocity from $4/\lambda$ to $200/\lambda$ resolution

Fig. 11.3 Comparison of the numerical dispersion performance of the ET mesh and the IRT mesh.

$$r_{ET}(RRF) = \frac{r_{IRT}}{RRF} \tag{11.7}$$

This procedure was implemented in MATLAB to determine nature of the relationship between the resolution for the ET mesh and the RRF. The relationship between the RRF and the resolution of the ET mesh, $r_{ET}(RRF)$ can be seen in Fig. 11.4 where $r_{ET}(RRF)$ is nearly a linear function. Hence, the RRF improves linearly when the resolution is prograsively increased.

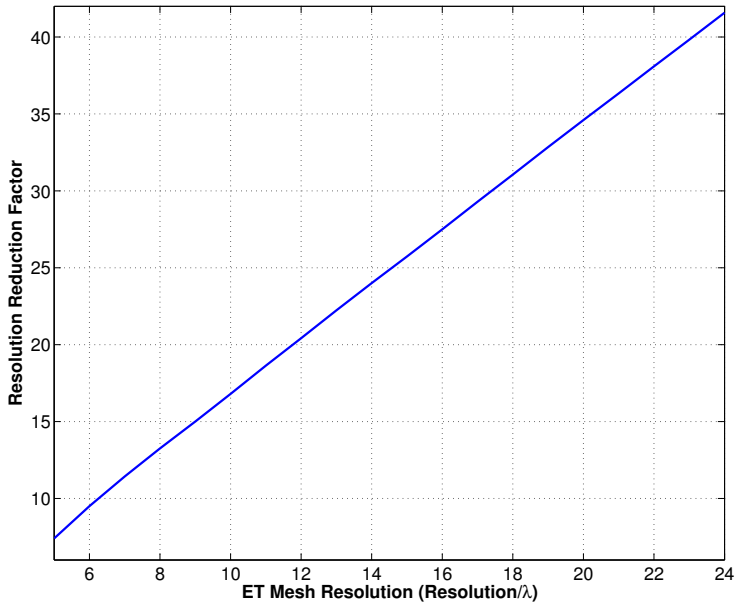


Fig. 11.4 Resolution Reduction Factor vs resolution of the ET mesh, $\mathscr{R}_{ET}$

11.1.3 Comparing Numerical Dispersion of Meshes by Simulation

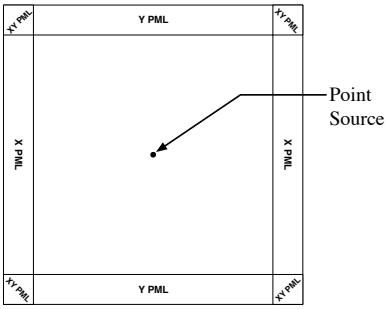
The theoretical analysis of numerical dispersion shown in Sections 11.1, 11.1.1 and 11.1.2 can be verified by running simulations with both type of mesh using the C++ code. To compare the numerical dispersion of both the meshes, a very simple setup was made. A point source was placed at the centre of a square computational domain of free-space ($\mu_r = 1$ and

$\varepsilon_r = 1$). The computational domain was also surrounded by appropriate PML boundaries, as shown in Fig. 11.5a. The point source placed at the centre is an E_z field continuous sine wave source with frequency 1 Hz (normalised). For all the simulations, $c\Delta t/\Delta l = 0.1$ (where, Δt is the length of 1 time step and Δl is the length of both the x and y sides of an element in the IRT mesh and the length of any side of that of the ET mesh) was maintained.

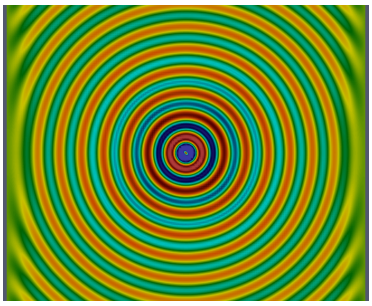
Initially the simulations were performed at a resolution of $10/\lambda$. Figures 11.5b and 11.5c show the E_z field profile after 2000 time steps. At this resolution, the standard deviation of the normalised phase velocity, v_p/c, is 3.041×10^{-3} for the IRT mesh and only 1.032×10^{-5} for the ET mesh (see Fig. 11.3a). Although the standard deviation of v_p/c for the ET mesh is much smaller than that of the IRT mesh, however in this case both the values are small enough to make the E_z field profiles almost identical (to the naked eye) for small computational domains, as shown in Fig. 11.5b and 11.5c.

However, at lower resolution, the effect of the numerical dispersion can be easily visualised on the field profile, even in a small computational domain. Hence, simulations were performed at a resolution of $4/\lambda$. At this resolution, the standard deviation of v_p/c for the IRT mesh is 2.598×10^{-2} and that for the ET mesh is 6.611×10^{-4}. The standard deviation of v_p/c for the ET mesh at this resolution is lower than that of the $10/\lambda$ IRT mesh. So the E_z field profile of the $4/\lambda$ ET mesh and the $10/\lambda$ IRT mesh should be almost identical. As the standard deviation of v_p/c for the IRT mesh at this resolution is higher, the visible distortion must be present in the E_z field profile.

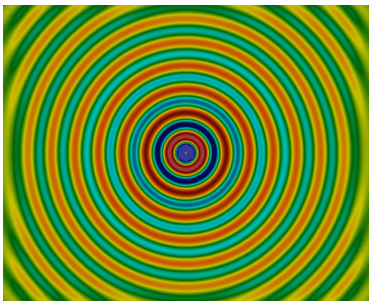
Figures 11.5d and 11.5f show the E_z field profiles obtained after 2000 time steps for the $4/\lambda$ IRT and the $4/\lambda$ ET meshes respectively. As has been discussed in the previous paragraph, the impact of the lower resolution is clearly visible in the field profile presented in Fig. 11.5d. At this resolution with the IRT mesh, the evolution of the E_z field is no longer circular; rather it became somewhat square in profile. This is due to the speed variation in different angle of propagation. As shown in Fig. 11.1(c), v_p/c at 0°, 45°, 90° are 0.8707, 0.9443 and 0.8707, respectively. These data indicate that the speeds of propagation at 0° and 90° will be 7.36% slower than that of the propagation at 45°. The variation in v_p/c is also a continuous function of the propagation angle. As a result, the E_z field profile presented in



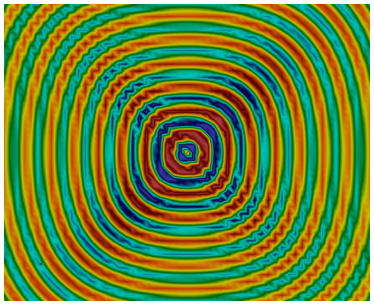
(a) Computational domain with a point source at the center and PML boundaries near the boundary of the domain



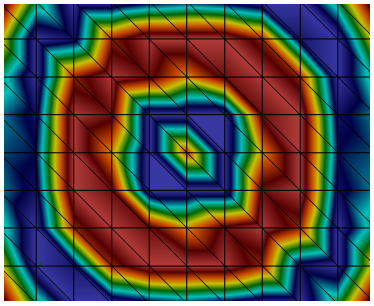
(b) E_z field profile after 2000 time steps with $10/\lambda$ IRT mesh



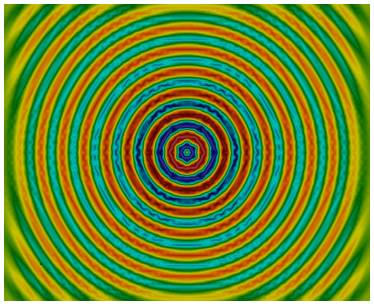
(c) E_z field profile after 2000 time steps with $10/\lambda$ ET mesh



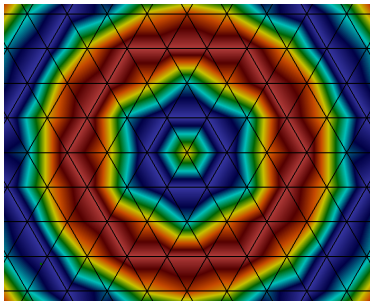
(d) E_z field profile after 2000 time steps with $4/\lambda$ IRT mesh



(e) Magnified view of the central region with mesh overlay of the field shown in Fig. 11.5d



(f) E_z field profile after 2000 time steps with $4/\lambda$ ET mesh



(g) Magnified view of the central region with mesh overlay of the field shown in Fig. 11.5f

Fig. 11.5 Simulation results of the proposed 2D FETD with the IRT and the ET meshes

Fig. 11.5d became a rounded square shape instead of circle, as it should have been in the ideal case.

For the ET mesh at $4/\lambda$ resolution, the highest point of v_p/c in Fig. 11.1(c) is at the 60° angle and the lowest point is at the 30° angle with their values 0.9124 and 0.9105, respectively, Showing a difference of only 0.019%. Again, the variation of v_p/c is a continuous function of the propagation angle. As the difference of speed is much smaller compared to that of the IRT mesh of same resolution, the ET mesh at the $4/\lambda$ resolution retains the near circular shape of propagation of the E_z field profile in Fig. 11.5f.

Results presented in Figs. 11.5d and 11.5f can be further explained by a closer examination of the evolution of field near the point source. Figures 11.5e and 11.5g show the field close to the point source with an overlay of the mesh used during the simulation. It can be seen from Fig. 11.5e that, the points surrounding the point source in the IRT mesh are not equidistant. As a result, the calculated field at the direction of the farthest point moves faster than that of the closest points. As the points closest to the source are at 0° and 90° and the farthest point is at 45°, the maximum v_p/c is found at 45° and minima can be found along angles of 0° and 90°.

However, for the ET mesh, all six points surrounding the source are equidistant, as shown in Fig. 11.5g. The closest point to the source is at 30° at the middle of an edge of the element. The six equidistant points are located at 0°, 60°, 120°, 180°, 240° and 300° and along these angles the maximum values of v_p/c can be found. The minimum values of v_p/c can be interpolated at a point on the outer edge of surrounding elements at 30°, 90°, 150°, 210°, 270° and 330°.

The above discussion highlights the accuracy of the method when used with the ET mesh. Even a low resolution ET mesh of $4/\lambda$ produces an acceptable solution, where the $4/\lambda$ IRT mesh is numerically unusable, even in the smallest possible computational domain.

In a more practical situation use of only ET meshes may not be able to represent the whole device to be analyzed. A small number of other irregular types of elements may have to be introduced into the mesh system to represent arbitrary shape more conveniently. These non ET elements can introduce some additional numerical dispersion into the simulation.

As long as most of the elements are close to equilateral, the overall numerical dispersion of the entire computational domain will remain considerably small.

11.1.4 Comparison with the FDTD Method

To compare the proposed method with the IRT and the ET mesh with regular FDTD, first the numerical dispersion relation for regular FDTD for 2D was considered from [12]. Equations derived for the IRT and ET meshes were compared with the regular FDTD dispersion relation. Then the regular FDTD method was implemented in 2D and a comparison was performed to confirm the derived theory.

To compare the numerical dispersion of the proposed method with that of the more widely used FDTD method, Eq. 11.5 can be further simplified for both the IRT mesh of Fig. 11.1(a) and the ET mesh of Fig. 11.1(b).

For the IRT mesh, the nodal data of Fig. 11.1(a) can be applied to Eq. 11.5 and the equation can be simplified as

$$\begin{aligned}\frac{1}{v_p^2 t^2}\left(e^{j\omega\frac{t}{2}}-e^{-j\omega\frac{t}{2}}\right)^2 &= \frac{1}{a^2}\left[\left(e^{-j\tilde{\kappa}\frac{a}{2}\cos\phi}-e^{j\tilde{\kappa}\frac{a}{2}\cos\phi}\right)\cdot\left(e^{-j\tilde{\kappa}\frac{a}{2}\cos\phi}-e^{j\tilde{\kappa}\frac{a}{2}\cos\phi}\right)\right.\\ &\quad\left.+\left(e^{-j\tilde{\kappa}\frac{a}{2}\sin\phi}-e^{j\tilde{\kappa}\frac{a}{2}\sin\phi}\right)\cdot\left(e^{-j\tilde{\kappa}\frac{a}{2}\sin\phi}-e^{j\tilde{\kappa}\frac{a}{2}\sin\phi}\right)\right]\\ \Rightarrow\left[\frac{a}{v_p t}\sin\left(\frac{\omega t}{2}\right)\right]^2 &= \underline{\left[\sin\left(\frac{\tilde{\kappa}a\cos\phi}{2}\right)\right]^2}+\underline{\left[\sin\left(\frac{\tilde{\kappa}a\sin\phi}{2}\right)\right]^2}\end{aligned}\tag{11.8}$$

Similarly, the nodal data from Fig. 11.1(b) was applied on Eq. 11.5 and simplified as

$$\begin{aligned}\left[\frac{a}{v_p t}\sin\left(\frac{\omega t}{2}\right)\right]^2 &= \underline{\left[\sin\left(\frac{\tilde{\kappa}a\cos\phi}{2}\right)\right]^2}+\underline{\left[0.577\cdot\sin\left(\tilde{\kappa}0.866a\sin\phi\right)\right]^2}\\ &\quad+\underline{\left[0.577\cdot\left(\cos\left(\frac{\tilde{\kappa}a\cos\phi}{2}\right)-\cos\left(\tilde{\kappa}0.866a\sin\phi\right)\right)\right]^2}\end{aligned}\tag{11.9}$$

Two parts of Eq. 11.8 are underlined in red and blue and similarly, three parts of Eq. 11.9 are underlined with red, blue and cyan colors respectively. As can be seen, the part under-

lined with red is common to both the equations. The constants of the blue underlined parts are different in the two equations. The cyan underlined part in Eq. 11.9 is absent in Eq. 11.8. Due to this extra element in Eq. 11.9, the proposed method with the ET mesh shows a more stable solution for all possible angles, compared to Eq. 11.8.

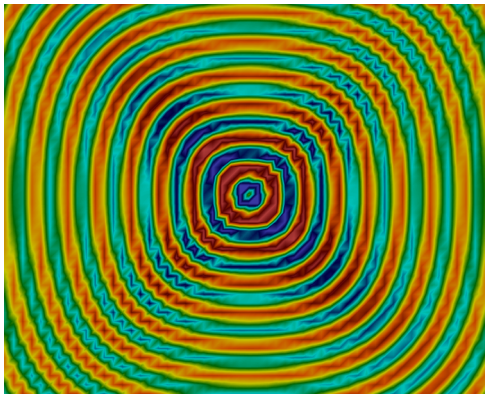


Fig. 11.6 E_z field profile after 2000 time steps with the FDTD method (in 2D)

In the work of Hagness and Taflove [12], the numerical dispersion relation for the FDTD method has been given, which is identical to that the proposed method when used with the IRT mesh, as shown in Eq. 11.8. Therefore, the numerical dispersion characteristics of the FDTD method will be similar to that of the proposed method when used with the IRT mesh.

To prove the similarity of the propagation with IRT mesh and the FDTD method, the FDTD method was implemented and a simulation was performed with a resolution of $4/\lambda$ (keeping all the other parameters same as shown in Section 11.1.3). Figure 11.6 shows the E_z field profile after 2000 time steps. It can be observed that, both Figs. 11.5d and 11.6 show similar rounded square propagation for the FDTD and the proposed method with the IRT mesh respectively. With the ET mesh with the same resolution, the proposed method however, retains the near circular shape in Fig. 11.5f, confirming the superiority of the method proposed here.

11.2 Numerical Dispersion for Three-dimensional Formulation

For convenience of derivation of the numerical dispersion repletion for three-dimensional formulation, Eqs. 7.9 can be rewritten as,

$$\left\{h_{x\langle l\rangle}^{(m)}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^{T} = -\frac{1}{\mu}\left(\left\{e_{zk}^{[n]}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^{T} - \left\{e_{yk}^{[n]}\right\}\left\{\frac{\partial N_k}{\partial z}\right\}^{T}\right) \tag{11.10a}$$

$$\left\{h_{y\langle l\rangle}^{(m)}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^{T} = \frac{1}{\mu}\left(\left\{e_{zk}^{[n]}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^{T} - \left\{e_{xk}^{[n]}\right\}\left\{\frac{\partial N_k}{\partial z}\right\}^{T}\right) \tag{11.10b}$$

$$\left\{h_{z\langle l\rangle}^{(m)}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^{T} = -\frac{1}{\mu}\left(\left\{e_{yk}^{[n]}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^{T} - \left\{e_{xk}^{[n]}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^{T}\right) \tag{11.10c}$$

$$\left\{e_{x\langle k\rangle}^{(n)}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^{T} = \frac{1}{\varepsilon}\left(\left\{h_{zl}^{[m]}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}^{T} - \left\{h_{yl}^{[m]}\right\}\left\{\frac{\partial N_l}{\partial z}\right\}^{T}\right) \tag{11.10d}$$

$$\left\{e_{y\langle k\rangle}^{(n)}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^{T} = -\frac{1}{\varepsilon}\left(\left\{h_{zl}^{[m]}\right\}\left\{\frac{\partial N_l}{\partial x}\right\}^{T} - \left\{h_{xl}^{[m]}\right\}\left\{\frac{\partial N_l}{\partial z}\right\}^{T}\right) \tag{11.10e}$$

$$\left\{e_{z\langle k\rangle}^{(n)}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^{T} = \frac{1}{\varepsilon}\left(\left\{h_{yl}^{[m]}\right\}\left\{\frac{\partial N_l}{\partial x}\right\}^{T} - \left\{h_{xl}^{[m]}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}^{T}\right) \tag{11.10f}$$

In a similar way to the 2D numerical dispersion calculation in Section 11.1 - Eqs. 11.1, in Eqs. 11.10, the field components for the space node of the main and the auxiliary meshes are denoted with k and l subscripts, respectively. The field components for the members of **M** and **N** are denoted with (m) and (n) superscripts, respectively. The angle brackets $\langle\ \rangle$ are used to denote the centroid of the current element and the square brackets $[\ ]$ are used to denote the current time.

A monochromatic source was assumed for the TE mode of propagation where **E** and **H** field components can be expressed as

$$e^{(n)}_{x\langle k\rangle} = E_{x0}e^{j(\omega t^{(n)}-\tilde{\kappa}_x x_k-\tilde{\kappa}_y y_k-\tilde{\kappa}_z z_k)} \tag{11.11a}$$

$$e^{(n)}_{y\langle k\rangle} = E_{y0}e^{j(\omega t^{(n)}-\tilde{\kappa}_x x_k-\tilde{\kappa}_y y_k-\tilde{\kappa}_z z_k)} \tag{11.11b}$$

$$e^{(n)}_{z\langle k\rangle} = E_{z0}e^{j(\omega t^{(n)}-\tilde{\kappa}_x x_k-\tilde{\kappa}_y y_k-\tilde{\kappa}_z z_k)} \tag{11.11c}$$

$$h^{(m)}_{x\langle l\rangle} = H_{x0}e^{j(\omega t^{(m)}-\tilde{\kappa}_x x_l-\tilde{\kappa}_y y_l-\tilde{\kappa}_z z_l)} \tag{11.11d}$$

$$h^{(m)}_{y\langle l\rangle} = H_{y0}e^{j(\omega t^{(m)}-\tilde{\kappa}_x x_l-\tilde{\kappa}_y y_l-\tilde{\kappa}_z z_l)} \tag{11.11e}$$

$$h^{(m)}_{z\langle l\rangle} = H_{z0}e^{j(\omega t^{(m)}-\tilde{\kappa}_x x_l-\tilde{\kappa}_y y_l-\tilde{\kappa}_z z_l)} \tag{11.11f}$$

where, $\underline{\tilde{\kappa}} = \hat{x}\tilde{\kappa}_x + \hat{y}\tilde{\kappa}_y + \hat{z}\tilde{\kappa}_z$ is the numerical wave vector, ω is the frequency of the source and E_{x0}, E_{y0}, E_{z0}, H_{x0}, H_{y0} and H_{z0} are the amplitudes of the E_x, E_y, E_z, H_x, H_y and H_z field components, respectively.

Applying Eqs. 11.11 on Eq. 11.10a and Eq. 11.10b the expression of H_{x0}, H_{y0}, H_{x0}, E_{x0}, E_{y0} and E_{z0} can be obtained as,

$$H_{x0} = -\frac{1}{\mu}\cdot\frac{\left(E_{z0}\left\{e^{-j\underline{\tilde{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^T - E_{y0}\left\{e^{-j\underline{\tilde{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial z}\right\}^T\right)}{\left\{e^{j\omega\Delta t^{(m)}}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^T} \tag{11.12a}$$

$$H_{y0} = \frac{1}{\mu}\cdot\frac{\left(E_{z0}\left\{e^{-j\underline{\tilde{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^T - E_{x0}\left\{e^{-j\underline{\tilde{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial z}\right\}^T\right)}{\left\{e^{j\omega\Delta t^{(m)}}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^T} \tag{11.12b}$$

$$H_{z0} = -\frac{1}{\mu}\cdot\frac{\left(E_{y0}\left\{e^{-j\underline{\tilde{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^T - E_{x0}\left\{e^{-j\underline{\tilde{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^T\right)}{\left\{e^{j\omega\Delta t^{(m)}}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^T} \tag{11.12c}$$

$$E_{x0} = \frac{1}{\varepsilon}\cdot\frac{\left(H_{z0}\left\{e^{-j\underline{\tilde{\kappa}}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}^T - H_{y0}\left\{e^{-j\underline{\tilde{\kappa}}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial z}\right\}^T\right)}{\left\{e^{j\omega\Delta t^{(n)}}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^T} \tag{11.12d}$$

$$E_{y0} = -\frac{1}{\varepsilon} \cdot \frac{\left(H_{z0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial x} \right\}^T - H_{x0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial z} \right\}^T \right)}{\left\{ e^{j\omega \Delta t^{(n)}} \right\} \left\{ \frac{dQ^{(n)}}{dt} \right\}^T} \tag{11.12e}$$

$$E_{z0} = \frac{1}{\varepsilon} \cdot \frac{\left(H_{y0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial x} \right\}^T - H_{x0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial y} \right\}^T \right)}{\left\{ e^{j\omega \Delta t^{(n)}} \right\} \left\{ \frac{dQ^{(n)}}{dt} \right\}^T} \tag{11.12f}$$

Here, $\Delta x_{k(i)} = x_{k(i)} - x_{\langle l \rangle}$, $\Delta y_{k(i)} = y_{k(i)} - y_{\langle l \rangle}$, $\Delta z_{k(i)} = z_{k(i)} - z_{\langle l \rangle}$, $\Delta \mathbf{r}_k = (\Delta x_k, \Delta y_k, \Delta z_k)$, $\Delta x_{l(i)} = x_{l(i)} - x_{\langle k \rangle}$, $\Delta y_{l(i)} = y_{l(i)} - y_{\langle k \rangle}$, $\Delta z_{l(i)} = z_{l(i)} - z_{\langle k \rangle}$, $\Delta \mathbf{r}_l = (\Delta x_l, \Delta y_l, \Delta z_l)$, and $\Delta t_\tau^{(m)} = t_\tau^{(m)} - t^{[n]}$, $\Delta t_\tau^{(n)} = t_\tau^{(n)} - t^{[m]}$ and $\underline{\tilde{\kappa}} = (\tilde{\kappa}_x, \tilde{\kappa}_y, \tilde{\kappa}_z)$.

Replacing the value of H_{x0} and H_{y0} from Eqs. 11.12a and 11.12b in Eq. 11.12f,

$$\begin{aligned}
&E_{z0} \cdot \left\{ e^{j\omega \Delta t^{(n)}} \right\} \left\{ \frac{dQ^{(n)}}{dt} \right\}^T \cdot \left\{ e^{j\omega \Delta t^{(m)}} \right\} \left\{ \frac{dQ^{(m)}}{dt} \right\}^T = \\
&v_p^2 \cdot \Bigg[E_{z0} \Bigg(\left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_k} \right\} \left\{ \frac{\partial N_k}{\partial x} \right\}^T \cdot \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial x} \right\}^T + \\
&\left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_k} \right\} \left\{ \frac{\partial N_k}{\partial y} \right\}^T \cdot \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial y} \right\}^T \Bigg) - \\
&\Bigg(E_{x0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial x} \right\}^T + \\
&E_{y0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial y} \right\}^T \Bigg) \cdot \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_k} \right\} \left\{ \frac{\partial N_k}{\partial z} \right\}^T \Bigg]
\end{aligned} \tag{11.13}$$

Here, $v_p = \frac{1}{\sqrt{\mu \varepsilon}}$.

Multiplying $\left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial x} \right\}^T$ with Eq. 11.12d and $\left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial y} \right\}^T$ with Eq. 11.12e and then adding them the following expression can be obtained.

$$\begin{aligned}
&E_{x0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial x} \right\}^T + \\
&E_{y0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial y} \right\}^T = -E_{z0} \left\{ e^{-j\underline{\tilde{\kappa}} \cdot \Delta \mathbf{r}_l} \right\} \left\{ \frac{\partial N_l}{\partial z} \right\}^T
\end{aligned} \tag{11.14}$$

Applying Eq. 11.14 into Eq. 11.13 the numerical dispersion relation for the 3D formulation of the proposed FETD can be obtained,

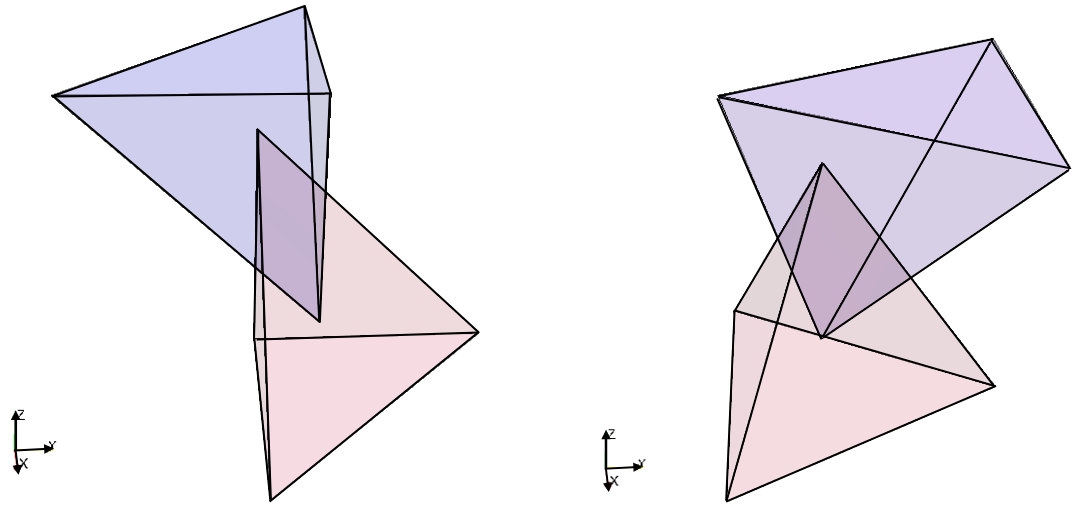
$$\begin{aligned}
&\left\{e^{j\omega\Delta t^{(n)}}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^T\cdot\left\{e^{j\omega\Delta t^{(m)}}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^T = \\
&v_p^2\cdot\left[\left\{e^{-j\tilde{\underline{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^T\cdot\left\{e^{-j\tilde{\underline{\kappa}}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial x}\right\}^T\right. \\
&+\left\{e^{-j\tilde{\underline{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^T\cdot\left\{e^{-j\tilde{\underline{\kappa}}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}^T \\
&\left.+\left\{e^{-j\tilde{\underline{\kappa}}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial z}\right\}^T\cdot\left\{e^{-j\tilde{\underline{\kappa}}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial z}\right\}^T\right]
\end{aligned} \tag{11.15}$$

Similar to Section 11.1 for omnidirectional propagation in an isotropic medium, Eq. 11.15 can be written as

$$\begin{aligned}
&\left\{e^{j\omega\Delta t^{(n)}}\right\}\left\{\frac{dQ^{(n)}}{dt}\right\}^T\cdot\left\{e^{j\omega\Delta t^{(m)}}\right\}\left\{\frac{dQ^{(m)}}{dt}\right\}^T = \\
&v_p^2\cdot\left[\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}^T\cdot\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial x}\right\}^T\right. \\
&+\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}^T\cdot\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}^T \\
&\left.+\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial z}\right\}^T\cdot\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial z}\right\}^T\right]
\end{aligned} \tag{11.16}$$

Here, $\tilde{\underline{\kappa}} = (\tilde{\kappa}_x,\, \tilde{\kappa}_y,\, \tilde{\kappa}_z) = (\tilde{\kappa}\cos\theta\sin\phi,\, \tilde{\kappa}\sin\theta\sin\phi,\, \tilde{\kappa}\cos\phi) = \tilde{\kappa}\cdot(\cos\theta\cdot\sin\phi,\, \sin\theta\cdot\sin\phi,\, \cos\phi) = \tilde{\kappa}\cdot\underline{\Theta}$ and $\underline{\Theta} = (\cos\theta\cdot\sin\phi,\, \sin\theta\cdot\sin\phi,\, \cos\phi)$.

In a way similar to Section 11.1, Eq. 11.16 can be used with Newton's iterative method to obtain the numerical wave vector $\tilde{\kappa}$. The normalised propagation velocity, $v_p/c = 2\pi/\tilde{\kappa}_{final}$ can be calculated using the final converged value for specific angles of propagation θ and ϕ.



(a) Two elements of the coupled the IRT3D mesh

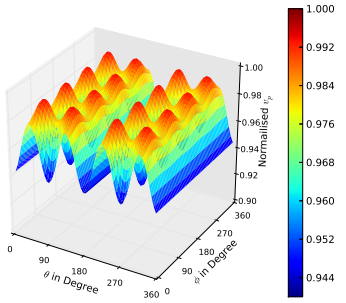
(b) Two elements of the coupled the ET3D mesh

Fig. 11.7 Two coupled element of the IRT3D and ET3D mesh systems, respectively. The element from the main mesh is shown in red and the element from the auxiliary mesh is shown in blue colours, respectively

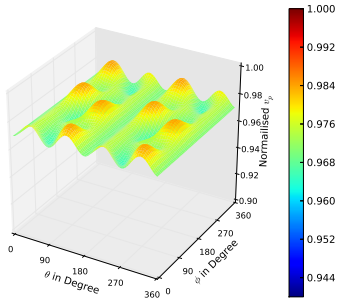
11.2.1 Calculation of Numerical Dispersion

To measure the numerical dispersion for the 3D formulation of the proposed FETD method, Newton's iterative method was implemented in Python using Eq. 11.16. The code can be used to measure the numerical dispersion for any type of tetrahedron. To investigate the numerical dispersion performance, the isosceles right angled tetrahedral mesh (introduced in Section 8.1) was used. To compare the performance of the IRT3D mesh with **"Equilateral Tetrahedral" (ET3D)** mesh, the code was used to calculated the numerical dispersion performance of the the ET3D mesh. Figure 11.7 shows the coupled mesh arrangement for the IRT3D and the ET3D meshes, respectively. In both the Figs. 11.7a and 11.7b, the element of the main mesh is shown in red while the element from the auxiliary mesh is shown in blue colours, respectively.

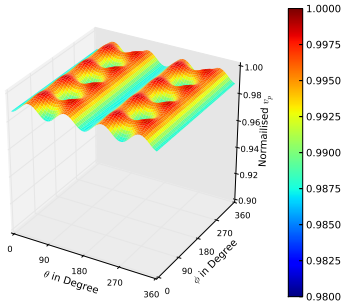
Figure 11.8 presents a side by side comparison of normalised phase velocity v_p distribution for the IRT3D and the ET3D meshes for $5/\lambda$, $10/\lambda$ and $30/\lambda$, respectively. It can be noted that the Figs. 11.8a, 11.8c and 11.8e are showing the normalised $v_p(\theta,\phi)$ distributions of the IRT3D mesh for the resolutions $5/\lambda$, $10/\lambda$ and $30/\lambda$, respectively. Fig-



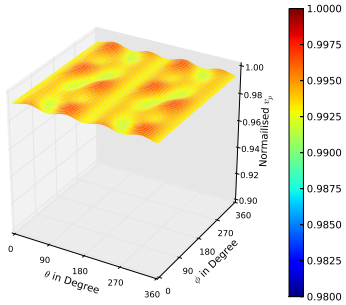
(a) Normalised $v_p(\theta,\phi)$ distribution for $5/\lambda$ resolution for the IRT3D mesh



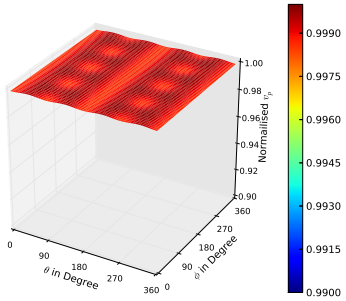
(b) Normalised $v_p(\theta,\phi)$ distribution for $5/\lambda$ resolution for the ET3D mesh



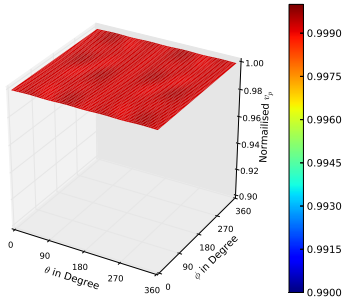
(c) Normalised $v_p(\theta,\phi)$ distribution for $10/\lambda$ resolution for the IRT3D mesh



(d) Normalised $v_p(\theta,\phi)$ distribution for $10/\lambda$ resolution for the ET3D mesh



(e) Normalised $v_p(\theta,\phi)$ distribution for $30/\lambda$ resolution for the IRT3D mesh



(f) Normalised $v_p(\theta,\phi)$ distribution for $30/\lambda$ resolution for the ET3D mesh

Fig. 11.8 Side by side comparison of normalised v_p of the IRT3D and the ET3D meshes for resolution $5/\lambda$, $10/\lambda$ and $30/\lambda$, respectively

ures. 11.8b, 11.8d and 11.8f are showing the normalised $v_p(\theta, \phi)$ distributions of the ET3D mesh for the resolutions $5/\lambda$, $10/\lambda$ and $30/\lambda$, respectively.

It can also be seen from figure pairs (11.8a, 11.8b), (11.8c, 11.8d) and (11.8e, 11.8f) that each of the pairs are plotted with the same scale in amplitude. Which also kept the colour scheme of the surface plot to be the same. All the figure pairs presented in Fig. 11.8 shows that for all resolution the variation of normalised v_p in ET3D mesh is lower than the IRT3D mesh. Hence, the numerical dispersion of the ET3D mesh is lower that the IRT3D mesh for all resolution. This result is consistent with the finding in Section 11.1.1 for the 2D formulation.

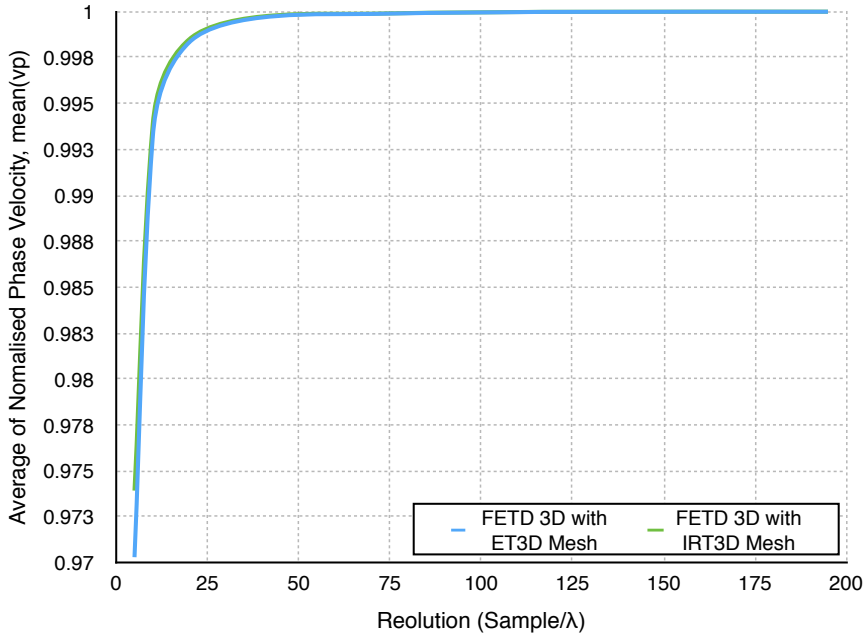
To measure the improvement of numerical dispersion with ET3D mesh over the normalised v_p distribution was calculated with the Python code for resolutions from $5/\lambda$ to $200/\lambda$ for both ET3D and IRT3D meshes. Mean and the standard deviation for each resolution was calculated. Figure 11.9 shows the comparison between of the mean and Standard deviation of ET3D and IRT3D meshes.

As it can be seen in Fig. 11.9a, the mean of the normalised v_p is almost the same (for ET3D mesh, the mean is slightly higher than the IRT3D mesh). But the standard deviation of for the two meshes in Fig. 11.9b is quite different. The standard deviation of the normalised v_p for the ET3D mesh is significantly lower than the same for the IRT3D mesh. Which suggest that although the average propagation speed on both the meshes will be almost the same, the directional variation of the ET3D mesh is significantly less than the IRT3D mesh. In other words, the numerical dispersion of the ET3D mesh is significantly lower than the IRT3D mesh. This result also agrees with the analysis in Section 11.1.2 for 2D formulation.

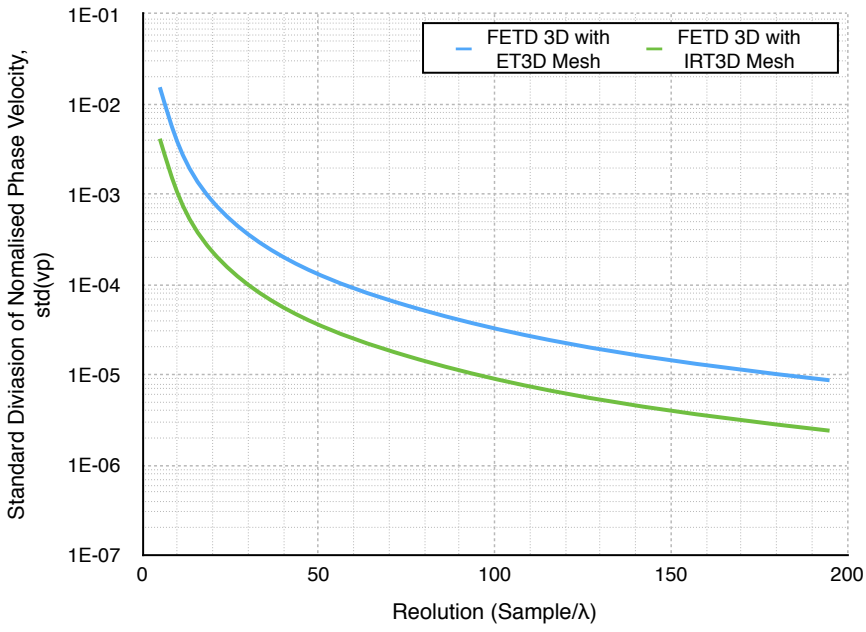
As the numerical dispersion of the ET3D mesh is significantly lower than the IRT3D mesh the resolution reduction factor introduced in Section 11.1.2 can be calculated for the 3D formulation in a similar manner.

11.2.2 Calculation of Resolution Reduction Factor

To calculate the resolution reduction factor for IRT3D and ET3D mesh, first the functional relation between the resolutions of IRT3D and ET3D mesh should be evaluated. To find



(a) Comparison of the mean of normalised v_p for ET3D and IRT3D meshes



(b) Comparison of the standard deviation of normalised v_p for ET3D and IRT3D meshes

Fig. 11.9 Mean and Standard Deviation of the ET3D and the IRT3D meshes from $5/\lambda$ to $200/\lambda$ resolutions

the relation of the resolutions of the IRT3D mesh and the ET3D mesh, the resolutions of each mesh time can be described as two sets $\mathbf{RES_{IRT3D}}$ and $\mathbf{RES_{ET3D}}$ respectively, in a way similar to Section 11.1.2, RRF relation for 3D can be derived.

Figure 11.10 shows the relation between the ET3D and IRT3D mesh for the 3D case. As can be seen, the relation is a linear function.

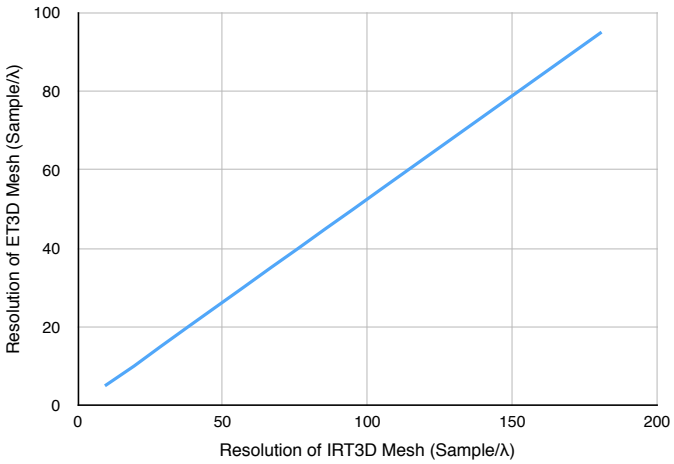


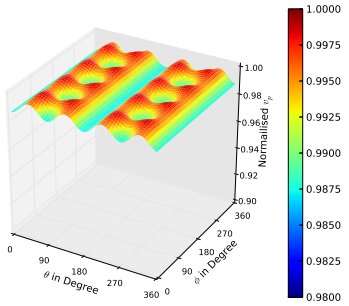
Fig. 11.10 Resolution Relation of the ET3D and the IRT3D meshes for resolution $5/\lambda$ to $500/\lambda$

Therefore, **"Resolution Reduction Factor (RRF)"** introduced in Section 11.1.2, Eq. 11.6 is a constant for all resolutions when IRT3D and ET3D meshes are considered. From the line in Fig 11.10 the **RRF** was calculated to be **1.903** for all the resolutions.

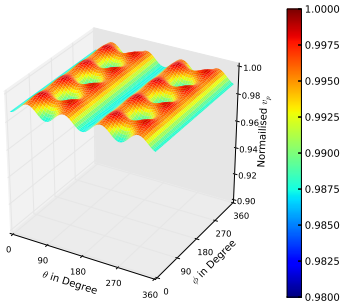
11.2.3 Comparison with the FDTD Method

To compare the numerical dispersion performance of the proposed FETD method with the FDTD method, the numerical dispersion relation for the 3D FDTD method was considered. Equation 11.17 [12, 25] shows the numerical dispersion relation for the FDTD method.

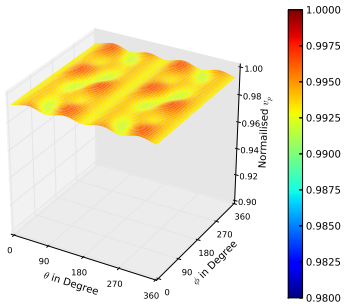
$$\left[\frac{1}{\Delta t}\sin\left(\frac{\omega\Delta t}{2}\right)\right]^2 = \left[\frac{1}{\Delta x}\sin\left(\frac{\tilde{k}_x\Delta x}{2}\right)\right]^2 + \left[\frac{1}{\Delta y}\sin\left(\frac{\tilde{k}_y\Delta y}{2}\right)\right]^2 + \left[\frac{1}{\Delta z}\sin\left(\frac{\tilde{k}_z\Delta z}{2}\right)\right]^2 \tag{11.17}$$



(a) Normalised $v_p(\theta,\phi)$ distribution for the 3D FDTD Method



(b) Normalised $v_p(\theta,\phi)$ distribution for the proposed 3D FETD method with IRT3D mesh



(c) Normalised $v_p(\theta,\phi)$ distribution for the proposed 3D FETD method with ET3D mesh

Fig. 11.11 Side by side comparison of normalised v_p of the 3D FDTD and the proposed 3D FETD scheme with the IRT3D and the ET3D meshes at $10/\lambda$ resolution

Here, $\tilde{k} = \hat{x}\tilde{k}_x + \hat{y}\tilde{k}_y + \hat{z}\tilde{k}_z$ is the numerical wave vector, Δx, Δy and Δz are the divisions in x, y and z directions, respectively.

Similar to Sections 11.1, 11.2 and [12, 25], a iterative code was generated in Python to calculate the normalised phase velocity for different resolution. The result obtained from the Python code for the 3D FDTD method was compared with the result obtained in Section 11.2.1 for the IRT3D mesh and the ET3D mesh.

Figure 11.11c presents the normalised phase velocity distribution at $10/\lambda$ for the 3D FDTD method, the proposed FETD3D method with IRT3D mesh and the propose FETD3D method with ET3D mesh, respectively.

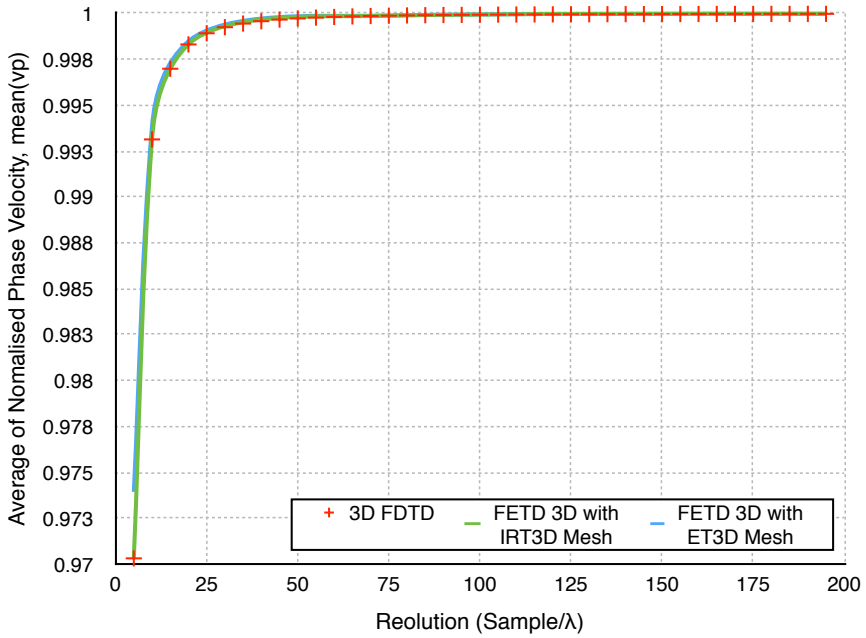
It should be noted that the normalised v_p distribution for the 3D FDTD (Fig. 11.11a)

and proposed method when used with the IRT3D mesh (Fig. 11.11b) are exactly the same. But the normalised v_p distribution for proposed 3D FETD with ET3D mesh is more stable (Fig. 11.11c) than the previous two. Therefore, the numerical dispersion of the 3D FDTD method and the proposed FETD method with IRT3D mesh is exactly the same and the numerical dispersion of the proposed method with ET3D mesh is better than the previous two.

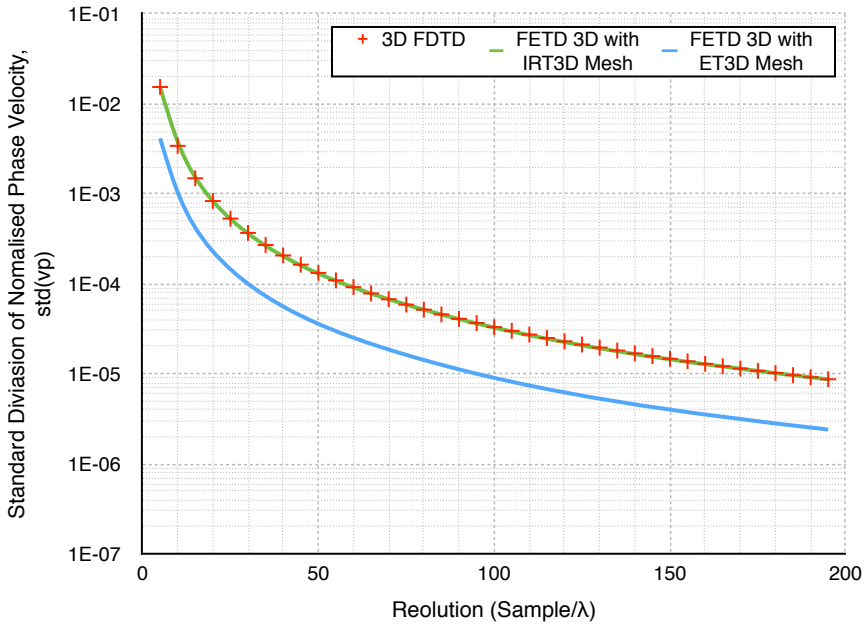
To investigate it farther, the Python program for normalised v_p calculation was used to generate mean and standard divination of the normalised v_p for resolutions from $5/\lambda$ to $200/\lambda$ to compare the results with the results presented in Fig. 11.9. The results of the comparison is presented in Fig. 11.12.

In Fig. 11.12, the FDTD results for mean and standard deviation of the normalised v_p are plotted with "+" (Red Coloured Plus) symbols. It can be noted in Figs. 11.12a and 11.12b, the mean and standard deviation of the 3D FDTD method and the proposed 3D FETD with IRT3D mesh coincides with each other, although the mean of normalised v_p of the proposed method with ET3D mesh shows a slight improvement, but the standard deviation of normalised v_p of the proposed method with ET3D mesh shows significant improvement compared to the 3D FDTD method.

As the standard deviation of the 3D FDTD and the proposed method with IRT3D mesh are same for all resolution, the RRF for the proposed method with ET3D mesh when compared with the 3D FDTD with cubic grid is **1.903** as it was found in Section 11.2.2.



(a) Comparison of the mean of normalised v_p for 3D FDTD method and proposed FETD3D method with IRT3D and ET3D meshes



(b) Comparison of the standard deviation of normalised v_p for 3D FDTD method and proposed FETD3D method with IRT3D and ET3D meshes

Fig. 11.12 Mean and Standard Deviation of the 3D FDTD method and the proposed 3D FETD with the IRT3D and the ET3D meshes for $5/\lambda$ to $200/\lambda$ resolutions

Chapter 12

Theoretical CPU Performance

For any numerical method speed of execution is a crucial matter. The speed of design, development and analysis using the method depends on the speed on execution. For a time domain analysis tool like the FDTD [14], the proposed FETD method, the time domain methods discussed in [17, 27, 33–40] and all other available theoretically and commercially, CPU performance is one of the most important factor that directly relates to its success.

The reason the FDTD method introduced in 1966, for which it is still the most dominant method for time domain electromagnetics, is the FDTD method is designed to be fastest in execution time for each Yee's Cell. No other method introduced after FDTD could outperform it. The time domain analysis is a time consuming task as for each time-step, the entire computational domain has to be evaluated using the governing equations of the method in use. Usually the time-step size is very small compared to the duration of evaluation. Therefore, a method slower in calculating each time-step will slow down the entire simulation significantly even if the speed difference is very small; *i.e.,* if a method takes 1 min to calculate 1 time-step and another method takes 1.1 mins and if both method needs to calculate 1000 time-steps, the first method will compete the entire simulation in 1000 mins and the second method will require 1100 mins. The difference is 100 mins, which is a significant issue.

As a result, a theoretical speed comparison with the FDTD method is required to demonstrate the usefulness of the software as a design tool and also check the commercial prospect

of the proposed method.

In this chapter the CPU performance comparison is carried out considering Equilateral mesh (both 2D and 3D) for the proposed method, square (for 2D) and cubic (for 3D) grid for the FDTD method, resolution reduction factor (RRF) (Sections 11.1.2 and 11.2.2), CPU optimised formulation and Intel Haswell CPU general purpose instructions and Single Instruction Multiple Data (SIMD) instructions sets [85] for both two-dimensions and three-dimensions.

The Equilateral meshes were considered because, it improves the numerical performance of the mesh much better than the FDTD methods in both two and three-dimensions. Therefore, the equivalent resolution for the proposed method with equilateral meshes are much lower than that of the FDTD. Which means similar output can be obtained from a much lower resolution. The equivalent resolution for the proposed method can be found by dividing the FDTD resolution with the RRF of that resolution (discussed in Sections 11.1.2, 11.1.4, 11.2.2 and 11.2.3).

To compare the proposed FETD method with the FDTD in Sections 11.1.4 and 11.2.3, the square and cubic grids, respectively were considered to compare the numerical dispersion performance. To maintain the continuity of the analysis and to consider the numerical advantage shown in Chapter 11, the square (for 2D) and cubic (for 3D) are also used in this chapter for the FDTD method. Besides, the continuity of the analysis the square and cubic grid allows maximum CPU optimisation of the governing equations.

The CPU optimised formulation for both the FDTD and the proposed FETD method were used to have a fair comparison of performance. For CPU optimisation all repeated calculations were performed prior to the execution of the governing equations and stored in memory. Thereby reducing the number of operation required to minimum for both methods.

The Intel instruction sets were used because the Intel CPU's are the most widely used CPUs in desktops, workstations and even in supercomputers (427 out of top 500 supercomputers used Intel CPUs) [86] these days.

12.1 CPU Performance for Two-dimensionals Formulation

To compare the CPU performance of the FDTD and the proposed FETD method in two-dimensions the governing equations for the 2D formulation has to be optimised for the CPU for both the 2D FDTD method and proposed 2D FETD method. In this section only the governing equations in Eqs. 2.7 for TE wave will be taken for derivation and calculation of the CPU optimised formulation because, the TE and TM (Eqs. 2.7 and 2.8, respectively) governing equations are disjoint and similar steps are required to find the CPU optimised formulation of the TM governing equations.

CPU optimised version of a governing equation will always try to reduce CPU time by avoiding costly CPU operations *i.e.* division, multiplication etc. as much as possible. Costly operations will only be considered when there is no other way to reduce the number of CPU cycles (Latency).

This chapter also utilised the Single Instruction Multiple Data (SIMD) instructions provided in the Intel 64-bit x86 Haswell microprocessors. SMID instructions are special instructions developed by CPU Intel which performs the same operation on multiple sets of data with the same latency of a general purpose instruction (applies it on only one set of data). For example, general purpose ADD operation could add a pair of integers in 3 CPU cycles. On the other hand the SIMD ADD operation could perform the same operation on 8 pairs of integers with the same 3 CPU cycles on a Haswell chip. This is advantageous for a computationally heavy method like the proposed method as the SIMD instructions allow further parallelisation of the method in sub equation level. SIMD instructions are widely used for performance enhancement of computationally heavy algorithms [87–93].

12.1.1 CPU Optimised Formulation of the FDTD in Two-dimensions

The discretised form TE propagation governing equations for the FDTD method in 2D has been presented in [12, 25] as,

$$H_x|_{i,j+1/2}^{n+1/2} = -\frac{\Delta t}{\mu_{i,j+1/2}}\left[\frac{E_z|_{i,j+1}^{n} - E_z|_{i,j}^{n}}{\Delta y}\right] + H_x|_{i,j+1/2}^{n-1/2} \tag{12.1a}$$

$$H_y|_{i+1/2,j}^{n+1/2} = \frac{\Delta t}{\mu_{i+1/2,j}}\left[\frac{E_z|_{i+1,j}^{n} - E_z|_{i,j}^{n}}{\Delta x}\right] + H_y|_{i+1/2,j}^{n-1/2} \tag{12.1b}$$

$$E_z|_{i,j}^{n+1} = \frac{\Delta t}{\varepsilon_{i,j}}\left[\frac{H_y|_{i+1/2,j}^{n+1/2} - H_y|_{i-1/2,j}^{n+1/2}}{\Delta x} - \frac{H_x|_{i,j+1/2}^{n+1/2} - H_x|_{i,j-1/2}^{n+1/2}}{\Delta y}\right] + E_z|_{i,j}^{n-1} \tag{12.1c}$$

As the square grid is considered for the 2D formulation, the CPU optimal 2D formulation of the TE governing equations can be written as,

$$H_x|_{i,j+1/2}^{n+1/2} = \mathbb{A}_{i,j+1/2}\left[E_z|_{i,j}^{n} - E_z|_{i,j+1}^{n}\right] + H_x|_{i,j+1/2}^{n-1/2} \tag{12.2a}$$

$$H_y|_{i+1/2,j}^{n+1/2} = \mathbb{A}_{i+1/2,j}\left[E_z|_{i+1,j}^{n} - E_z|_{i,j}^{n}\right] + H_y|_{i+1/2,j}^{n-1/2} \tag{12.2b}$$

$$E_z|_{i,j}^{n+1} = \mathbb{B}_{i,j}\left[\left(H_y|_{i+1/2,j}^{n+1/2} - H_y|_{i-1/2,j}^{n+1/2}\right) - \left(H_x|_{i,j+1/2}^{n+1/2} - H_x|_{i,j-1/2}^{n+1/2}\right)\right] + E_z|_{i,j}^{n-1} \tag{12.2c}$$

Here, $\Delta x = \Delta y = \Delta$, $\mathbb{A}_{i,j} = \frac{\Delta t}{\Delta \cdot \mu_{i,j}}$ and $\mathbb{B}_{i,j} = \frac{\Delta t}{\Delta \cdot \varepsilon_{i,j}}$.

Equations 12.2 do not perform any time consuming division operations. It should be noted that Eqs. 12.2 avoid performing any repetitive calculation by storing $\mathbb{A}_{i,j}$ and $\mathbb{B}_{i,j}$ into the memory. All the equations are composed of addition, subtraction and multiplication operations which are less time consuming. Therefore, the formulation is CPU optimised. Similar optimisation can be performed on Eqs. 2.8 to obtain the CPU optimised form.

Table 12.1 Compute Operations and Latencies for 2D FDTD Method with General Purpose Instructions

Equ.	**Add**		**Sub**		**Mult**		**Total**	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
12.2a	1	3	1	3	1	5	3	11
12.2b	1	3	1	3	1	5	3	11
12.2c	1	3	3	3	1	5	5	17
Total for $\mathbf{E} \rightarrow \mathbf{H}$:							**6**	**22**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:							**5**	**17**
Total for One Time-step:							**11**	**39**

Latency for General Purpose Instructions

Table 12.1 shows the number of instructions required for each equations in Eqs. 12.2 when only the general purpose instructions are used for calculation. The table also list the latency (no. of clock cycles) required for each and operation and show the theoretical minimum latency required by each equation for execution. The total latency shows the time required for calculating one time-step on a single Yee's cell.

It can be noticed that the FDTD method in 2D requires only 3 general purpose instructions with a total latency of 11 cycles for both Eq. 12.2a and 12.2b. Total Eq. 12.2c requires 5 instructions and the minimum latency is 17 cycles.

Latency for General Purpose and SIMD Instructions

Table 12.2 presents the performance of the 2D FDTD method when SIMD instructions are used alongside the general purpose instruction set. In can be noticed that only two subtraction can be replaced with one SIMD subtraction in Eq. 12.2c. There is no change in Eqs. 12.2a and 12.2b as they only use one instance of addition, subtraction and multiplication. Therefore, execution latency for Eq. 12.2c is reduced by only 3 cycles.

Table 12.2 Compute Operations and Latencies for 2D FDTD Method with General Purpose and SIMD Instructions

Equ.	**Add**		**Sub**		**SIMD Sub**		**Mult**		**Total**	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
12.2a	1	3	1	3	0	3	1	5	3	11
12.2b	1	3	1	3	0	3	1	5	3	11
12.2c	1	3	1	3	1	3	1	5	4	14
Total for $\mathbf{E} \rightarrow \mathbf{H}$:									**6**	**22**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:									**4**	**14**
Total for One Time-step:									**10**	**36**

12.1.2 CPU Optimised Formulation for Proposed FETD in Two-dimensions

The discretised form TE propagation governing equations for the proposed FETD method in 2D has been presented in Eqs. 2.13. Although only the equilateral mesh is considered, no optimisation is performed for the specific shape of the elements. Because, the reason of using FE based method is to have the flexibility to modify the shape of the element to have accurate representation of the structure. Therefore, the optimisation will only consider the minimisation of the CPU instructions by pre-calculating and storing repeated operations. Similar to Section 12.1.1. Although the formulation in Eqs. 2.13 allows unequal division of time-step, for optimisation calculation equal division of time-step is considered. The reasons are, firstly, the FDTD method always carried out equal division for **E** field and **H** field calculation. To have a fair comparison for the space discretisation, it is assumed that the time discretisation should remain the same. Therefore to consider the output presented in Section 11.1.1 and 11.1.2 for CPU performance calculation equal division is required.

The CPU optimal formulation for proposed FETD method in 2D can be represented as,

$$h_x^{(n+1)} = \mathbb{A}\left(\sum_{i=1}^{3} \frac{\partial N_i}{\partial y} e_{zi}^{(n)}\right) + h_x^{(n-1)} \tag{12.3a}$$

$$h_y^{(n+1)} = \mathbb{B}\left(\sum_{i=1}^{3} \frac{\partial N_i}{\partial x} e_{zi}^{(n)}\right) + h_y^{(n-1)} \tag{12.3b}$$

$$e_z^{(n+1)} = \mathbb{C}\left(\sum_{i=1}^{3} \frac{\partial N_i}{\partial x} h_{yi}^{(n)} - \sum_{i=1}^{3} \frac{\partial N_i}{\partial y} h_{xi}^{(n)}\right) + e_z^{(n-1)} \tag{12.3c}$$

Here, $\mathbb{A} = -1/\left(\mu \frac{dQ_2}{dt}\right)$, $\mathbb{B} = 1/\left(\mu \frac{dQ_2}{dt}\right)$ and $\mathbb{C} = 1/\left(\varepsilon \frac{dQ_2}{dt}\right)$ can be stored into the memory.

Latency for General Purpose Instructions

Table 12.3 presents the the instructions and latency for Eqs. 12.3. It can be noted that for Eqs. 12.3a and 12.3b require a total of 29 cycles latency each. This is much higher than that for the 2D FDTD, where equivalent Eqs. 12.2a and 12.2b require only 11 cycles each as shown in Table 12.1. For Eq. 12.3c the comparison is even wider as it requires 53 cycles compared to Eq. 12.2c, which requires only 17 in Table 12.1. Theoretically, the proposed 2D FETD is 2.846 times slower than 2D FDTD method when only general purpose instructions are used and the number of cells/elements are kept equal in both methods.

Table 12.3 Compute Operations and Latencies for 2D FETD Method with General Purpose Instructions

Equ.	**Add**		**Sub**		**Mult**		**Total**	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
12.3a	3	3	0	3	4	5	7	29
12.3b	3	3	0	3	4	5	7	29
12.3c	5	3	1	3	7	5	13	53
Total for $\mathbf{E} \rightarrow \mathbf{H}$:							**14**	**58**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:							**13**	**53**
Total for One Time-step:							**27**	**111**

Latency for General Purpose and SIMD Instructions

Table 12.4 shows the theoretical performance of the proposed 2D FETD when both general purpose and SIMD instructions are used for implementation. It can be noted that, Eqs. 12.3 scale better than Eqs. 12.2 when SIMD instructions are used to optimise the performance of both the methods. Eqs. 12.3a and 12.3b each requires only 19 cycles compares to 29 in Table 12.3. Eq. 12.3c scales even better reducing the latency from 53 in Table 12.3 to 27 in Table 12.4.

Therefore, combining general purpose instructions with SIMD instructions is desired for the proposed method. On the contrary the SIMD implementation of the 2D FDTD method sees very little improvement.

Although with SIMD instruction added the proposed 2D FETD method become significantly faster compared to the general purpose implementation, Theoretically it is still slower than both the SIMD and non-SIMD 2D FDTD implementations by a factor of $\mathbf{1.806}$ and $\mathbf{1.667}$, respectively when the number of elements considered to be equal for both the methods.

Table 12.4 Compute Operations and Latencies for 2D FETD Method with General Purpose and SIMD Instructions

Equ.	**Add**		**SIMD Add**		**Sub**		**Mult**		**SIMD Mult**		**Total**	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
12.3a	3	3	0	3	0	3	1	5	1	5	5	19
12.3b	3	3	0	3	0	3	1	5	1	5	5	19
12.3c	1	3	2	3	1	3	1	5	2	5	7	27
Total for $\mathbf{E} \rightarrow \mathbf{H}$:											**10**	**38**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:											**7**	**27**
Total for One Time-step:											**17**	**65**

12.1.3 Comparing Proposed FETD and FDTD considering RRF in Two-dimensions

As we have demonstrated in Section 11.1.4 that, when the numerical dispersion is considered the resolution in for the 2D FDTD method is not equal to the resolution of the 2D FETD method, when ET mesh is used with the proposed 2D FETD. With ET mesh and proposed FETD the similar accuracy in numerical dispersion can be achieved with lower density mesh. Therefore, to have a fair comparison the resolution reduction factor for two-dimensions presented in Section 11.1.2 should be considered.

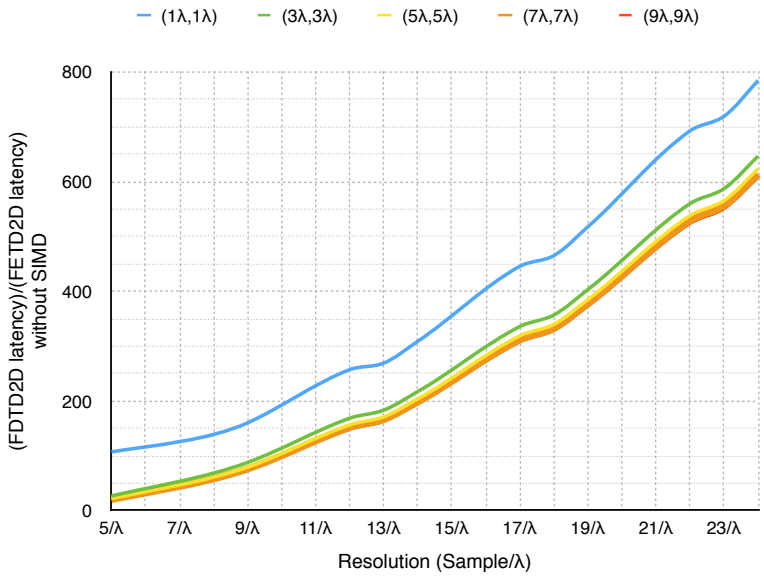
To compare the CPU latency performance of the proposed method and the FDTD2D method and to consider the RRF, a programme was developed in Python which calculates the CPU latency of one time-step for the proposed FETD2D method without the SIMD improvements using results from Table 12.3 and calculate CPU latency with SIMD improvements using results from Table 12.4. It also calculated the CPU latency without and with SIMD improvements for the FDTD2D method using the results from Tables 12.1 and 12.2, respectively. The program implements the **RRF** of Eq. 11.6 as a subroutine and uses the input FETD2D resolution to calculate the equivalent FDTD2D resolution by the following equation,

$$r_{FDTD} = RRF(r_{ET}) \cdot r_{ET} \tag{12.4}$$

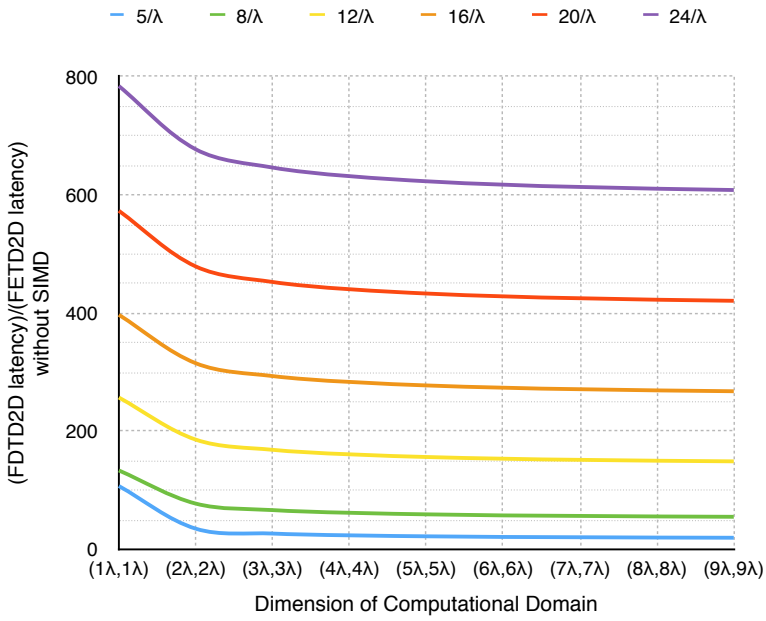
The output of the subroutine for Eq. 12.4 is used by the program to calculate the CPU latencies (without and with SIMD improvement) for the FDTD2D method.

The performance can be compared by calculating the ratio of the FDTD2D CPU latency over FETD2D CPU latency.

The program was used to calculate the CPU latency of one time-step for different size of computational domain at different FETD2D resolutions. Figure 12.1 presents the result obtained from the calculations. To make calculation simple, only square computational domains was considered and only multiples of wavelength was considered as dimensions of

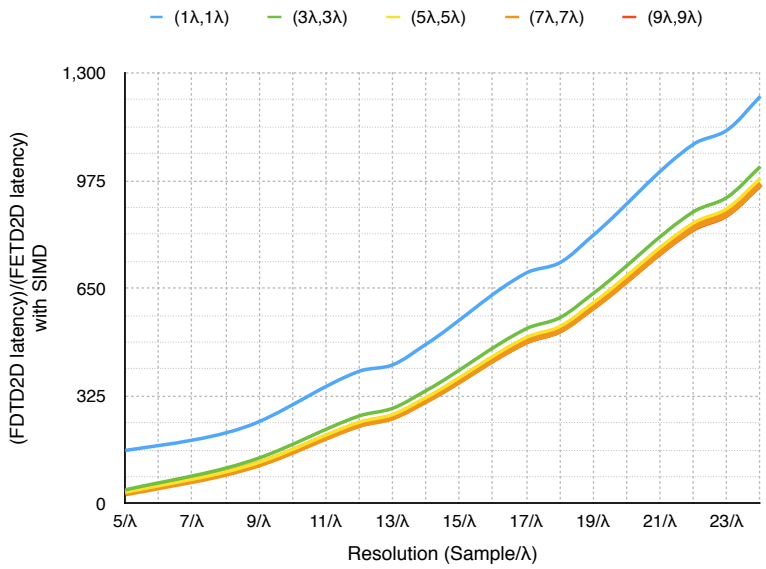


(a) FDTD2D latency over FETD2D latency vs resolution for different size of computational domain

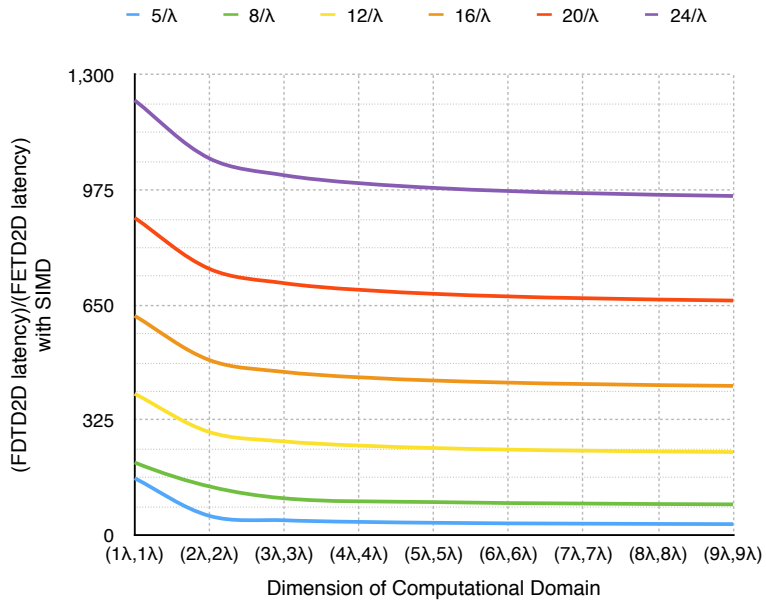


(b) FDTD2D latency over FETD2D latency vs size of computational domain for different resolutions

Fig. 12.1 CPU latency (without SIMD enhancement) comparison between FDTD2D and FETD2D with ET mesh



(a) FDTD2D latency over FETD2D latency vs resolution for different size of computational domain



(b) FDTD2D latency over FETD2D latency vs size of computational domain for different resolutions

Fig. 12.2 CPU latency (with SIMD enhancement) comparison between FDTD2D and FETD2D with ET mesh

domains. For analysis domain size is represented as $(a\lambda, a\lambda)$ where, $a \in \mathbb{N}$.

Figure 12.1a shows affect of increasing resolution on the CPU latency ratio. The resolution is defined as "sample/node per wavelength" or $sample/\lambda$. This format of resolution has been chosen because, the characteristics apply to all resolution as the sample per wavelength are increased. The resolution used in the x-axis of Fig. 12.1a is the r_{ET} in Eq. 12.4. As mentioned earlier the equivalent resolution for the FDTD was calculated using Eq. 12.4 and was for the simulation. But has been plotted against r_{ET}. This ensures the comparison for equivalent result (or error level based on STD). This approach has been followed for the SIMD comparison and also for the 3D analysis. As can be seen, the ratio increases with the increase in resolution for all size of computational domain. It can also be noticed that for bigger computational domain the ratio is decreasing. It also can be noted that the reduction of speed is saturating as the domain size is increased. This phenomenon can be better illustrated in Fig. 12.1.

In Fig. 12.1, the relation between the size of the domain and the CPU latency ratio is shown. As can be seen, the ratio initially decreases and settles down as the reduction slows down. This happens for all the resolutions.

Similar relations can be observer when SIMD enhancements are considered. Figure 12.2 shows the results when SIMD enhancement is considered. Similar to Fig. 12.1a for all domain size the CPU latency ratio increases with the resolution.

Figure 12.2 shows that with the increasing domain size similar to Fig. 12.1, the CPU latency radio decreases and saturated at for all the resolutions.

In fact the only difference between Figs. 12.1 and Figs. 12.2 is the increase in the CPU latency ratio for the SIMD implementation. The SIMD performance is on an average **1.57** times better than non-SIMD performance. In both occasions the CPU latency performance of the proposed FETD2D method is far better than the FDTD2D method because for all resolutions used in Figs. 12.1 and 12.2 the CPU latency ratio is always higher than 1 meaning higher than the speed of the FDTD2D when **RRF** and equivalent resolution is considered.

12.2 CPU Performance for Three-dimensionals Formulation

Similar to the two-dimensional CPU performance analysis in Section 12.1 a CPU optimisation has to be performed on both the FDTD and the proposed FETD 3D governing equations. In this section Eqs. 7.9 will be used to optimise and derive the CPU optimised form of the FETD3D method.

12.2.1 CPU Optimised Formulation of the FDTD in Three-dimensions

The discretised form governing equations for the FDTD method in 3D has been presented in [12, 25] as,

$$H_x|_{i,j,k}^{n+1/2} = -\frac{\Delta t}{\mu_{i,j,k}}\left[\frac{E_z|_{i,j+1/2,k}^{n} - E_z|_{i,j-1/2,k}^{n}}{\Delta y} - \frac{E_y|_{i,j,k+1/2}^{n} - E_y|_{i,j,k-1/2}^{n}}{\Delta z}\right] + H_x|_{i,j,k}^{n-1/2} \tag{12.5a}$$

$$H_y|_{i,j,k}^{n+1/2} = \frac{\Delta t}{\mu_{i,j,k}}\left[\frac{E_z|_{i+1/2,j,k}^{n} - E_z|_{i-1/2,j,k}^{n}}{\Delta x} - \frac{E_x|_{i,j,k+1/2}^{n} - E_x|_{i,j,k-1/2}^{n}}{\Delta z}\right] + H_y|_{i,j,k}^{n-1/2} \tag{12.5b}$$

$$H_z|_{i,j,k}^{n+1/2} = -\frac{\Delta t}{\mu_{i,j,k}}\left[\frac{E_y|_{i+1/2,j,k}^{n} - E_y|_{i-1/2,j,k}^{n}}{\Delta x} - \frac{E_x|_{i,j+1/2,k}^{n} - E_x|_{i,j-1/2,k}^{n}}{\Delta y}\right] + H_z|_{i,j,k}^{n-1/2} \tag{12.5c}$$

$$E_x|_{i,j,k}^{n+1/2} = \frac{\Delta t}{\varepsilon_{i,j,k}}\left[\frac{H_z|_{i,j+1/2,k}^{n} - H_z|_{i,j-1/2,k}^{n}}{\Delta y} - \frac{H_y|_{i,j,k+1/2}^{n} - H_y|_{i,j,k-1/2}^{n}}{\Delta z}\right] + E_x|_{i,j,k}^{n-1/2} \tag{12.5d}$$

$$E_y|_{i,j,k}^{n+1/2} = -\frac{\Delta t}{\varepsilon_{i,j,k}}\left[\frac{H_z|_{i+1/2,j,k}^{n} - H_z|_{i-1/2,j,k}^{n}}{\Delta x} - \frac{H_x|_{i,j,k+1/2}^{n} - H_x|_{i,j,k-1/2}^{n}}{\Delta z}\right] + E_y|_{i,j,k}^{n-1/2} \tag{12.5e}$$

$$E_z|_{i,j,k}^{n+1/2} = \frac{\Delta t}{\varepsilon_{i,j,k}}\left[\frac{H_y|_{i+1/2,j,k}^{n} - H_y|_{i-1/2,j,k}^{n}}{\Delta x} - \frac{H_x|_{i,j+1/2,k}^{n} - H_x|_{i,j-1/2,k}^{n}}{\Delta y}\right] + E_z|_{i,j,k}^{n-1/2} \tag{12.5f}$$

To calculate the minimum number of CPU instructions required for the calculation of a single time-step, all 6 equations in Eqs. 12.5 have to be optimised for the CPU in a way similar to Section 12.1.1. The CPU optimal formulation of the 3D FDTD method can be presented as,

$$H_x|_{i,j,k}^{n+1/2} = \mathbb{A}\left[\left(E_y|_{i,j,k+1/2}^{n} - E_y|_{i,j,k-1/2}^{n}\right) - \left(E_z|_{i,j+1/2,k}^{n} - E_z|_{i,j-1/2,k}^{n}\right)\right] + H_x|_{i,j,k}^{n-1/2} \tag{12.6a}$$

$$H_y|_{i,j,k}^{n+1/2} = \mathbb{A}\left[\left(E_z|_{i+1/2,j,k}^{n} - E_z|_{i-1/2,j,k}^{n}\right) - \left(E_x|_{i,j,k+1/2}^{n} - E_x|_{i,j,k-1/2}^{n}\right)\right] + H_y|_{i,j,k}^{n-1/2} \tag{12.6b}$$

$$H_z|_{i,j,k}^{n+1/2} = \mathbb{A}\left[\left(E_x|_{i,j+1/2,k}^{n} - E_x|_{i,j-1/2,k}^{n}\right) - \left(E_y|_{i+1/2,j,k}^{n} - E_y|_{i-1/2,j,k}^{n}\right)\right] + H_z|_{i,j,k}^{n-1/2} \tag{12.6c}$$

$$E_x|_{i,j,k}^{n+1/2} = \mathbb{B}\left[\left(H_z|_{i,j+1/2,k}^{n} - H_z|_{i,j-1/2,k}^{n}\right) - \left(H_y|_{i,j,k+1/2}^{n} - H_y|_{i,j,k-1/2}^{n}\right)\right] + E_x|_{i,j,k}^{n-1/2} \tag{12.6d}$$

$$E_y|_{i,j,k}^{n+1/2} = \mathbb{B}\left[\left(H_x|_{i,j,k+1/2}^{n} - H_x|_{i,j,k-1/2}^{n}\right) - \left(H_z|_{i+1/2,j,k}^{n} - H_z|_{i-1/2,j,k}^{n}\right)\right] + E_y|_{i,j,k}^{n-1/2} \tag{12.6e}$$

$$E_z|_{i,j,k}^{n+1/2} = \mathbb{B}\left[\left(H_y|_{i+1/2,j,k}^{n} - H_y|_{i-1/2,j,k}^{n}\right) - \left(H_x|_{i,j+1/2,k}^{n} - H_x|_{i,j-1/2,k}^{n}\right)\right] + E_z|_{i,j,k}^{n-1/2} \tag{12.6f}$$

Here, $\mathbb{A} = \frac{\Delta t}{\mu_{i,j,k}\Delta}$ and $\mathbb{B} = \frac{\Delta t}{\varepsilon_{i,j,k}\Delta}$ can be stored in the memory for each node in the grid.

It can be noted that the Eqs. 12.6 avoid all redundant operations by storing them onto the memory. It should also be noted that the equations are composed of CPU efficient addition, subtraction and multiplication.

Latency for General Purpose Instructions

Table 12.5 shows the instructions required for each equation in Eqs. 12.6 when general purpose instruction of Intel Haswell architecture. Unlike the 2D formulation in Eqs. 12.2 all equations in Eqs. 12.6 requires 5 instructions and 17 cycles each.

Table 12.5 Compute Operations and Latencies for 3D FDTD Method with General Purpose Instructions

Equ.	**Add**		**Sub**		**Mult**		**Total**	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
12.6a	1	3	3	3	1	5	5	17
12.6b	1	3	3	3	1	5	5	17
12.6c	1	3	3	3	1	5	5	17
12.6d	1	3	3	3	1	5	5	17
12.6e	1	3	3	3	1	5	5	17
12.6f	1	3	3	3	1	5	5	17
Total for **E** $\rightarrow$ **H**:							**15**	**51**
Total for **H** $\rightarrow$ **E**:							**15**	**51**
Total for One Time-step:							**30**	**102**

Table 12.6 Compute Operations and Latencies for 3D FDTD Method with General Purpose and SIMD Instructions

Equ.	**Add**		**Sub**		**SIMD Sub**		**Mult**		**Total**	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
12.6a	1	3	1	3	1	3	1	5	4	14
12.6b	1	3	1	3	1	3	1	5	4	14
12.6c	1	3	1	3	1	3	1	5	4	14
12.6d	1	3	1	3	1	3	1	5	4	14
12.6e	1	3	1	3	1	3	1	5	4	14
12.6f	1	3	1	3	1	3	1	5	4	14
Total for $\mathbf{E} \rightarrow \mathbf{H}$:									**14**	**42**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:									**14**	**42**
Total Operation for One Time-step:									**28**	**84**

Latency for General Purpose and SIMD Instructions

Table 12.6 presented the latency distribution for the 3D FDTD method. As can be noticed, unlike the 2D FDTD, theoretically all the Eqs. 12.6 benefited from the use of SIMD instructions with the general purpose instructions. Compared to latency distribution presented in Table 12.5 for only general purpose instructions, every equation in Table 12.6 gets a latency reduction for the use of 1 SIMD subtraction instruction. Therefore, each of them gets a small latency reduction. The total impact is significant as the total latency for calculating 1 time-step reducers from 102 to **84**.

12.2.2 CPU Optimised Formulation for Proposed FETD in Three-dimensions

The discretised form of the proposed 3D FETD method has been presented in Eqs. 7.9. Similar to the 2D formulation in Section 12.1.2, no optimisation are performed for using only the ET3D mesh. This is to preserve the capability to move the nodes of the tetrahedron to produce any shape if necessary. All optimisations are done to reduce redundant computation by performing them prior to the execution of the governing equations and storing them into memory. The CPU optimised form of the governing equations can be presented as follows,

$$h_x^{(n+1)} = \mathbb{C}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial z}e_{yi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial y}e_{zi}^{(n)}\right) + h_x^{(n-1)} \tag{12.7a}$$

$$h_y^{(n+1)} = \mathbb{C}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}e_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}e_{xi}^{(n)}\right) + h_y^{(n-1)} \tag{12.7b}$$

$$h_z^{(n+1)} = \mathbb{C}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial y}e_{xi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial x}e_{yi}^{(n)}\right) + h_z^{(n-1)} \tag{12.7c}$$

$$e_x^{(n+1)} = \mathbb{D}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial y}h_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}h_{yi}^{(n)}\right) + e_x^{(n-1)} \tag{12.7d}$$

$$e_y^{(n+1)} = \mathbb{D}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial z}h_{xi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial x}h_{zi}^{(n)}\right) + e_y^{(n-1)} \tag{12.7e}$$

$$e_z^{(n+1)} = \mathbb{D}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}h_{yi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial y}h_{xi}^{(n)}\right) + e_{zk}^{(n-1)} \tag{12.7f}$$

Here, $\mathbb{C} = 1/\left(\mu\frac{dQ_2}{dt}\right)$ and $\mathbb{D} = 1/\left(\varepsilon\frac{dQ_2}{dt}\right)$ can be stored in the memory for each associated element.

Latency for General Purpose Instructions

Table 12.7 Compute Operations and Latencies for 3D FETD Method with General Purpose Instructions

Equ.	Add		Sub		Mult		Total	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
12.7a	7	3	1	3	9	5	17	69
12.7b	7	3	1	3	9	5	17	69
12.7c	7	3	1	3	9	5	17	69
12.7d	7	3	1	3	9	5	17	69
12.7e	7	3	1	3	9	5	17	69
12.7f	7	3	1	3	9	5	17	69
Total for $\mathbf{E} \rightarrow \mathbf{H}$:							**51**	**207**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:							**51**	**207**
Total Operation for One Time-step:							**102**	**414**

Table 12.7 list the latency distribution for the CPU optimised formulation for the proposed 3D FETD governing equations in Eqs. 12.7. As can be seen in the table, each of the equations cause same latency similar to the 3D FDTD method presented in Table 12.5. But each of the equations for the 3D FETD requires **69** CPU cycles compared to **17**. The main reason for the excessive use of the addition and multiplication operations. Excessive use of general instructions made the total performance for each time-step much slower than the 3D FDTD performance with general purpose instructions. The 3D FDTD theoretically requires **102** CPU cycles compared to the proposed 3D method requiring **414**. Therefore, the performance of the proposed 3D method with general purpose instructions are almost **4** times slower than the 3D FDTD with general purpose instructions.

Table 12.8 presents the latencies associated with the proposed method with general propose and SIMD instructions together. It can be noted that when SIMD instructions are used for additions and multiplications, the latency of all the equations in Eqs. 12.7 decrease significantly from 69 CPU cycles to **27** CPU cycles. Therefore, the total reduction of latency is more than **60**% from 414 CPU cycles in Table 12.7 to **162** in Table 12.8.

Latency for General Purpose and SIMD Instructions

Table 12.8 Compute Operations and Latencies for 3D FETD Method with General Purpose and SIMD Instructions

Equ.	**Add**		**SIMD Add**		**Sub**		**Mult**		**SIMD Mult**		**Total**	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
12.7a	1	3	2	3	1	3	1	5	2	5	7	27
12.7b	1	3	2	3	1	3	1	5	2	5	7	27
12.7c	1	3	2	3	1	3	1	5	2	5	7	27
12.7d	1	3	2	3	1	3	1	5	2	5	7	27
12.7e	1	3	2	3	1	3	1	5	2	5	7	27
12.7f	1	3	2	3	1	3	1	5	2	5	7	27
Total for $\mathbf{E} \rightarrow \mathbf{H}$:											**21**	**81**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:											**21**	**81**
Total Operation for One Time-step:											**42**	**162**

Although the reduction is more significant compared to the 3D FDTD method in Table 12.5, the CPU latency for the proposed 3D method with the SIMD enhancements is still almost **twice** slower compared to the 3D FDTD method with similar enhancement when the number of elements/cells are equal for both the methods. As it was mentioned before, the proposed method does not employ any optimisation for the shape of the mesh used. Whereas the 3D FDTD method is optimised for the cubic grid. This is the primary cause of the slower performance.

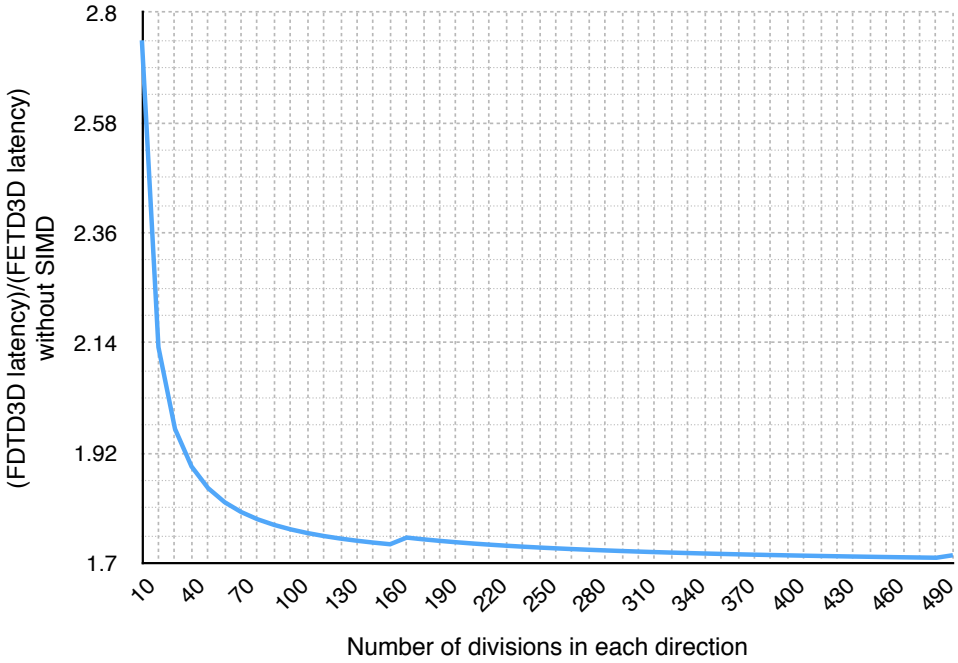
12.2.3 Comparing Proposed FETD and FDTD considering RRF in Three-dimensions

To make the method faster than the 3D FDTD method a technique should be applied which reduces the number of elements without sacrificing the quality of the result. Similar to the 2D FETD in Section 11.1.2, the resolution reduction factor in Section 11.2.2 provides a way to reduce the number of elements on a ET3D mesh. As we mentioned at the beginning of this chapter, only equilateral meshes are going to be used. This section will discuss the affect of considering RRF with the proposed method.

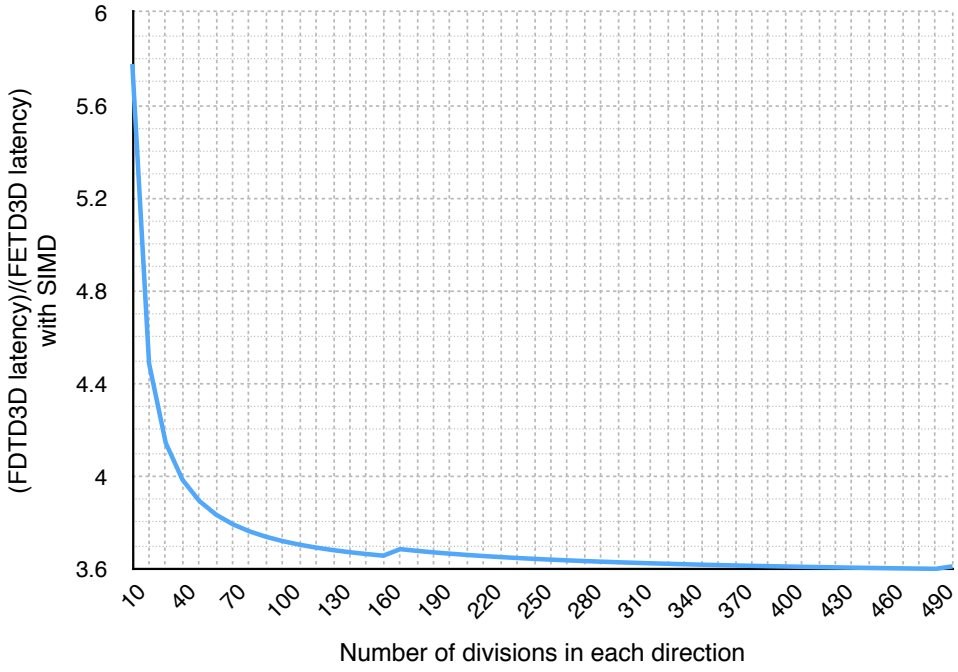
As mentioned in Section 11.2.2 the RRF is **1.903** for all resolutions for ET3D mesh and the cubic FDTD grid, this will allow us to reduce the resolution by almost half. But the actual reduction of number of elements would much higher as the number of elements will be a cubic function of the effective resolution.

To analyse the affect of the RRF on the speed of the proposed method a Python code was developed. Since the RRF is insensitive of resolution, for this analysis the resolution for the proposed 3D FETD was considered to be $10/\lambda$. Therefore, the resolution for the 3D FDTD method was 19. The domain size was determined interns of number of division in each direction. For simplicity, a cubic domain was considered.

The program generated required number of elements necessary for a cubic domain with ET3D mesh and calculated the latency using the information presented in Table 12.7 and 12.8. It also generated the 3D cubic grid with necessary number of cells and calculated



(a) FDTD2D latency over FETD2D latency vs size of computational domain without SIMD Instructions



(b) FDTD2D latency over FETD2D latency vs size of computational domain with SIMD Instructions

Fig. 12.3 CPU latency comparison between FDTD2D and FETD2D with ET3D mesh

the CPU latency using information from Table 12.5 and 12.6.

A theoretical CPU performance ratio can be calculated by dividing the CPU latency of the 3D FDTD method by the CPU latency of the proposed 3D method.

Figure 12.3 shows the outputs of the Python code[1]. As can be seen in Fig. 12.3a, the CPU performance ratio is much higher at when the number of division is much lower. It decrease with a saturation curve and settles down around **1.71** at higher resolutions when only general purpose instructions are considered with the RRF.

Figure 12.3b shows the performance with SIMD instructions. Similar to Fig. 12.3a, the CPU performance ratio is higher at the beginning and settles around **3.6** at higher resolution.

Although both Fig. 12.3a and 12.3b shows higher performance compared to the 3D FDTD method, the an implementation with SIMD instructions are more likely to hold the CPU performance advantage when implemented for real world usage.

For this performance analysis only the advantage of better numerical dispersion of the proposed FETD is considered to illustrate possible CPU time advantage of the method over the FDTD method.

Moreover, the FETD method can use irregular mesh for many problems required no. of node points for the FETD method can be much less than the FDTD method which uses regular rectangular grid to discretise the problem domain. This advantage of the proposed was not studied here.

[1]The small increase in performance at the FETD resolution 170 is due to a round off issue. As resolutions are integers the Python code used `round()` function from the numpy package to round off the equivalent resolution for the FDTD calculation. As the RRF is 1.903, the round off function started to chose higher value from the FETD resolution 170. Hence, both the plot shows a small performance increase at 170.

Part IV

Future Plan and Conclusions

Chapter 13

Future Works

The FE-based method presented in this thesis shows significant improvement in both accuracy and CPU performance over the FDTD method by showing faster executing time. To achieve the level of performance the method only utilised the resolution reduction factor for both 2D and 3D implementations. But to utilise full potential of the method proposed further developments are necessary. Specially to improve the speed farther, increase the accuracy and reduce the memory requirement of the method further development is needed. To convert the proposed method into a faster, more efficient and more accurate replacement for the FDTD method following future research can be carried out.

13.1 Unstructured Mesh

The method presented in this thesis has used structured grid. To utilise the full potential of the FE mesh of the proposed method it is essential to use unstructured mesh with the method. An unstructured mesh could make the method more efficient by reducing the number of elements in materials where the wavelength is larger. It might be able to make dense and corse mesh as per requirement of the analysis more efficiently then the FD counterpart. Therefore, improve the efficiency of the method even higher.

Unlike other FE-based methods, the proposed method was developed keeping performance and accuracy as the top priority. As a result, there are a few requirements of method

that has to be satisfied by the unstructured mesh. Therefore, an off-the-shelf meshing library or algorithm may not be attached to the method without any preconditioning. The requirements are as follows,

The Mesh: The mesh has to be as it is a requirement by the first order implementation of the method. At present there is no meshing library which produces a mesh like the mesh used in this work similar to the meshes discussed in Chapter 3 and 8 for 2D and 3D, respectively. Therefore, mesh generated by any off the shelf library has to be preprocessed before using with the method. The mesh used is also one of the reasons for the higher CPU performance of the proposed method.

Equilateral or Near Equilateral Elements: The shape of the element is crucial for both 2D and 3D. Specially the numerical dispersion performance heavily depends on the shape. As has been discussed in Chapter 11, Sections 11.1.1 and 11.2.1, equilateral elements provide much better numerical dispersion characteristics that might allow simulation at a lower resolution (discussed in section 11.1.2 and 11.2.2). Although in many practical situations it might not be possible to have all equilateral elements, having most of the elements equilateral or near equilateral will allow the similar advantages. Thereby it will not only reduce the error level of the simulation but also reduce the CPU time and memory resulting in faster simulation.

Double Mesh System: To make the calculation explicit, a double mesh system consisting of the main and auxiliary mesh (Chapters 3 and 8) was developed using the regular meshes in this thesis. Similar double mesh system has to be developed for the unstructured mesh by generating the auxiliary mesh by connecting some inner points (*i.e.* centroid) of the elements. Although it may sound trivial, but with unstructured mesh generating an auxiliary mesh might be quite tricky.

Gradual Change in Density: From Fig. 11.1 it can be noted that with the increase in resolution the average speed of propagation is also increasing. With unstructured grid in use if the difference in resolutions between the dense and course regions are high,

the speed of propagation will abruptly change. Thereby, causing an impedance mismatch and will cause spurious reflection. Therefore, any change in resolution has to be gradual enough to make reduce the mismatch in impedance negligible.

The unstructured mesh is very important because it will not only make discretisation of the structure easier and more accurate but also allow the user to define dense and course according to his necessity. For example, EM wave inside a denser material (Silica) requires higher density of points to represent it accurately compared to the same in a lighter material (Air). Therefore, it is desirable to have smaller size elements in dense materials and larger elements in lighter materials. This can only be achieved efficiently by adopting a unstructured meshing algorithm for the method. The result, will be farther reduction of total number of elements making the proposed method even faster in execution.

13.2 Variable Time-stepping

With the use of unstructured mesh, there will be small and large elements in the same mesh. Therefore, to maintain stability the time-step of the simulation has to be calculated with the smallest element in the domain. In many occasions the number of small elements will be small compared to the total number of computational domain. This choice of smallest possible time-step might be wasteful in-terms of CPU time. As most of the elements are course, a more sensible choice will be to select a larger time-step. But that will cause instability in the smaller elements.

To avoid instability with larger time-step a variable time-stepping mechanism has to be developed, where smaller elements will use small time-step multiple times before the entire mesh uses a large time-step to go forward in time. This was the smaller elements (inferior in number) will perform required number of time-steps while the larger elements will be spared from executing over and over. Hence, this technique could save significant CPU time.

13.3 Partial Mesh Solution for Pulse Propagation

In the special case of pulse propagation, the computational domain is excited with time limited pulse which normally propagates in one or more specific direction depending on the structure of the device. An intelligent technique can be developed which will track the propagation of power inside the computational domain and have the ability to switch on an off elements for computation depending on the movement of the pulse in side the computational domain. This could cut down majority of the calculation as for pulse propagation the power normally stays in one or more small areas of the entire domain.

13.4 Higher Order Implementation

Higher order implementation of the proposed technique could increase the speed and accuracy of the method even higher as higher order implementation will allow bigger elements. Thereby, it might reduce the error level in the simulation farther.

Chapter 14

Conclusion

The principal objective of this research work was to develop a finite element based technique to solve the electromagnetic time domain problem in a fast and efficient manner which can be considered as a realistic alternative to the standard FDTD method. All the objectives mentioned in the motivation section of the introductory chapter have been fulfilled in this research. To develop the technique the accuracy of result and CPU performance were given the top most priorities. To develop an explicit and data parallel formulation the Maxwell's equations in their differential form was considered which is similar as in the FDTD method.

The explicit formulation of the equations were discretised with using linear finite elements (triangles for 2D and tetrahedrons for 3D). As the Maxwell's equations are coupled a unique coupled dual mesh system was introduced in this thesis in Chapters 3 and 8. Hence the number of elements required were reduced to half for 2D and $1/5$ times for 3D when compared with full mesh systems.

Several benchmarking simulations were performed and reported in Chapters 6 and 10 to show the effectiveness and of the method compared to other numerical techniques like the FEM method for modal analysis. To show the strength of the method complex metamaterial simulation was performed and the obtained result was compared with the published result. For reflection and interference performance in 3D nano power splitter was simulated and the results were analysed in Section 10.3.

To analyse the accuracy of the method the numerical dispersion relation of both the 2D

and 3D implementation was derived from the governing equations in Chapter 11. It has been shown in this chapter that for both 2D and 3D equilateral elements for the proposed method provide significant advantage in terms of numerical dispersion. In these chapters a new parameter called resolution reduction factor was introduced and it was shown by simulation using both the proposed method and the FDTD method that a lower resolution simulation with the proposed method is more accurate when equilateral elements are used (Sections 11.1.1, 11.1.4 and 11.2.1). The resolution reduction factors for both 2D and 3D are measured and discussed in Sections 11.1.2 and 11.2.2.

In Chapter 12 a theoretical analysis was performed on the CPU performance of the proposed method for both 2D and 3D implementations. Theoretical CPU performance of the proposed method was compared with the theoretical CPU performance of the FDTD method. To do this analysis CPU latency information was obtained from latest CPU documentation from Intel [85]. In this chapter it was shown that when the resolution reduction factor is considered the CPU performance of the proposed method become much faster than that of the FDTD method. When Single Instruction Multiple Data (SIMD) instructions are used alongside the general purpose instructions, the method become much faster than the FDTD method.

At the end of the research higher CPU efficiency was achieved for both 2D and 3D by only considering the resolution reduction factor. As the proposed method is an FE-based method, all advantage of Finite Elements can be incorporated with the method. At the end of the research probable direction for further research and development was pointed out in Chapter 13. All the proposed improvement in the chapter will utilise the FE-roots of the method to increase the speed of simulation even faster by reducing the number of computational elements by incorporating intelligent techniques which are only possible to implement in FE based method and not applicable in FD based FDTD method. Thus this thesis clearly shows the way to make the proposed method the faster, more efficient and more accurate future replacement for the FDTD method.

References

[1] B. A. Rahman and A. Agrawal, *Finite Element Modeling Methods for Photonics*. Artech House, 2013.

[2] D. Schurig, J. Mock, B. Justice, S. Cummer, J. Pendry, A. Starr, and D. Smith, "Metamaterial electromagnetic cloak at microwave frequencies," *Science*, vol. 314, no. 5801, pp. 977–980, 2006.

[3] A. Silva, F. Monticone, G. Castaldi, V. Galdi, A. Alù, and N. Engheta, "Performing mathematical operations with metamaterials," *Science*, vol. 343, no. 6167, pp. 160–163, 2014.

[4] E. Yamashita, *Analysis methods for electromagnetic wave problems*. Artech House Boston London, 1990.

[5] R. F. Harrington and J. L. Harrington, *Field computation by moment methods*. Oxford University Press, 1996.

[6] M. M. Ney, "Method of moments as applied to electromagnetic problems," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 33, no. 10, pp. 972–980, 1985.

[7] E. Yamashita and R. Mittra, "Variational method for the analysis of microstrip lines," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 16, no. 4, pp. 251–256, 1968.

[8] K. Stamnes, S.-C. Tsay, W. Wiscombe, K. Jayaweera *et al.*, "Numerically stable algo-

rithm for discrete-ordinate-method radiative transfer in multiple scattering and emitting layered media," *Applied optics*, vol. 27, no. 12, pp. 2502–2509, 1988.

[9] C. Christopoulos, *The transmission-line modeling method: TLM*. Institute of Electrical and Electronics Engineers (New York and Oxford), 1995.

[10] W. J. Hoefer, "The transmission-line matrix method–theory and applications," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 33, no. 10, pp. 882–893, 1985.

[11] G. Mur, "A finite difference method for the solution of electromagnetic waveguide discontinuity problems," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 22, no. 1, pp. 54–57, Jan 1974.

[12] S. Hagness and A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method (Second Edition)*, 2nd ed. Artech House, 2000.

[13] J. Yamauchi, *Propagating beam analysis of optical waveguides*. Research Studies Press Exeter, 2003.

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

[15] B. Rahman and J. Davies, "Finite-element analysis of optical and microwave waveguide problems," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 32, no. 1, pp. 20–28, 1984.

[16] M. Koshiba, *Optical waveguide theory by the finite element method*. Ktk Scientific, 1993.

[17] S. Obayya, *Computational photonics*. John Wiley & Sons, 2011.

[18] G. R. Werner and J. R. Cary, "A stable fdtd algorithm for non-diagonal, anisotropic dielectrics," *Journal of Computational Physics*, vol. 226, no. 1, pp. 1085–1101, 2007.

[19] G. R. Werner, C. A. Bauer, and J. R. Cary, "A more accurate, stable, fdtd algorithm for electromagnetics in anisotropic dielectrics," *Journal of Computational Physics*, vol. 255, pp. 436–455, 2013.

[20] M. Clemens and T. Weiland, "Magnetic field simulation using conformal fit formulations," *Magnetics, IEEE Transactions on*, vol. 38, no. 2, pp. 389–392, 2002.

[21] T. Weiland, "A discretization model for the solution of maxwell's equations for six-component fields," *Archiv Elektronik und Uebertragungstechnik*, vol. 31, pp. 116–120, 1977.

[22] M. C. T. Weiland, "Discrete electromagnetism with the finite integration technique," *Progress In Electromagnetics Research*, vol. 32, pp. 65–87, 2001.

[23] J. Whinnery, S. Ramo, and T. Van Duzer, "Fields and waves in communication electronics," *J. Wiley and sons*, 1994.

[24] J. D. Joannopoulos, S. G. Johnson, J. N. Winn, and R. D. Meade, *Photonic Crystals: Molding the Flow of Light (Second Edition)*, 2nd ed. Princeton University Press, 2008.

[25] A. Taflove and S. Hagness, *Computational electrodynamics*. Artech house Boston, 1995.

[26] M. N. Sadiku, *Numerical techniques in electromagnetics*. CRC press, 2000.

[27] M. Koshiba, K. Hayata, and M. Suzuki, "Approximate scalar finite-element analysis of anisotropic optical waveguides," *Electronics Letters*, vol. 18, no. 10, pp. 411–413, 1982.

[28] B. Rahman and J. Davies, "Finite-element solution of integrated optical waveguides," *Lightwave Technology, Journal of*, vol. 2, no. 5, pp. 682–688, 1984.

[29] A. Berk, "Variational principles for electromagnetic resonators and waveguides," *Antennas and Propagation, IRE Transactions on*, vol. 4, no. 2, pp. 104–111, 1956.

[30] Y. Tsuji and M. Koshiba, "A finite element beam propagation method for strongly guiding and longitudinally varying optical waveguides," *Lightwave Technology, Journal of*, vol. 14, no. 2, pp. 217–222, 1996.

[31] M. Feit, J. Fleck Jr *et al.*, "Light propagation in graded-index optical fibers," *Applied optics*, vol. 17, no. 24, pp. 3990–3998, 1978.

[32] M. Feit and J. Fleck Jr, "Computation of mode properties in optical fiber waveguides by a propagating beam method," *Applied Optics*, vol. 19, no. 7, pp. 1154–1164, 1980.

[33] D. Pinto and S. Obayya, "Accurate perfectly matched layer finite-volume time-domain method for photonic bandgap devices," *Photonics Technology Letters, IEEE*, vol. 20, no. 5, pp. 339–341, 2008.

[34] A. Cangellaris, C. Lin, and K. Mei, "Point-matched time domain finite element methods for electromagnetic radiation and scattering," *Antennas and Propagation, IEEE Transactions on*, vol. 35, no. 10, pp. 1160–1173, 1987.

[35] M. Feliziani and E. Maradei, "Point matched finite element-time domain method using vector elements," *Magnetics, IEEE Transactions on*, vol. 30, no. 5, pp. 3184–3187, 1994.

[36] J. Lee, R. Lee, and A. Cangellaris, "Time-domain finite-element methods," *Antennas and Propagation, IEEE Transactions on*, vol. 45, no. 3, pp. 430–442, 1997.

[37] S. Obayya, "Efficient finite-element-based time-domain beam propagation analysis of optical integrated circuits," *Quantum Electronics, IEEE Journal of*, vol. 40, no. 5, pp. 591–595, 2004.

[38] F. Teixeira, "Time-domain finite-difference and finite-element methods for maxwell equations in complex media," *Antennas and Propagation, IEEE Transactions on*, vol. 56, no. 8, pp. 2150 –2166, aug. 2008.

[39] H. Songoro, M. Vogel, and Z. Cendes, "Keeping time with maxwell's equations," *Microwave Magazine, IEEE*, vol. 11, no. 2, pp. 42–49, 2010.

[40] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer, 2007, vol. 54.

[41] M. J. White, Z. Yun, and M. F. Iskander, "A new 3d fdtd multigrid technique with dielectric traverse capabilities," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 49, no. 3, pp. 422–430, 2001.

[42] P. Silvester, "Finite element solution of homogeneous waveguide problems," *Alta Frequenza*, vol. 38, pp. 313–317, 1969.

[43] K. Hayata, M. Koshiba, M. Eguchi, and M. Suzuki, "Vectorial finite-element method without any spurious solutions for dielectric waveguiding problems using transverse magnetic-field component," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 34, no. 11, pp. 1120–1124, 1986.

[44] M. Koshiba, Y. Tsuji, and M. Hikari, "Time-domain beam propagation method and its application to photonic crystal circuits," *Journal of lightwave technology*, vol. 18, no. 1, p. 102, 2000.

[45] T. W. J.S. Hesthaven, "High-order/spectral methods on unstructured grids i. time-domain solution of maxwell's equations," ICASE NASA Langley Research Center, Hampton, Virginia, Tech. Rep. 2001-6, March 2001.

[46] S. Gedney and U. Navsariwala, "An unconditionally stable finite element time-domain solution of the vector wave equation," *Microwave and Guided Wave Letters, IEEE*, vol. 5, no. 10, pp. 332–334, 1995.

[47] J.-M. Jin, *The finite element method in electromagnetics*. John Wiley & Sons, 2014.

[48] F. M. Tesche, M. Ianoz, and T. Karlsson, *EMC analysis methods and computational models*. John Wiley & Sons, 1997.

[49] J. Mackerle, "2d and 3d finite element meshing and remeshing: A bibliography (1990-2001)," *Engineering computations*, vol. 18, no. 8, pp. 1108–1197, 2001.

[50] W. Thacker, "A brief review of techniques for generating irregular computational grids," *International Journal for Numerical Methods in Engineering*, vol. 15, no. 9, pp. 1335–1341, 1980.

[51] D. S. Lo, *Finite Element Mesh Generation*. CRC Press, Jan 2015.

[52] O. Zienkiewicz and D. Phillips, "An automatic mesh generation scheme for plane and curved surfaces by 'isoparametric'co-ordinates," *International Journal for Numerical Methods in Engineering*, vol. 3, no. 4, pp. 519–528, 1971.

[53] G. L. Dirichlet, "Über die reduction der positiven quadratischen formen mit drei unbestimmten ganzen zahlen." *Journal für die reine und angewandte Mathematik*, vol. 40, pp. 209–227, 1850.

[54] G. Voronoi, "New parametric applications concerning the theory of quadratic forms-second announcement," *Journal Fur Die Reine Und Angewandte Mathematik*, vol. 134, pp. 198–287, 1908.

[55] B. Delaunay, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, no. 793-800, pp. 1–2, 1934.

[56] D. F. Watson, "Computing the n-dimensional delaunay tessellation with application to voronoi polytopes," *The computer journal*, vol. 24, no. 2, pp. 167–172, 1981.

[57] A. Bowyer, "Computing dirichlet tessellations," *The Computer Journal*, vol. 24, no. 2, pp. 162–166, 1981.

[58] J. C. Cavendish, "Automatic triangulation of arbitrary planar domains for the finite element method," *International Journal for Numerical Methods in Engineering*, vol. 8, no. 4, pp. 679–696, 1974.

[59] C. Lawson, "Software for c1 surface interpolation, math," *Software Syrup*, 1977.

[60] S. Lo, "A new mesh generation scheme for arbitrary planar domains," *International Journal for Numerical Methods in Engineering*, vol. 21, no. 8, pp. 1403–1426, 1985.

[61] M. S. Shephard and M. A. Yerry, "Approaching the automatic generation of finite element meshes," *Computers in Mechanical Engineering*, vol. 1, no. 4, pp. 49–56, 1983.

[62] M. A. Yerry and M. S. Shephard, "Automatic three-dimensional mesh generation by the modified-octree technique," *International Journal for Numerical Methods in Engineering*, vol. 20, no. 11, pp. 1965–1990, 1984.

[63] S. M. R. Kabir, B. Rahman, A. Agrawal, and K. T. V. Grattan, "Elimination of numerical dispersion from electromagnetic time domain analysis by using resource efficient finite element technique," *Progress In Electromagnetics Research*, vol. 137, pp. 487–512, 2013.

[64] O. M. Shir *et al.*, *Niching in derandomized evolution strategies and its applications in quantum control*. Natural Computing Group, LIACS, Faculty of Science, Leiden University, 2008.

[65] G. Mur, "Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations," *Electromagnetic Compatibility, IEEE Transactions on*, no. 4, pp. 377–382, 1981.

[66] U. S. Inan and R. A. Marshall, *Numerical electromagnetics: the FDTD method*. Cambridge University Press, 2011.

[67] A. Bayliss and E. Turkel, "Radiation boundary conditions for wave-like equations," *Communications on Pure and applied Mathematics*, vol. 33, no. 6, pp. 707–725, 1980.

[68] R. L. Higdon, "Numerical absorbing boundary conditions for the wave equation," *Mathematics of computation*, vol. 49, no. 179, pp. 65–90, 1987.

[69] J. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of computational physics*, vol. 114, no. 2, pp. 185–200, 1994.

[70] M. Kuzuoglu and R. Mittra, "Frequency dependence of the constitutive parameters of causal perfectly matched anisotropic absorbers," *Microwave and Guided Wave Letters, IEEE*, vol. 6, no. 12, pp. 447–449, 1996.

[71] J. A. Roden and S. D. Gedney, "Convolutional pml (cpml): An efficient fdtd implementation of the cfs-pml for arbitrary media," *Microwave and optical technology letters*, vol. 27, no. 5, pp. 334–338, 2000.

[72] Y. Hao and R. Mittra, *FDTD modeling of metamaterials*. Artech house, 2009.

[73] V. Veselago *et al.*, "The electrodynamics of substances with simultaneously negative values of ε and μ," *Physics-Uspekhi*, vol. 10, no. 4, pp. 509–514, 1968.

[74] D. R. Smith, W. J. Padilla, D. Vier, S. C. Nemat-Nasser, and S. Schultz, "Composite medium with simultaneously negative permeability and permittivity," *Physical review letters*, vol. 84, no. 18, p. 4184, 2000.

[75] J. Pendry, A. Holden, W. Stewart, and I. Youngs, "Extremely low frequency plasmons in metallic mesostructures," *Physical review letters*, vol. 76, no. 25, p. 4773, 1996.

[76] D. R. Smith and N. Kroll, "Negative refractive index in left-handed materials," *Physical Review Letters*, vol. 85, no. 14, p. 2933, 2000.

[77] R. Ziolkowski and E. Heyman, "Wave propagation in media having negative permittivity and permeability," *Physical review E*, vol. 64, no. 5, p. 56625, 2001.

[78] R. Ziolkowski, "Pulsed and cw gaussian beam interactions with double negative metamaterial slabs," *Opt. Express*, vol. 11, pp. 662–681, 2003.

[79] J. Berenger, "Perfectly matched layer for the fdtd solution of wave-structure interaction problems," *Antennas and Propagation, IEEE Transactions on*, vol. 44, no. 1, pp. 110–117, 1996.

[80] D. Leung, N. Kejalakshmy, B. Rahman, and K. Grattan, "Rigorous modal analysis of silicon strip nanoscale waveguides," *Optics Express*, vol. 18, no. 8, pp. 8528–8539, 2010.

[81] E. Kirby, J. Hamm, K. Tsakmakidis, and O. Hess, "Fdtd analysis of slow light propagation in negative-refractive-index metamaterial waveguides," *Journal of Optics A: Pure and Applied Optics*, vol. 11, no. 11, p. 114027, 2009.

[82] C. Themistos and B. Rahman, "Design issues of a multimode interference-based 3-db splitter," *Applied optics*, vol. 41, no. 33, pp. 7037–7044, 2002.

[83] A. Ferreras, F. Rodŕiguez, E. Gomez-Salas, J. De Miguel, and F. Hernandez-Gil, "Useful formulas for multimode interference power splitter/combiner design," *Photonics Technology Letters, IEEE*, vol. 5, no. 10, pp. 1224–1227, 1993.

[84] J. Juntunen and T. Tsiboukis, "Reduction of numerical dispersion in fdtd method through artificial anisotropy," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 48, no. 4, pp. 582–588, 2000.

[85] *Intel® 64 and IA-32 Architectures Optimization Reference Manual*, Intel Corporation, 2014.

[86] "http://www.top500.org," Tech. Rep., June 2014. [Online]. Available: http://www.top500.org

[87] L. H. Jamieson, P. T. Mueller, and H. J. Siegel, "Fft algorithms for simd parallel processing systems," *Journal of Parallel and Distributed Computing*, vol. 3, no. 1, pp. 48–71, 1986.

[88] S. Agaian and D. Gevorkian, "Synthesis of a class of orthogonal transforms. parallel simd-algorithms and specialized processors," *Pattern Recognition and Image Analysis*, vol. 2, no. 4, pp. 394–408, 1992.

[89] Y. Ben-Asher, D. Egozi, and A. Schuster, "2-d simd algorithms for perfect shuffle networks," *Journal of Parallel and Distributed Computing*, vol. 16, no. 3, pp. 250–257, 1992.

[90] J. Apostolakis, P. Coddington, and E. Marinari, "New simd algorithms for cluster labeling on parallel computers," *International Journal of Modern Physics C*, vol. 4, no. 04, pp. 749–763, 1993.

[91] H. Chen, N. S. Flann, and D. W. Watson, "Parallel genetic simulated annealing: a massively parallel simd algorithm," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 9, no. 2, pp. 126–136, 1998.

[92] I. Hong, S. Chung, H. Kim, Y. Kim, Y. Son, and Z. Cho, "Ultra fast symmetry and simd-based projection-backprojection (ssp) algorithm for 3-d pet image reconstruction," *Medical Imaging, IEEE Transactions on*, vol. 26, no. 6, pp. 789–803, 2007.

[93] F. Goualard, "Fast and correct simd algorithms for interval arithmetic," in *PARA'08*. Springer, 2010.