A Software Platform For The Simulation Of Light Propagation In Photonic Crystal Structures With Defects

Santiago Echeverri Chacón
sechev14@eafit.edu.co

Departamento de Ciencias Básicas

Escuela de Ciencias y Humanidades

Universidad EAFIT

Medellín, Colombia

May 2013

A Software Platform For The Simulation Of Light Propagation In Photonic Crystal Structures With Defects

Santiago Echeverri Chacón
sechev14@eafit.edu.co

Advisor: Nicolás Guarín Zapata
nicoguarin@gmail.com

Thesis document presented as a partial prerequisite for the title of Bachelor in Engineering Physics

Departamento de Ciencias Básicas

Escuela de Ciencias y Humanidades

Universidad EAFIT

Medellín, Colombia

May 2013

Contents

Acknowledgements **9**

Introduction **11**

1 Problem **13**

1.1 Problem statement 13

1.2 Objectives 13

1.2.1 General objectives 13

1.3 Justification 14

2 Electromagnetic waves in periodic media **17**

2.1 Maxwell equations 17

2.2 Wave equation for electric fields 19

2.3 Time harmonic fields, reciprocal space and Fourier transform 21

2.4 Electromagnetism as an eigenvalue problem 22

2.4.1 Hermiticity 22

2.4.2 Symmetry groups 24

2.4.3 Discrete translational symmetries 25

2.4.4 Conservation of energy for waves in source less media 28

2.4.5 Energy functional of electromagnetic waves 28

3 Finite element method **30**

3.1 Weak formulation of the problem . . . 31

3.2 Boundary conditions . . . 32

3.3 Abstract form of the equation . . . 33

3.4 Base functions and discretization . . . 34

3.5 Edge elements vs Node Elements . . . 41

3.6 Time domain formulation . . . 41

3.6.1 Weak formulation in time dependent problem . . . 42

3.6.2 Boundary conditions in time dependent problem . . . 42

3.6.3 Base functions and abstract form . . . 43

3.7 Mass matrices . . . 45

3.8 Explicit formulation . . . 45

4 Implementation **47**

4.1 Object Oriented Paradigm . . . 47

4.2 General stages in a simulation . . . 49

4.2.1 Pre-Processing . . . 49

4.2.2 Analysis . . . 50

4.2.3 Post-Processing . . . 50

4.3 Classes, Diagrams and flow charts of PeyeQM . . . 50

4.4 PeyeQM usage . . . 51

4.5 Python . . . 55

4.6 Gmsh . . . 56

4.7 Paraview . . . 56

5 Results **58**

5.1 Electrostatic benchmark tests . . . 58

5.1.1 Electric field due to charged elements . . . 58

5.2 Harmonic benchmark tests 64

5.2.1 Eigenvalues and modes in wave-guides 64

5.2.2 Bloch periodic lattices and dispersion curves 69

6 Conclusions and future work **74**

6.1 Conclusions 74

6.2 Future work 75

References **77**

List of Figures

1 Computational electromagnetism is a highly interdisciplinary field. Taken from [1] 11

2 a) Line defects acting as a wavegide [1]. b) Point defects where resonant modes are confined [2]. 12

1.1 Maxwell equations by the hand of numerical methods have the power to impact many scientific and technological areas. Taken from Jin et al [1] 15

2.1 Illustration of a bi dimensional point lattice: a) Section of a square lattice with primitive lattice vectors a_1 and a_2 b) and one of the possible correspondent unitary cells where corners of four points are shared by one cell. Taken with permission from [3]. 26

2.2 Illustration of a bi dimensional point lattice in the reciprocal domain: a) Section of a square lattice with primitive reciprocal lattice vectors b_1 and b_2 b) and a reciprocal unitary cells best known as the first Brillouin zone. Taken with permission from [3]. 27

3.1 Abstract representation of a simulation domain Ω and its boundary Γ 31

3.2 Discretized domain defined by a set of elements and their nodes. Taken with permission from [3] 34

3.3 Three examples of commonly used two-dimensional elements. The one on the center is a 9 node quadrilateral element with a complete basis. To the left is a serendipity QUAD8 elements with 8 nodes, and to the right a 6 node triangular element.[4] . . 35

3.4 In order to ease the definition of base functions, every element is transformed to a generalized "isoparametric element" by means of a function $g(x, y)$ [4] so that all elements share the same base. The element on the left has an arbitrary shape and position, and by means of function $g(x, y)$ is mapped to a standard element in a domain in r, s that goes from -1 to 1 35

4.1 Representation of how much of the simulation process is performed by using PeyeQM's routines. 49

4.2 Diagram of general simulation stages as abstracted by PeyeQM. 50

4.3 This diagram shows the main classes defined PeyeQM. Classes that are called by a method are connected with lines that end with a black diamonds, and classes that are attributes of other classes are connected by white diamonds. 52

4.4 This diagram shows the classes defined in the module gmsh_library.py. Classes that are called by a method are connected with lines that end with a black diamonds, inheritance is shown with a white arrow, and classes that are attributes of other classes are connected by white diamonds. 57

5.1 Cross sectional view of parallel plate capacitor. 60

5.2 Solution of electric field in parallel plate capacitor. 60

5.3 Whole simulation of Electric field due to a charged cylinder. Left, Numerical result for the x component of the field. Middle, x component of the solution, and, right, mesh and error calculated between analytic formula and results. 61

5.4 Segment of the simulation for Electric field due to a charged cylinder. a) Numerical result for the y component of the field, b) x component of the solution, c) Mesh and error calculated between analytic formula and results. 62

5.5 Components of the electric field after simulating a problem of two cylinders with opposite charges. 63

5.6 Arrow glyphs of the solution for the dipole problem. Length and color of the arrow indicate magnitude of the field. As expected, the field points from the positively charged cylinder to the negative. 63

5.7 Error calculation of simulation against exact solution. The method reproduces the nature of the problem, but a finer mesh is needed to obtain precise results. 63

5.8 Results for three different modes from the simulation of a square shaped wave-guide with side $a = b = 2$ 66

5.9 Results for three different modes from the simulation of a rectangle shaped waveguide with sides $a = 4$ and $b = 2$. 66

5.10 Results for three different modes from the simulation of circular wave-guide with radius $a = 2$. 67

5.11 Degeneracy of modes from the simulation of hexagonal wave-guide. 67

5.12 Results for the first 8 modes from the simulation of an elliptical wave-guide with minor axis $a = 2$ and mayor axis $b = 3$. 68

5.13 First 4 modes from the simulation of two isoespectral shapes as taken from [5]. . . 68

5.14 A lattice of dielectric rods embedded in a domain causes peaks of field magnitude in the places with higher dielectric constant. 70

5.15 Illustration of reduced Brillouin zone in spectral domain, and reference points Γ, χ and M. 71

5.16 Dispersion plot for a square lattice of r/a = 0.1 and $\epsilon_r = 8.9$. 71

5.17 Dispersion plot for a square lattice of r/a = 0.2 and $\epsilon_r = 8.9$. 72

5.18 Dispersion plot for a square lattice of r/a = 0.4 and $\epsilon_r = 8.9$. 72

5.19 Frequency surfaces for square lattice of 5.17, where each point in a surface i represents the eigenvalue i of the solution, given a pair of $\vec{k}_x$, $\vec{k}_y$. 73

"The first principle is that you must not fool yourself - and you are the easiest person to fool..."
Richard P. Feynman

Acknowledgements

I would like to thank my family for their support and inspiration, specially my mom, who planted in me the seed of scientific curiosity, my big sister who I admire, and my dad who has been of great company and support for these years.

I also owe my gratitude to my friend and advisor Nicolás Guarín Zapata who accompanied me through the making this project. His endless patience, time and knowledge were invaluable resources.

To all the professors of the Engineering Physics Program for my academic and personal formation, and particularly to professors Mario Elkin Vélez and Juan Manuel Jaramillo, who fed my curiosity with interesting conversations and research opportunities.

Introduction

This document describes the process of understanding and designing a computational tool for the simulation of electromagnetic waves in periodic crystals. It gives account of the theory and concepts that are necessary for modeling electromagnetic fields using the Finite Element Method (FEM), and goes further by presenting the approach we used to implement it in the form of a software platform based in the Object Oriented Programming Paradigm. The document concludes by showing a set of results from simulations obtained by using the program, and compares these to analytic solutions and also to Photonic Crystal simulations taken from the literature. Good agreement with the references is found, and the software platform proved to be a promising tool for the modeling of Photonic Crystal devices.

The project is thus, framed in the context of a field known as computational electromagnetism, which can bee seen as an intersection between the theory of Electromagnetic Physics, the mathematics of numerical methods and application of techniques and devices product of computer science. All of this together, gives rise to the computational platforms that enable engineers to model, design, and test novel electromagnetic components before actually making them. Saving thus, production costs and optimizing functionality. In our case, the novel technology that the tool we design is allowing to test is that of **Photonic Crystals**. Photonic Crystals are materials designed to have light manipulation properties capable of revolutionizing the field of telecommu-

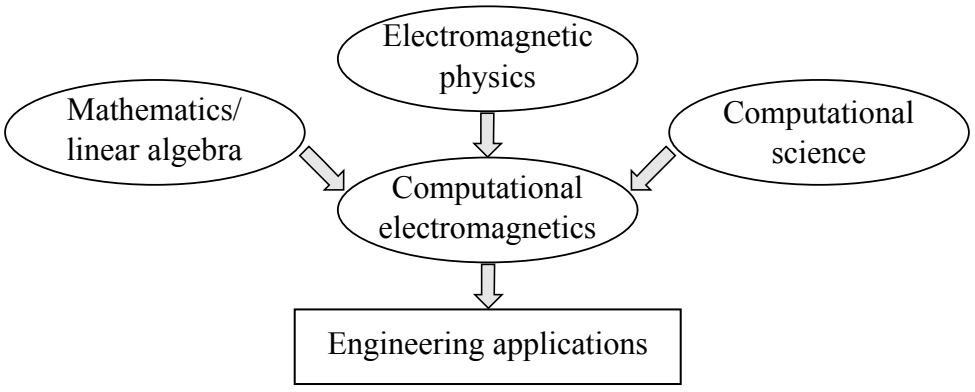

Figure 1: Computational electromagnetism is a highly interdisciplinary field. Taken from [1]

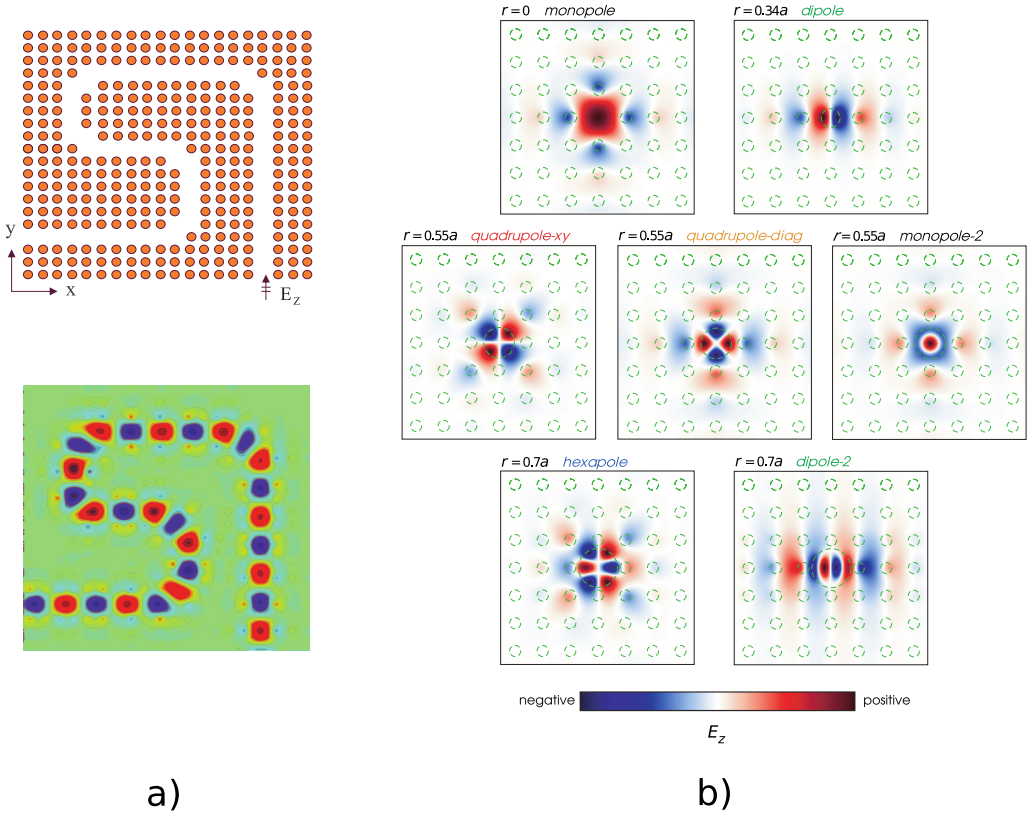

Figure 2: a) Line defects acting as a wavegide [1]. b) Point defects where resonant modes are confined [2].

nications and signal processing. With them, we can design devices that are able to confine, and mold light (like in figure 2) in ways that are similar to how electronic circuits manipulate charges and currents. Opening the doors to realizable photonic circuits and new technologies.

Chapter 1

Problem

1.1 Problem statement

In this project we proposed to develop a simulation software package that allows analysis of electromagnetic (EM) wave propagation, in the context of Photonic Crystal (PC) defects modelling and design.

1.2 Objectives

The following were the objectives as stated in the project proposal document:

1.2.1 General objectives

- Build a software platform capable of simulating electromagnetic field propagation in perfect PC and also in PC with defects such as cavities and inclusions.
- The platform architecture must be structured in a way that facilitates future upgrading, possible optimization schemes, and integration with other kinds of simulations like confinement of electrons in potential wells and crystals.

Specific objectives

1. Apprehend the theory behind EM fields and its applications in photonics. Particularly:
   (a) general overview of computational methods in electromagnetism and deeper understanding of methods that fit the problem statement.

   (b) Photonic crystals and the mathematical tools for modelling periodicity, as well as

   (c) technological applications of defects in Photonic Crystals and common case studies.

2. Define a set of implementation requirements and constraints, in terms of the numerical method, and expected functionality.

3. Implement algorithms to solve problems that relate to technological applications studied in the appropriation stage.

   (a) Solve simple scalar-static 2D problems

   (b) Implement vector field solver for stationary non periodic conditions (this may qualify as spectral solution or not)

   (c) Vector field solutions for time dependent problems

   (d) Introduce periodic conditions

4. Document and share the results of the simulations making an analysis of the relevance and feasibility of the platform.

1.3 Justification

Most of the artificial environment we as a society have built to live in, is in one way or another a consequence of our ability to understand and transform materials present in nature. This understanding is so important that human history studies have come to address cultural stages of our evolution from: from the Stone Age through Bronce Age, as the materials we used to overcome survival challenges [2]. Even in recent years, extraordinary changes in the way we evolve as a species can be attributed to achievements only possible by improvements on our understanding of material properties. Take for example the paradigm change caused by the discovery of electronic properties of materials such as semiconductors [6]. And the ever growing stream of technological applications derived from it.

The next frontier in the context of material science, relies in a level of understanding that allows us to not only use and transform materials, but to design them. This is the field of metamaterials, a field in which by means of manipulating the microscopical structure and geometry of a crystal-like solid, we can obtain macroscopic properties that are otherwise impossible to find in nature [7, 8]. One of the sub domains of the metamaterials frontier is that of media capable of molding and manipulating electromagnetic waves. The ability to harness light has the prospect to support a new technological revolution of a scale similar to that of transistor revolution [2]. Advances in engineered optical materials will benefit the development of areas such as high speed computing, spectroscopy, laser engineering, biomedicine and quantum computing [6]. This project concerns about one candidate of optical metamaterial set known as Photonic Crystals, and particularly, the light guiding properties that arise when taking advantage of lattice cavities. Photonic crystals are:

"Artificially created periodic low-loss dielectric medium in which electromagnetic waves of certain frequencies cannot propagate "[9].

Light in that certain frequency gap is virtually stopped because of local resonances linked to the periodicity of dielectric dispersers. In a way it can be seen as destructive interference caused by the geometry of dispersers. This property is used in technical application for many forms of technological developments. For example Kärtner's et al. developed high resolution PC based Analogic-Digital (AD) conversion circuits [10] that are showing all-optical circuits which outperform electronic based components. Telecommunications also benefit from technologies such as ultra low loss Photonic Crystal Fibers [11]. As well as low cost, and high definition spectroscopy technologies that take advantage of spectral selectivity in PCs; Pervez [12] has used a phenomenon called out-coupling to fabricate cheap PC based spectrometers. On the other hand, a lot of effort has been made to exploit the light confinement properties of PCs to make micro laser cavities and even single-photon sources. Theoretical background and numerical studies on spontaneous emission using microcavities in PCs has been undertaken by many researchers including works from Yablonovich and Vileneuve [13, 14, 15]. Experimental measures of ultra-fast nano-cavity lasers was performed by Hatice [16] in 2006, and Moore [17] in 2008. Even state of the art quantum computing experiments are being possible by using PC based waveguides and cavities as shown by Wolters in [18].

The nature of most PC application problems, is of such complexity that very often numerical tools are the only way of modelling new designs and performance. Figure 1.1 shows some of the fields that benefit from electromagnetic numerical simulations as stated by Jin [1]. Computational methods such as: Plane Wave Expansion (PWE) [19, 20], Finite Difference Time Domain (FDTD) [11, 21], Method of Moments (MoM) or Boundary Element Method (BEM) and Finite Element Methods (FEM)[22, 23], have had their share in the modelling of electromagnetic fields under many different scenarios. Every one of them having their own strengths and weaknesses in terms of: ease of implementation, precision, and use of computational resources.

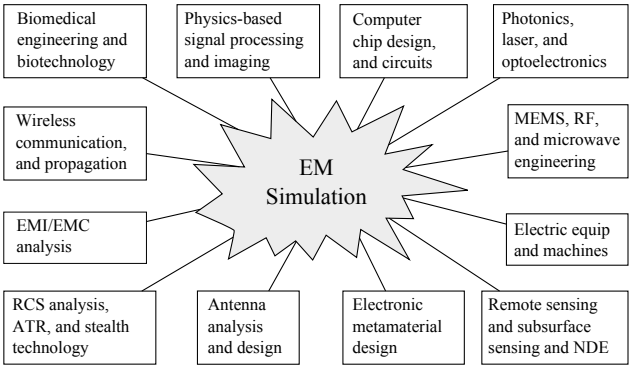

Figure 1.1: Maxwell equations by the hand of numerical methods have the power to impact many scientific and technological areas. Taken from Jin et al [1]

.

In the context of Universidad EAFIT, and specifically of Engineering Physics, we see a growing

research background in computational physics that goes from modelling seismic waves [3], and resonant modes in musical instruments [24]; passing through quantum mechanics like relativistic phenomena in graphene [25], and quantum potential wells and crystals [26]; to finally address a computational module for digital holography and speckle [27], and research on Plane Wave Methods for calculation of bandgap structures in Photonic Crystals [20]. It can be stated that this trend in numerical methods applied to crystal-like materials has been embraced and supported by the Computational and Theoretical Physics Research Initiative of engineering physics students (SFTyC). This work expands and continues the initiative towards strengthening undergraduate research and software development abilities.

Chapter 2

Electromagnetic waves in periodic media

This chapter focuses on basic concepts and principles related to electromagnetic fields and waves based on Maxwell's equations. More over, some emphasis will be taken on stating these equations as a linear Hermitian Eigenvalue problem that has many common points with the Schrödinger equation and its application to crystals in the field of Solid State Physics [28]. In this way, propagation of light in a PC can be defined using linear operators acting over wave functions that are solution to a given eigenvalue problem. Periodicity is then invoked by means of discrete translational symmetry operators, and their associated phase shift eigenvalue.

2.1 Maxwell equations

The macroscopic behavior of electromagnetism, including propagation of light in Photonic Crystals (PCs) are governed by **Maxwell equations**. Maxwell's equations are four (vectorial) equations that relate the electric and magnetic fields to their respective sources, i.e. electric charges and currents. They were established by James Clerk Maxwell (1831-1879) based on experimental discoveries of André-Marie Ampere (1775-1836) and Michael Faraday (1791-1867), and a law of electricity made by Carl Friederich Gauss (1777-1855) [1].

These equations are herein expressed in both their Integral and Differential forms. Further information on their derivation can be found in classical textbooks on Electromagnetism (e.g. [29]).

Maxwell equations in integral form are closer to the fundamental postulates that inspired them and use integration of the fields over volumes surfaces and closed paths to relate fields and their

sources. They are:

$$\oint_C \mathbf{E} \cdot d\mathbf{l} = -\frac{d}{dt} \int_S \mathbf{B} \cdot d\mathbf{S} \qquad \text{(Faraday)} \tag{2.1}$$

$$\oint_C \mathbf{H} \cdot d\mathbf{l} = \frac{d}{dt} \int_S \mathbf{D} \cdot d\mathbf{S} + \int_S \mathbf{J} \cdot d\mathbf{S} \qquad \text{(Ampere-Maxwell)} \tag{2.2}$$

$$\int_S \mathbf{D} \cdot d\mathbf{S} = \int_V \rho dV \qquad \text{(Gauss)} \tag{2.3}$$

$$\int_S \mathbf{B} \cdot d\mathbf{S} = 0 \qquad \text{(Gauss for magnetism),} \tag{2.4}$$

In differential form Maxwell's equations are as follows

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \qquad \text{(Faraday)} \tag{2.5}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} \qquad \text{(Ampere-Maxwell)} \tag{2.6}$$

$$\nabla \cdot \mathbf{D} = \rho \qquad \text{(Gauss)} \tag{2.7}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \text{(Gauss for magnetism)} \tag{2.8}$$

$$\nabla \cdot \mathbf{J} = -\frac{\partial \rho}{\partial t} \qquad \text{(Continuity),} \tag{2.9}$$

In both set of equation we have

- **E**: Electric field intensity (volts/meter),
- **D**: Electric flux density (coulombs/meter^2),
- **H**: Magnetic field intensity (amperes/meter),
- **B**: Magnetic flux density (webers/meter),
- **J**: Electric current density (amperes/meter^2),
- ρ: Electric charge density (coulombs/meter^3).

Even though they may appear in this document it is important to say that in the implementation and simulations no current or charge densities are considered so $\rho = 0$ and $\mathbf{J} = 0$, what is common for the case of electromagnetic waves far away from the sources. Being structures from which light must propagate, Photonic Crystals are made of dielectric, translucent materials that have a given permeability or dielectric constant called ϵ. When fields propagate through dielectric media, the dielectric gets electrically polarized and this affects the field. A polarization field known as polarization vector **P** appears and then the electric flux density is defined for a material with dielectric constant ϵ. A similar phenomenon happens with magnetic fields and materials that get magnetized, and an equivalent constitutive relation is constructed but now with the parameter μ

or permeability. So, the two constitutive relations in EM are:

$$\mathbf{D} = \epsilon_0 \mathbf{E} + \mathbf{P} \tag{2.10}$$

$$\mathbf{B} = \mu_0 \mathbf{H} + \mathbf{M}, \tag{2.11}$$

where $\mathbf{P}$ is the polarization vector that tell us information about the response of the material to the external field due to orientation of the molecules inside of it, and $\mathbf{M}$ is the magnetization vector that is the analogous of the polarization for the magnetic case. If the fields are small enough the behavior of the material is linear and we can express the polarization and magnetization vectors as linear functions of $\mathbf{E}$ and $\mathbf{H}$

$$\mathbf{D} = \bar{\bar{\epsilon}} \cdot \mathbf{E} \tag{2.12}$$

$$\mathbf{B} = \bar{\bar{\mu}} \cdot \mathbf{H}, \tag{2.13}$$

where the double bar refers to a second order tensor, and $\cdot$ is a tensor-vector product. A tensorial formulation is necessary if the material is anisotropic, this is, if its properties vary depending of the direction from where you look at them. If the material is isotropic (its the same from every angle), then we can assume $\bar{\bar{\epsilon}}$ and $\bar{\bar{\mu}}$ as scalars. Thorough the following chapters, and in the implementation of the software, these two quantities are assumed as scalars.

Another important fact to note is that PC's do not generally involve magnetic constitutive materials, so from here on we will assume no magnetization $\mathbf{M} = 0$, $\mu = \mu_0$.

2.2 Wave equation for electric fields

As we can see in equations 2.5 2.6 the electric and magnetic fields are coupled. One of the great achievements from Maxwell's work on stating these equations, was the discovery that the right operations for the uncoupling of these equations leads to a wave equation that explains electromagnetic radiation.

If we take the curl to (2.5), we get

$$\nabla \times \nabla \times \mathbf{E} = -\frac{\partial}{\partial t} \nabla \times \mathbf{B} = -\frac{\partial}{\partial t} \nabla \times (\bar{\bar{\mu}} \cdot \mathbf{H}) \ ,$$

assuming a (piece wise[1]) homogeneous material,

$$\begin{aligned} \nabla \times \nabla \times \mathbf{E} = -\frac{\partial}{\partial t} \nabla \times \mathbf{B} &= \frac{\partial}{\partial t} (\bar{\bar{\mu}} \cdot \nabla \times \mathbf{H}) \\ &= -\frac{\partial}{\partial t} \left(\bar{\bar{\mu}} \cdot \left[\frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} \right] \right) \ , \end{aligned}$$

[1] Because in our formulation of the Finite Element we need the functions to be smooth (in the first derivative) only inside elements

and if the polarization and magnetization do not vary with time –there is not hysteresis– we have:

$$\nabla \times \nabla \times \mathbf{E} = -\bar{\bar{\mu}} \cdot \left[\frac{\partial^2 \mathbf{D}}{\partial t^2} + \frac{\partial \mathbf{J}}{\partial t} \right]$$
$$\nabla \times \nabla \times \mathbf{E} = -\bar{\bar{\mu}} \cdot \left[\bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathbf{E}}{\partial t^2} + \frac{\partial \mathbf{J}}{\partial t} \right] \quad . \tag{2.14}$$

Similarly, if we take the curl to (2.6), under the same assumptions we get

$$\begin{aligned} \nabla \times \nabla \times \mathbf{H} &= \frac{\partial \nabla \times \mathbf{D}}{\partial t} + \nabla \times \mathbf{J} \\ &= \frac{\partial \nabla \times (\bar{\bar{\epsilon}} \cdot \mathbf{E})}{\partial t} + \nabla \times \mathbf{J} \\ &= \frac{\partial \bar{\bar{\epsilon}} \cdot \nabla \times \mathbf{E}}{\partial t} + \nabla \times \mathbf{J} \\ &= \frac{\partial \bar{\bar{\epsilon}} \cdot \left(-\frac{\partial \mathbf{B}}{\partial t}\right)}{\partial t} + \nabla \times \mathbf{J} \quad , \end{aligned}$$

and thus

$$\nabla \times \nabla \times \mathbf{H} = -\bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathbf{B}}{\partial t^2} + \nabla \times \mathbf{J}$$
$$\nabla \times \nabla \times \mathbf{H} = -\bar{\bar{\epsilon}} \cdot \bar{\bar{\mu}} \cdot \frac{\partial^2 \mathbf{H}}{\partial t^2} + \nabla \times \mathbf{J} \quad . \tag{2.15}$$

If we apply the vector identity $\nabla \times \nabla \times \mathbf{A} = \nabla(\nabla \cdot \mathbf{A}) - \nabla^2 \mathbf{A}$, and assume that we do not have electrical charges we can rewrite (2.14) and (2.15) as

$$\nabla^2 \mathbf{E} = \bar{\bar{\mu}} \cdot \bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathbf{E}}{\partial t^2} - \bar{\bar{\mu}} \cdot \frac{\partial \mathbf{J}}{\partial t} \tag{2.16}$$
$$\nabla^2 \mathbf{H} = \bar{\bar{\epsilon}} \cdot \bar{\bar{\mu}} \cdot \frac{\partial^2 \mathbf{H}}{\partial t^2} - \nabla \times \mathbf{J} \tag{2.17}$$

Neglecting terms $\frac{\partial \mathbf{J}}{\partial t}$ and $\nabla \times \mathbf{J}$ –the source terms in the equations– and assuming isotropy we get the well known expressions:

$$\left(\nabla^2 - \mu\epsilon \frac{\partial^2}{\partial t^2} \right) \mathbf{E} = \mathbf{0} \tag{2.18}$$
$$\left(\nabla^2 - \mu\epsilon \frac{\partial^2}{\partial t^2} \right) \mathbf{H} = \mathbf{0} \quad , \tag{2.19}$$

that are vectorial wave equations with phase speed $c = \sqrt{\epsilon\mu}$.

This form of (2.14) is relevant for us because the program is currently based on this approximation. This particular topic will be treated furthermore in the implementation section 4.

2.3 Time harmonic fields, reciprocal space and Fourier transform

Linearity of Maxwell equations, and a restriction to time harmonic variations permits the separation of time and spatial dependencies in the form:

$$\mathbf{E}(\mathbf{r},t) = \mathbf{E}(\mathbf{r})e^{-i\omega t}$$

This is known as a phasorial notation, and from here on, when dealing with time harmonic fields we will be interested only on the unknown phasor function $\mathbf{E}(\mathbf{r})$ which will represent the field distribution at a fixed time.

Considering a single harmonic wave propagating with angular frequency ω we get:

$$\nabla \times \nabla \times \mathbf{E} = -\omega^2 \bar{\bar{\mu}} \cdot \bar{\bar{\epsilon}} \cdot \mathbf{E} - i\omega \bar{\bar{\mu}} \cdot \mathbf{J} \tag{2.20}$$

$$\nabla \times \nabla \times \mathbf{H} = -\omega^2 \bar{\bar{\epsilon}} \cdot \bar{\bar{\mu}} \cdot \mathbf{H} + \nabla \times \mathbf{J} \ , \tag{2.21}$$

These are the expressions for the frequency domain. The following relations are useful if the reader is more used to transmission line notation, and one is interested in impedance and wave numbers:

$$\bar{\bar{\mu}} = \mu_0 \bar{\bar{\mu_r}} \tag{2.22}$$

$$\bar{\bar{\epsilon}} = \epsilon_0 \bar{\bar{\epsilon_r}} \tag{2.23}$$

$$\mu_0 = \sqrt{\mu_0 \epsilon_0} \sqrt{\frac{\mu_0}{\epsilon_0}} \tag{2.24}$$

$$k_0 = \omega \sqrt{\mu_0 \epsilon_0} \tag{2.25}$$

$$Z_0 = \sqrt{\frac{\mu_0}{\epsilon_0}} \tag{2.26}$$

$$c = \frac{1}{\sqrt{\mu_0 \epsilon_0}} \tag{2.27}$$

where $\bar{\bar{\mu_r}}$,$\bar{\bar{\epsilon_r}}$ are the relative permeability and permitivity, respectively, k_0, Z_0, and c are the free space wave number intrinsic impedance, and speed of light in free space. Using this we get:

$$\bar{\bar{\mu_r}}^{-1} \nabla \times \nabla \times \mathbf{E} = -\omega^2 \mu_0 \epsilon_0 \bar{\bar{\epsilon_r}} \cdot \mathbf{E} - i\omega\mu_0 \mathbf{J} \tag{2.28}$$

$$\bar{\bar{\mu_r}}^{-1} \nabla \times \nabla \times \mathbf{E} = k_0^2 \bar{\bar{\epsilon_r}} \cdot \mathbf{E} - ik_0 Z_0 \mathbf{J} \tag{2.29}$$

And we actually work with either of these equations:

$$\frac{1}{\mu_r}\nabla^2\mathbf{E} = -\omega^2\mu_0\epsilon_0\epsilon_r\mathbf{E} \tag{2.30}$$

$$\frac{1}{\mu_r}\nabla^2\mathbf{E} = k_0^2\epsilon_r\mathbf{E} \tag{2.31}$$

2.4 Electromagnetism as an eigenvalue problem

The current section treats the harmonic wave equation in a formalism similar to that of quantum mechanics. It might be seen as a sumarized version of the treatment of Joannopoulos [2], and Johnson [30]. Treating the equation like that will help us understand interesting qualities of electric fields and explain how they are related to the properties and geometry of the domain.

Going back to equation 3.4, we can see that the form of the equation is one of a eigenvalue problem. Where a series of operations on a function $\mathbf{E}$ (eigenfunction or eigenvector) gives us the same function multiplied by a constant scalar (eigenvalue). In this case we will label the operator acting on $\mathbf{E}$ as $\hat{\Theta}$ in order to make the equation look like a simpler-looking eigenvalue problem:

$$\hat{\Theta}\mathbf{E} = \left(\frac{\omega}{c}\right)^2\mathbf{E} \tag{2.32}$$

Where:

$$\hat{\Theta}\mathbf{E} \triangleq \bar{\bar{\mu_r}}^{-1}\nabla\times\nabla\times\mathbf{E} \tag{2.33}$$

Here the eigenvectors or phasors $\mathbf{E}$ represent spatial patterns of the harmonic modes, and eigenvalues $\left(\frac{\omega}{c}\right)^2$ give information about the frequency ω (and thus wave number k_0) of such modes.

2.4.1 Hermiticity

As in Quantum Mechanics with the Hamiltonian, some key properties of the eigenfunctions that satisfy equation 2.32 are [2]:

- Have real eigenvalues,
- are orthogonal,
- can be obtained by a variational principle
- and can be catalogued by symmetry properties .

[2]This proposition holds if we are not considering lossy materials, where the properties are treated as complex valued tensor and then the operator is no longer Hermitian.

All of these properties rely on the fact that the linear operator is from a kind known as **Hermitian operator**. To understand why $\hat{\Theta}$ is Hermitian we need first to understand the inner product of two wavefunctions $\mathbf{F}(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$:

$$(\mathbf{F}(\mathbf{r}), \mathbf{G}(\mathbf{r})) \triangleq \int \mathbf{F}^*(\mathbf{r}) \cdot \mathbf{G}(\mathbf{r}) \tag{2.34}$$

where * denotes complex conjugation. From definition 2.34 we can see that: $(\mathbf{F}, \mathbf{G}) = (\mathbf{G}, \mathbf{F})^*$ for and $\mathbf{F}$ and $\mathbf{G}$. Also, $(\mathbf{F}, \mathbf{F})$ is always real and non-negative, and $(\mathbf{F}, \mathbf{F})$ is known as the L^2 norm of $\mathbf{F}$. A normalized wavefunction is one where $(\mathbf{F}, \mathbf{F}) = 1$.

Having defined an inner product for two wavefunctions, we say that any operator $\hat{\Theta}$ is **Hermitian** if $\left(\mathbf{F}, \hat{\Theta}\mathbf{G}\right) = \left(\hat{\Theta}\mathbf{F}, \mathbf{G}\right)$ for any fields $\mathbf{F}$ and $\mathbf{G}$.

Using this, we can check that the operator in equation 2.29 is Hermitian and derive nice properties around that conclusion:

$$\begin{aligned}
\left(\mathbf{F}, \hat{\Theta}\mathbf{G}\right) &= \int \mathbf{F}^* \cdot \frac{1}{\bar{\bar{\mu}}_r} \nabla \times \nabla \times \mathbf{G} && (2.35)\\
&= \int (\nabla \times \mathbf{F})^* \cdot \frac{1}{\bar{\bar{\mu}}_r} (\nabla \times \mathbf{G}) + \int \nabla \cdot \left(\mathbf{F} \times \frac{1}{\bar{\bar{\mu}}_r} \nabla \times \mathbf{G}\right) && (2.36)\\
&= \int \left[\nabla \times \left(\frac{1}{\bar{\bar{\mu}}_r} \nabla \times \mathbf{F}\right)\right]^* \cdot \mathbf{G} + \int \nabla \cdot \left(\nabla \times \mathbf{F} \times \frac{1}{\bar{\bar{\mu}}_r} \nabla \times \mathbf{G}\right) + \int \nabla \cdot \left(\mathbf{F} \times \frac{1}{\bar{\bar{\mu}}_r} \nabla \times \mathbf{G}\right) && (2.37)\\
&= \int \left[\nabla \times \left(\frac{1}{\bar{\bar{\mu}}_r} \nabla \times \mathbf{F}\right)\right]^* \cdot \mathbf{G} && (2.38)\\
&= \left(\hat{\Theta}\mathbf{F}, \mathbf{G}\right) && (2.39)
\end{aligned}$$

Here the vector cross product identity[3] is used twice and surface terms that derive from using the Divergence Theorem are neglected.

The importance of knowing that the operator in the left side of 2.29 is hermitian resides in the fact that Hermitian operators have real numbers as eigenvalues. This can be seen by taking the inner product $\left(\mathbf{E}, \hat{\Theta}\mathbf{E}\right)$ and observing that operation of $\hat{\Theta}$ on one of its eigenvectors $\mathbf{E}$ produces the same eigenvector multiplied by the eigenvalue as shown in 2.32. Then:

$$\left(\mathbf{E}, \hat{\Theta}\mathbf{E}\right) = \left(\frac{\omega}{c}\right)^2 (\mathbf{E}, \mathbf{E})$$

[3] $\nabla \cdot (\mathbf{A} \times \mathbf{B}) = \mathbf{B} \cdot (\nabla \times \mathbf{A}) - \mathbf{A} \cdot (\nabla \times \mathbf{B})$ With $\mathbf{A} = \mathbf{F}$ and $\mathbf{B} = \nabla \times \mathbf{G}$, first and then $\mathbf{A} = \nabla \times \mathbf{F}$ and $\mathbf{B} = \mathbf{G}$

Given that $(\mathbf{E}, \mathbf{E})$ is real, as we mentioned before, when taking the complex conjugate of that product we have:

$$\left(\mathbf{E}, \hat{\Theta}\mathbf{E}\right)^* = \left(\frac{\omega^2}{c^2}\right)^* (\mathbf{E}, \mathbf{E})$$

and being Hermitian:

$$\begin{aligned}\left(\mathbf{E}, \hat{\Theta}\mathbf{E}\right)^* &= \left(\mathbf{E}, \hat{\Theta}\mathbf{E}\right)^* \\ \left(\frac{\omega^2}{c^2}\right)^* (\mathbf{E}, \mathbf{E}) &= \left(\frac{\omega^2}{c^2}\right) (\mathbf{E}, \mathbf{E})\end{aligned}$$

Speed of light c is real, so $\omega^2 = \left(\omega^2\right)^*$ must be real.

2.4.2 Symmetry groups

As in Quantum Mechanics and any other eigen-problems, we must introduce the concept of orthogonality in order to understand other operations. We say that two wavevectors are **orthogonal** if $(\mathbf{E}, \mathbf{E}) = 0$, and this happens whenever they have different frequencies. This can be seen as a product of Hermiticity of operator $\hat{\Theta}$ by the following operation on two different modes $\mathbf{E}_1$ and $\mathbf{E}_2$:

$$\begin{aligned}c^2 \left(\mathbf{E}_2, \hat{\Theta}\mathbf{E}_1\right) &= c^2 \left(\hat{\Theta}\mathbf{E}_2, \mathbf{E}_1\right) \\ \omega_1^2 \left(\mathbf{E}_2, \mathbf{E}_1\right) &= \omega_2^2 \left(\mathbf{E}_2, \mathbf{E}_1\right) \\ \left(\omega_1^2 - \omega_2^2\right) \left(\mathbf{E}_2, \mathbf{E}_1\right) &= 0\end{aligned}$$

From this one can see that if two eigenfunctions have different frequencies ($\omega_1^2 - \omega_2^2 \neq 0$) then $(\mathbf{E}, \mathbf{E}) = 0$ must be true and they are orthogonal. Now, if $\left(\omega_1^2 - \omega_2^2\right) = 0$ then $\mathbf{E}_1$ and $\mathbf{E}_2$ are not necessarily orthogonal, and the value is known as a **degenerate** frequency because there is more than one state or wavefunction with eigenvalue $\left(\frac{\omega^2}{c^2}\right)$. In this section we will see that degeneracy is related with symmetries of the domain, and that will be useful for the solution of periodic problems.

Symmetry operations are operations that do not transform the wavefunction of a mode. Symmetries of a problem are such that if a symmetry operation is applied to the wavefunction, it remains the same but multiplied by a scalar. This is called being invariant under the operation. So rotation, inversion, and reflections, are common symmetry operations that can be applied to systems that are symmetric under rotation, inversion and reflection, respectively. Probably the most immediate symmetry operation one can think of is the identity operation: $\hat{E}\mathbf{E}(\mathbf{r}) = \mathbf{E}(\mathbf{r})$, that transform a system into itself.

Another interesting operation is that of Inversion ($\hat{O}_I$), which takes a function $\mathbf{E}(\mathbf{r})$ and inverts its argument: $\hat{O}_I\mathbf{E}(\mathbf{r}) = \mathbf{E}(-\mathbf{r})$. If $\mathbf{E}(-\mathbf{r}) = \mathbf{E}(\mathbf{r})$ we say that the mode is invariant under inversion, or invertible. Finally, the translation operator is one where the argument gets shifted in space

by a given displacement $\mathbf{d}$: $\hat{T}_d\mathbf{E}(\mathbf{r}) = \mathbf{E}(\mathbf{r}-\mathbf{d})$. One function that is invariant over translation operations is a plane wave like e^{ikz}, because operation of $\hat{T}$ over the planewave gives the same wavefunction by a scalar $e^{-ik\mathbf{d}}$:

$$\hat{T}e^{ikz} = e^{ik(z-\mathbf{d})} = e^{-ik\mathbf{d}}e^{ikz}$$

We can see here that this is an eigenvalue problem with eigenfunctions of the form e^{ikz} and complex eigenvalues $e^{-ik\mathbf{d}}$. Now, operators can act on other operators, and they can also be invariant under symmetry operations. If we have an operator $\hat{\Theta}$ whose intrinsic medium properties are characterized by $\bar{\bar{\epsilon}}$ and $\bar{\bar{\mu}}$, and those properties are homogeneous in space, we will be acting on a system with translational symmetry. That is, the operator is the same at different points in space. So, if we have a wave function $\mathbf{E}(\mathbf{r})$ displace it with $\hat{T}_d$, then act on it with $\hat{\Theta}$ and displace it back with an inverted translation operator $\hat{T}_d{}^{-1}$ we will get the same result as if only $\hat{\Theta}$ was used:

$$\hat{\Theta}\mathbf{E}(\mathbf{r}) = \hat{T}_d{}^{-1}\hat{\Theta}\hat{T}_d\mathbf{E}(\mathbf{r})$$
$$\hat{\Theta} = \hat{T}_d{}^{-1}\hat{\Theta}\hat{T}_d$$

Whenever that happens with operators, we can write it in the form:

$$\hat{T}_d\hat{\Theta} - \hat{\Theta}\hat{T}_d = 0$$

and bring the definition of **commutator** between two operators as in Quantum Mechanics:

$$\left[\hat{A},\hat{B}\right] \triangleq \hat{A}\hat{B} - \hat{B}\hat{A} \tag{2.40}$$

If $\left[\hat{A},\hat{B}\right] = 0$ we say that $\hat{\Theta}$ commutes with $\hat{T}_d$, and that implies that that there are wavefunctions common to both. This is useful because sometimes eigenvalues and eigenfunctions of simple symmetry operators are easier to determine than those of $\hat{\Theta}$. The set of symmetry operators that commute with $\hat{\Theta}$ forms the symmetry group of the problem. The fact that plane wave solutions are a solution of the homogeneous isotropic wave equation problem is a consequence of them being solution to a continuous translational symmetry operation that commutes with the operator of the problem. "When one has commuting operators, one can choose simultaneous eiigenvectors of both operators" [2]

2.4.3 Discrete translational symmetries

Photonic crystals, like crystals of atoms or molecules do not have continuous translational symmetry, instead, they have discrete translational symmetry [2]. This means that the ruling operator for the field inside a PC is invariant only for certain translation operations, specifically those in

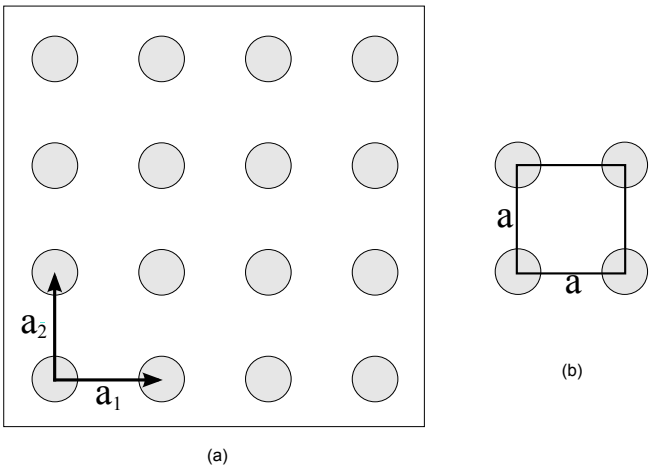

Figure 2.1: Illustration of a bi dimensional point lattice: a) Section of a square lattice with primitive lattice vectors a_1 and a_2 b) and one of the possible correspondent unitary cells where corners of four points are shared by one cell. Taken with permission from [3].

which the distance of translation r' is multiple of some fixed length that we will call a **primitive lattice vector**.

So, if we have a 2D lattice of dielectric rods in air like in figure 2.1 where the dielectric constant is a periodic function of space $\epsilon = \epsilon_0\epsilon_r(r)$[4], then following what we learn in a solid state course we can define the crystal as a base (for example one rod) and a lattice. Which is the set of all discrete translation operations that are a linear combination of multiples of the two primitive translation lattice vectors a_1 and a_2[28]. Or in a mathematical notation:

$$r' = r + u_1 a_1 + u_2 a_2 \tag{2.41}$$

$$\hat{T}_{r'}\mathbf{E}(\mathbf{r}) = \mathbf{R}(\mathbf{r} - \mathbf{r'}) \tag{2.42}$$

where u_1 and u_2 are integers. If $\mathbf{E}$ is a plane wave, then we have

$$\hat{T}_{r'}e^{i\vec{k}\cdot r} = e^{i\vec{k}\cdot(r-r')}$$

$$\hat{T}_{r'}e^{i\vec{k}\cdot r} = e^{i\vec{k}\cdot r}e^{i\vec{k}\cdot r'}$$

with $\vec{k} = \vec{k}_x + \vec{k}_y$ as the wave vector with two components.

$$\hat{T}_{r'}e^{i\vec{k}\cdot r} = e^{i\vec{k}\cdot r}e^{i\vec{k}_x u_1 a_1}e^{i\vec{k}_x u_2 a_2} \tag{2.43}$$

Now, 2.43 does not immediately satisfy the form of equation 2.42 and we don't have an eigenfunction of the lattice. In order to get a solution that is invariant under discrete translations r', the factor

[4]In which ϵ for air is $\epsilon(r) = \epsilon_0$ and inside the rod we have some positive realizable value ϵ_r. And $r = x\hat{x} + y\hat{y}$

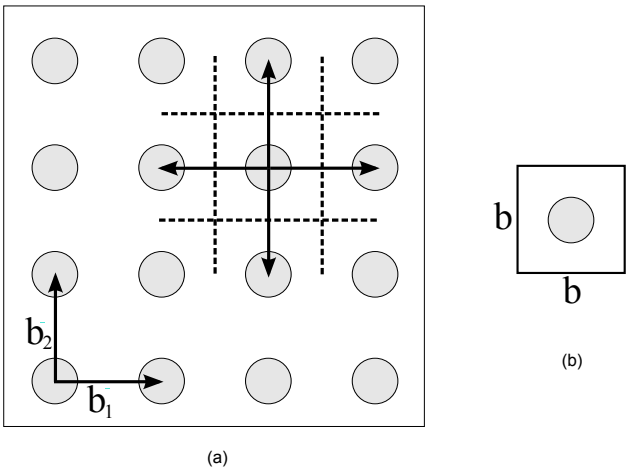

Figure 2.2: Illustration of a bi dimensional point lattice in the reciprocal domain: a) Section of a square lattice with primitive reciprocal lattice vectors b_1 and b_2 b) and a reciprocal unitary cells best known as the first Brillouin zone. Taken with permission from [3].

$e^{i\vec{k}\cdot r'}$ must be equal to 1, and this is obtained when one of two things happen, either $\vec{k}\cdot r' = 0$ or $\vec{k}\cdot r' = 2\pi n$ for $n = 1, 2, 3....$ The first, is a trivial solution where there is no wave, on the other hand, the second is valid for any combination of $\vec{k_x}$ and $\vec{k_y}$ that happens to satisfy $\vec{k_x} = \frac{2\pi m}{a_1}$ and $\vec{k_y} = \frac{2\pi m}{a_2}$. The minimal quantities $b_1 = \frac{2\pi}{a_1}$ and $b_2 = \frac{2\pi}{a_2}$ are known as the primitive vectors of the reciprocal lattice, and the linear combination of integer multiples of them form the reciprocal lattice (shown in figure), which is the set of all arbitrary reciprocal lattice vectors defined by:

$$G = v_1 b_1 + v_2 b_2$$

The nice thing about reciprocal lattice vectors is that a given wave vector $\vec{k}$ will have an indistinguishable effect on the wave from a wave vector $\vec{k} + G$ that is translated by a reciprocal lattice vector. And remembering that $\hat{T}_d$ commutes with $\hat{\Theta}$, a general solution for $\mathbf{E}$ can be arranged by having a linear combination of the functions with different $\vec{k}+G$ that are solution to equation 2.42. This is done as follows:

$$\begin{aligned}
\mathbf{E}(\mathbf{r})_{\vec{k}} &= \sum_{m,n} C_{m,n} e^{i(\vec{k}+G_{m,n})\cdot r} \\
\mathbf{E}(\mathbf{x},\mathbf{y})_{k_x,k_y} &= \sum_{m,n} C_{m,n} e^{i(\vec{k}_x+mb_1)x} e^{i(\vec{k}_y+mb_2)y} \\
\mathbf{E}(\mathbf{x},\mathbf{y})_{k_x,k_y} &= e^{i\vec{k}_x x} e^{i\vec{k}_y y} \sum_{m,n} C_{m,n} e^{imb_1 x} e^{imb_2 y} \\
\mathbf{E}(\mathbf{r})_{\vec{k}} &= e^{i\vec{k}\cdot r} \mathbf{u}_{\vec{k}}(r)
\end{aligned}$$

Where $\mathbf{u}_{\vec{k}}(r)$ is by definition a bi-periodic function in r that satisfies $\mathbf{u}_{\vec{k}}(r+r') = \mathbf{u}_{\vec{k}}(r)$ for a given $\vec{k}$.This result is commonly known as Bloch's Theorem, in solid state physics, and in mechanics as Floquet's. $C_{m,n}$ are expansion coefficients to be solved by explicit solution and are the key for the formulation of the Plane Wave Expansion (PWE) method [20, 2, 19]. We however are not interested in $C_{m,n}$, because we are not solving for plane waves. Instead we will use this as a boundary condition by identifying how is the field after a translation of an arbitrary lattice vector R:

$$\mathbf{E}(\mathbf{r}+\mathbf{R})_{\vec{k}} = e^{i\vec{k}\cdot(r+R)}\mathbf{u}_{\vec{k}}(r+R) \tag{2.44}$$

$$\mathbf{E}(\mathbf{r}+\mathbf{R})_{\vec{k}} = e^{i\vec{k}\cdot R}e^{i\vec{k}\cdot r}\mathbf{u}_{\vec{k}}(r) \tag{2.45}$$

$$\mathbf{E}(\mathbf{r}+\mathbf{R})_{\vec{k}} = e^{i\vec{k}\cdot R}\mathbf{E}(\mathbf{r})_{\vec{k}} \tag{2.46}$$

We now know that if R is a lattice vector like r', the solution of the field at that point will be exactly the same than the solution of the filed in the point before the translation multiplied by a complex factor $e^{i\vec{k}\cdot R}$. So given a known wave vector $\vec{k}$ and the solution of every point inside the base, or unitary cell, one can know the value of $\mathbf{E}$ everywhere. This is because the boundaries of a unit cell are separated by a distance that is either a_1 in x or a_2 in y. Details about how to implement this condition in a numerical solution are given in [3].

2.4.4 Conservation of energy for waves in source less media

We define the energy of a field $\mathbf{w}$ as its norm $||\mathbf{w}||^2 = (\mathbf{w}, \mathbf{w})$. Conservation of energy in time in a source-less problem like that of equation 2.14 is obtained when $||\mathbf{w}||^2$ variation in time is zero, thus:

$$\frac{\partial||\mathbf{w}||^2}{\partial t} = \frac{\partial\,(\mathbf{w},\mathbf{w})}{\partial t} = (\dot{\mathbf{w}},\mathbf{w}) - (\mathbf{w},\dot{\mathbf{w}}) = \left(\hat{\Theta}\mathbf{w},\mathbf{w}\right) + \left(\mathbf{w},\hat{\Theta}\mathbf{w}\right) = \left(\hat{\Theta}\mathbf{w},\mathbf{w}\right) - \left(\hat{\Theta}\mathbf{w},\mathbf{w}\right) = 0 \tag{2.47}$$

Remembering that in a problem that is not harmonic we have $\hat{\Theta}\mathbf{E} = \frac{\partial^2\mathbf{E}}{\partial t^2}$This means that under Hermitic operators such as $\hat{\Theta}$ the energy of the field is conserved[5].

2.4.5 Energy functional of electromagnetic waves

By means of the **electromagnetic variational theorem** we can formulate an energy functional to to be minimized. This functional is defined as a normalized inner product between the function

[5]This expression for energy conservation holds only for cases where the operator does not change in time. A case where we have time varying material properties $\epsilon(r,t)$ can break conservation of energy if the time variation changes the norm[31]

and the function multiplied by the following operator known as Rayleigh Quotient:

$$U_f(\mathbf{E}) \triangleq \frac{\left(\mathbf{E}, \hat{\Theta}\mathbf{E}\right)}{(\mathbf{E}, \mathbf{E})} \tag{2.48}$$

It is similar to other formulations used in fields like Quantum Mechanics and Classical mechanics such as expectation values, and energy Lagrangians. The process of finding solutions that minimize such a potential is the core of what methods like Galerkin do for solving differential equations. In chapter 3 we will use a similar expression in order to solve the wave equation using Galerkin with Finite Elements.

Chapter 3

Finite element method

The Finite Element Method approximates an integral formulation that is –in some sense– equivalent with a differential equation. This integral (weak) formulation can be obtained from a variational principle, such as the Principle of Virtual Works, or the Integral of Action [32], but can also be obtained from a more general approach like the Weighted Residuals Method [33, 34]. In the case of electromagnetic waves, we can formulate a variational principle where the functional to be minimized is the energy carried by the electric and magnetic fields defined in 2.48.

This project uses the Galerkin Finite Element Method to construct an approximate solution of initial-boundary value and eigenvalue problems involving the wave equation of electromagnetic fields 2.29. Galerkin's method is one of these weighted residual methods in which both the weight functions and the bases for approximating the solution are defined as a linear combination of piece wise continuous polynomial functions h:

$$u = \sum_{i=1}^{N} u_i h_i$$

. Where u_i are the unknown expansion coefficients and h_i are the base functions.

So if we have a general partial differential equation of the form:

$$\mathcal{L}(u) = f \tag{3.1}$$

where $\mathcal{L}$ denotes a differential operator, u denotes the unknown solution to be found and f denotes source function, we can assume an approximation u' of u and build a residual defined as $\mathcal{R} = \mathcal{L}(u') - f$ to be minimized. What we want is to find a function u' that makes $\mathcal{R}$ a minimum in an *average* sense. A way to do this is to construct a functional that is defined as the integral over all space of the residual multiplied by a weighting function w, and then set this functional equal to

zero.

$$\int w\mathcal{R}(u') = 0 \tag{3.2}$$

Now depending on what bases we use to approximate u and what we choose to do with the weighting functions we will get a different weighted residual method to obtain u. Galerkin's method works by defining w in the same form of u as shown before.

In the rest of this chapter, we will show how to apply Galerkin's Finite Element formulation to electromagnetism problems, and specifically to the time harmonic wave equation described in 2.29. The concepts treated here are the basis of the code that has been implemented in the software platform PeyeQM.

3.1 Weak formulation of the problem

As mentioned before, in a weighted residual formulation weight functions are used in order to minimize a functional that is stated as the integral of the operator problem over a given simulation domain such as the one illustrated in 3.1.

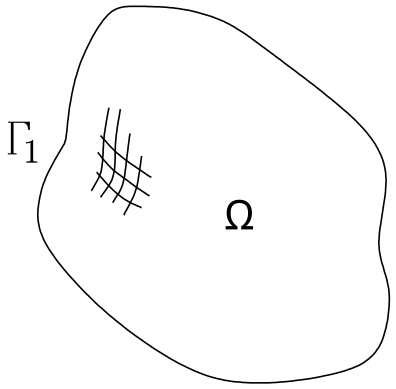

Figure 3.1: Abstract representation of a simulation domain Ω and its boundary Γ

If the field we want to obtain $\mathbf{E}$ is solution to equation 2.29, then it must satisfy the variational form that results by multiplying 2.29 by an arbitrary test function $\mathbf{W}$ and then integrating over Ω $\int_\Omega \mathbf{W}\cdot$(2.29):

$$\int_\Omega \mathbf{W}\cdot\left[\nabla\times\left(\bar{\bar{\mu_r}}^{-1}\nabla\times\mathbf{E}\right)-k_0^2\bar{\bar{\epsilon_r}}\cdot\mathbf{E}\right]=-ik_0Z_0\int_\Omega \mathbf{W}\cdot\mathbf{J} \tag{3.3}$$

Note that here the term $\nabla\times\left(\bar{\bar{\mu_r}}^{-1}\nabla\times\bullet\right)-k_0^2\bar{\bar{\epsilon_r}}\cdot\bullet$ stands as the operator $\mathcal{L}$ in equation 3.1, and $-ik_0Z_0\mathbf{J}$ as f. Separating the two terms of this new $\mathcal{L}$ we get:

$$\int_\Omega \mathbf{W}\cdot\nabla\times\left(\bar{\bar{\mu_r}}^{-1}\nabla\times\mathbf{E}\right)-k_0^2\int_\Omega \mathbf{W}\cdot\bar{\bar{\epsilon_r}}\cdot\mathbf{E}=-ik_0Z_0\int_\Omega \mathbf{W}\cdot\mathbf{J} \tag{3.4}$$

Now, lets focus on the first integral on the left hand side of the equation. Here we will invoke the vector identity: $\mathbf{A} \cdot \nabla \times \mathbf{B} = \mathbf{B} \cdot \nabla \times \mathbf{A} - \nabla \cdot (\mathbf{A} \times \mathbf{B})$ Where $\mathbf{A} = \mathbf{W}$ and $\mathbf{B} = \bar{\bar{\epsilon_r}} \nabla \times \mathbf{E}$ To express it as:

$$\int_\Omega \mathbf{W} \cdot \nabla \times \left(\bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E}\right) = \int_\Omega \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{W} - \int_\Omega \nabla \cdot \left(\mathbf{W} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E}\right) \tag{3.5}$$

The Divergence Theorem allows us to transform the volume integral on the second term of the right hand side to a closed surface integral:

$$\int_\Omega \nabla \cdot \left(\mathbf{W} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E}\right) = \oint_\Gamma \hat{n} \cdot \left(\mathbf{W} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E}\right) \tag{3.6}$$

And we will have two kinds of boundaries, Dirichlet and Neumann [35], thus $\oint_\Gamma = \int_{\Gamma_D} + \int_{\Gamma_N}$. Weight functions have been conveniently chosen to be zero at Dirichlet boundaries so, $\mathbf{W}\,|_{\Gamma_D} = 0$. One last step is to apply the triple product identity to the argument inside the Neumann integral so that:

$$\hat{n} \cdot \left(\mathbf{W} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E}\right) = \mathbf{W} \cdot \left(\bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E} \times \hat{n}\right) = -\mathbf{W} \cdot \left(\hat{n} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E}\right)$$

Substituting everything we get to the form:

$$\int_\Omega \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{W} - k_0^2 \mathbf{W} \cdot \bar{\bar{\epsilon_r}} \cdot \mathbf{E} + \int_{\Gamma_N} \mathbf{W} \cdot \left(\hat{n} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E}\right) = -ik_0 Z_0 \int_\Omega \mathbf{W} \cdot \mathbf{J} \tag{3.7}$$

3.2 Boundary conditions

For uniqueness of the solution and given that this is a boundary value problem, we must define how is the field or the derivative of the field at boundaries. The two conditions we come upon for defining how the system behaves at boundaries Γ_D and Γ_N are:

$$\hat{n} \times \mathbf{E} = \mathbf{P} \quad on\ \Gamma_D \tag{3.8}$$

$$\hat{n} \times \left(\bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E}\right) = \mathbf{K}_N \quad on\ \Gamma_N \tag{3.9}$$

$\mathbf{P}$ is the value for tangential electric field on Dirichlet boundaries Γ_D, and $\mathbf{K}_N$ is a function that represents boundary sources or electric field flux on Γ_N.

Replacing equation 3.9 in 3.7 we get:

$$\begin{aligned} \int_\Omega \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{W} - k_0^2 \mathbf{W} \cdot \bar{\bar{\epsilon_r}} \cdot \mathbf{E} \quad &= \quad -ik_0 Z_0 \int_\Omega \mathbf{W} \cdot \mathbf{J} \\ &\quad - \int_{\Gamma_N} \mathbf{W} \cdot \mathbf{K}_N \end{aligned} \tag{3.10}$$

by using:

$$-\mathbf{W} \cdot (\hat{n} \times (\hat{n} \times \mathbf{E})) = (\hat{n} \times \mathbf{W}) \cdot (\hat{n} \times \mathbf{E})$$

3.3 Abstract form of the equation

Now, in order to reduce verbosity and ease the calculations, we define a set of bi-linear operators[1] who represent the integrals in 3.10. These operators are meant to ease mathematical manipulations. Let's do some definitions first: $\mathbb{V}$ is a vector space of square integrable functions that vanishes at Dirichlet boundaries.

$$\mathbb{V} = \left\{ v \in L^2(a,b) : a(v,v) < \infty \wedge v(\Gamma_D) = 0 \right\}$$

$$L^2 = \left\{ v : \Omega \to \mathbb{R} \quad such\ that \quad \int_\Omega v^2 d\Omega = 0 \right\}$$

Operator a, represents the first term of the left hand side in equation 3.10:

$$\begin{aligned} a(\mathbf{W}, \mathbf{E}) \quad &: \mathbb{V} \times \mathbb{V} \to \mathbb{R} \\ a(\mathbf{W}, \mathbf{E}) \quad &= \int_\Omega \bar{\bar{\mu_r}}^{-1} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{W} d\Omega \end{aligned} \tag{3.11}$$

b is the second term in the left hand side in equation 3.10:

$$\begin{aligned} m(\mathbf{W}, \mathbf{E}) \quad &: \mathbb{V} \times \mathbb{V} \to \mathbb{R} \\ m(\mathbf{W}, \mathbf{E}) \quad &= \int_\Omega k_0^2 \mathbf{W} \cdot \bar{\bar{\epsilon_r}} \cdot \mathbf{E} d\Omega \end{aligned} \tag{3.12}$$

q represents boundary sources.

$$\begin{aligned} q(\mathbf{W}) \quad &: \mathbb{V} \to \mathbb{R} \\ q(\mathbf{W}) \quad &= \int_{\Gamma_N} \mathbf{W} \cdot \mathbf{K}_N \end{aligned} \tag{3.13}$$

and f body source terms. In the implementation $f = 0$.

$$\begin{aligned} f(\mathbf{W}) \quad &: \mathbb{V} \to \mathbb{R} \\ f(\mathbf{W}) \quad &= ik_0 Z_0 \int_\Omega \mathbf{W} \cdot \mathbf{J} \end{aligned} \tag{3.14}$$

Where $\mathbf{J}$ and $\mathbf{K}_N$ are known field and boundary conditions. These operators take vectors as inputs and return scalars. Using them one can construct a form of equation 3.10 known as the abstract form.

[1] Which are similar to the definition of inner product made in 2.34

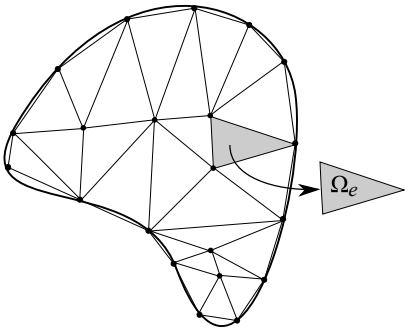

Figure 3.2: Discretized domain defined by a set of elements and their nodes. Taken with permission from [3]

$$a(\mathbf{W}, \mathbf{E}) - m(\mathbf{W}, \mathbf{E}) = -f(\mathbf{W}) - q(\mathbf{W}) \tag{3.15}$$

3.4 Base functions and discretization

However right now we don't have a proper Galerkin formulation because $\mathbf{E}$ is still a continuous function of all space, and thus, very hard to solve. Specifically if we are in a 2D domain where:

$$\mathbf{E} = E(x,y)_x \hat{a}_x + E(x,y)_y \hat{a}_y$$

To have a discrete problem to solve for $\mathbf{E}$, it must be approximated, and this is done by assuming that we can write it as a linear combination of base functions. Obviously, to have a computable problem we must take only a finite number of those base functions. What we do in FE is using piecewise polynomials as base functions, piecewise polynomials are functions that we define continuous only over disjoint finite regions of space. So a function like $\mathbf{E}$ that is continuous over all space can be split into a number N of functions that each is continuous over a finite region of the domain. Being so, the first step is to split the domain into elements like illustrated in figure 3.2. Each base function of the expansion will exist only within its corresponding finite region or element. With that we have

$$\mathbf{E} = \sum_{el=1}^{N_{el}} \mathbf{E}^{el} \tag{3.16}$$

Where $\mathbf{E}^{el}$ is a linear combination of interpolation functions h_i who are locally continuous inside elements el:

$$\mathbf{E}^{el} = \sum_{i=1}^{N} E_i^{el} h_i^{el} \tag{3.17}$$

and the values E_i^{el} are the expansion coefficients that allow the functions to reach $\mathbf{E}$. Now, these functions h_i can be seen as local interpolation functions of arbitrary order, and the order of the function will be proportional to the number of evaluation points in space needed to define it. The

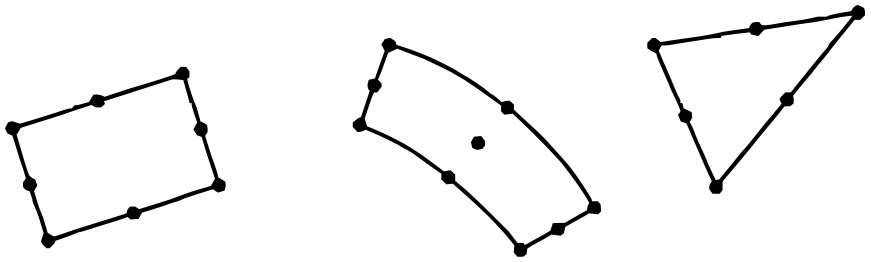

Figure 3.3: Three examples of commonly used two-dimensional elements. The one on the center is a 9 node quadrilateral element with a complete basis. To the left is a serendipity QUAD8 elements with 8 nodes, and to the right a 6 node triangular element.[4]

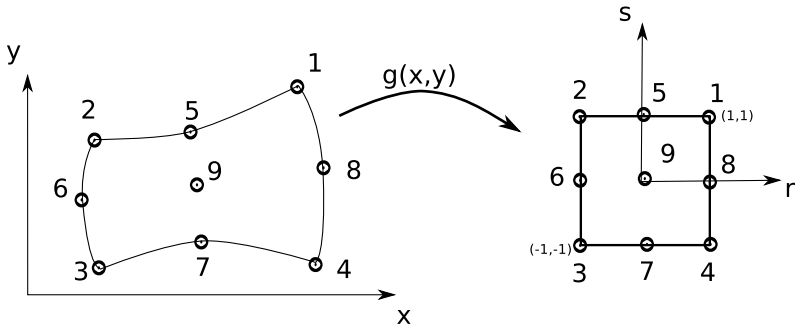

Figure 3.4: In order to ease the definition of base functions, every element is transformed to a generalized "isoparametric element" by means of a function $g(x, y)$ [4] so that all elements share the same base. The element on the left has an arbitrary shape and position, and by means of function $g(x, y)$ is mapped to a standard element in a domain in r, s that goes from -1 to 1

points will herein be called nodes, and they are a discrete representation of the coordinates in the domain. For higher order bases, one needs more evaluation points to define them, and because its easier to solve for lower order polynomials (fewer unknowns) we intentionally limit their order (usually to first or second order). Notice that an exact representation of $\mathbf{E}$ happens when $N \leftarrow \infty$.

For bi-dimensional elements like those in 3.3 one has to define one interpolation function for each node and they must be built so that on each node i the function h_i takes the value of 1 and zero at the rest. This is accomplished for a quadrilateral element of 4 to 9 nodes by using the functions given in table 3.1. The elements used in the software for the solution of electromagnetic fields are 8 node quadrilateral elements.

We will proceed by presenting some notation definitions for the future development: Lets say that the set of nodes that represent values of the field in the domain is called η. On nodes belonging to Dirichlet boundaries we will know the value of the field ($\mathbf{P}$) and it is convenient to split the set of nodes into its known and unknown elements $\eta_D \cup \eta \setminus \eta_D = \eta$. Each physical node will contain two

		Include if the node is present.				
		$i=5$	$i=6$	$i=7$	$i=8$	$i=9$
h_1	$\frac{1}{4}(1+r)(1+s)$	$-\frac{1}{2}h_5$			$-\frac{1}{2}h_8$	$-\frac{1}{2}h_9$
h_2	$\frac{1}{4}(1-r)(1+s)$	$-\frac{1}{2}h_5$	$-\frac{1}{2}h_6$			$-\frac{1}{2}h_9$
h_3	$\frac{1}{4}(1-r)(1-s)$		$-\frac{1}{2}h_6$	$-\frac{1}{2}h_7$		$-\frac{1}{2}h_9$
h_4	$\frac{1}{4}(1+r)(1-s)$			$-\frac{1}{2}h_7$	$-\frac{1}{2}h_8$	$-\frac{1}{2}h_9$
h_5	$\frac{1}{2}(1-r^2)(1+s)$					$-\frac{1}{2}h_9$
h_6	$\frac{1}{2}(1-s^2)(1-r)$					$-\frac{1}{2}h_9$
h_7	$\frac{1}{2}(1-r^2)(1-s)$					$-\frac{1}{2}h_9$
h_8	$\frac{1}{2}(1-s^2)(1+r)$					$-\frac{1}{2}h_9$
h_9	$\frac{1}{2}(1-r^2)(1-r^2)$					$-\frac{1}{2}h_9$

Table 3.1: Interpolation functions of four to nine variable-number-nodes two-dimensional isoparemtric elements.

components of the field[2], one for x and the other for y. So the number of evaluation nodes gets doubled.

If N is the total number of values:

$$N = n_x + n_y$$

.

Where n_x and n_y are the number of evaluation nodes associated to fields in x and y . When programming it is convenient to distinguish where does a node belongs, so we will treat n_x and n_y as sets as well in order to build a notation for the sums based on how an iterator surfs a set.

$$n_x = n_x \in \eta_D + n_x \in \eta \setminus \eta_D$$
$$n_y = n_y \in \eta_D + n_y \in \eta \setminus \eta_D$$

If we say:

$$i : n_x \in \eta_D$$

That will mean that the iteration will occur on the indexes that belong to the set of nodes in the Dirichlet region associated to x component of the field. The field fr each node, for each region and and for each component can then be approximated by:

$$E(x,y)_x \approx \sum_{i:\ n_x \in \eta_D} h_i E_i^x + \sum_{i:\ n_x \in \eta \setminus \eta_D} h_i E_i^x$$
$$E(x,y)_y \approx \sum_{i:\ n_y \in \eta_D} h_i E_i^y + \sum_{i:\ n_x \in \eta \setminus \eta_D} h_i E_i^y$$

[2]In a vectorial element formulation like the one used for EM fields in our program.

Left side is known values and right side is unknowns. In the same way for test functions W on the boundary we have:

$$\mathbf{W} = W(x,y)_x\hat{a}_x + W(x,y)_y\hat{a}_y$$

$$W(x,y)_x \approx \sum_{i:\ n_x\in\eta_D} h_i W_i^x + \sum_{i:\ n_x\in\eta\setminus\eta_D} h_i W_i^x$$
$$W(x,y)_y \approx \sum_{i:\ n_y\in\eta_D} h_i W_i^y + \sum_{i:\ n_x\in\eta\setminus\eta_D} h_i W_i^y$$

Moreover $\mathbf{J}$ and $\mathbf{K}_N$ can also be interpolated if there are Neumann boundary conditions or, sources.

$$\mathbf{J} = J(x,y)_x\hat{a}_x + J(x,y)_y\hat{a}_y$$

$$J(x,y)_x \approx \sum_{i:\ n_x\in\eta_D} h_i J_i^x + \sum_{i:\ n_x\in\eta\setminus\eta_D} h_i J_i^x$$
$$J(x,y)_y \approx \sum_{i:\ n_y\in\eta_D} J_i^y + \sum_{i:\ n_x\in\eta\setminus\eta_D} J_i^y$$

$$\mathbf{K}_N = K(x,y)_x\hat{a}_x + K(x,y)_y\hat{a}_y$$

$$K(x,y)_x \approx \sum_{i:\ n_x\in\eta_D} h_i K_i^x + \sum_{i:\ n_x\in\eta\setminus\eta_D} 0$$
$$K(x,y)_y \approx \sum_{i:\ n_y\in\eta_D} h_i K_i^y + \sum_{i:\ n_x\in\eta\setminus\eta_D} 0$$

Substitution of $\mathbf{E}$ and $\mathbf{W}$, into 3.10 is a mess. To make it easier lets use the operators and their nice properties. To be bi linear is to be linear in both arguments, to be linear means to satisfy additivity and homogeneity. An illustration of this:

$$\alpha u + \beta v, w) = \alpha a(u,w) + \beta a(v,w)$$
$$a(u, \alpha v + \beta w) = \alpha a(u,v) + \beta a(u,w)$$

The rotational and dot product inside the integrals defined in the abstract forms are linear. A proof for that is out of the reach of this document. The proof of

$$a(u,v) = a(v,u)$$

was made in equation 2.39, and a proof of

$$m(u,v) = m(v,u)$$

is obtained by assuming that v and u have the same base. Which we did by using the Galerkin weighting.

So the following can happen:

$$\begin{aligned}
a\left(W_x, E_x\right) = \quad & a\left(\sum_{i:\ n_x\in\eta_D} h_i W_i^x, \sum_{j:\ n_x\in\eta_D} h_j E_j^x\right) + a\left(\sum_{i:\ n_x\in\eta\setminus\eta_D} h_i W_i^x, \sum_{j:\ n_x\in\eta\setminus\eta_D} h_j E_j^x\right) \\
= \quad & \sum_{i:\ n_x\in\eta_D} W_i^x a\left(h_i, \sum_{j:\ n_x\in\eta_D} h_j E_j^x\right) + \sum_{i:\ n_x\in\eta\setminus\eta_D} W_i^x a\left(h_i, \sum_{j:\ n_x\in\eta\setminus\eta_D} h_j E_j^x\right) \\
= \quad & \sum_{i:\ n_x\in\eta_D} W_i^x \sum_{j:\ n_x\in\eta_D} E_j^x a\left(h_i, h_j\right) + \sum_{i:\ n_x\in\eta\setminus\eta_D} W_i^x \sum_{j:\ n_x\in\eta\setminus\eta_D} E_j^x a\left(h_i, h_j\right) \\
= \quad & \sum_{j:\ n_x\in\eta_D} a\left(h_i, h_j\right) E_j^x \sum_{i:\ n_x\in\eta_D} W_i^x + \sum_{j:\ n_x\in\eta\setminus\eta_D} a\left(h_i, h_j\right) E_j^x \sum_{i:\ n_x\in\eta\setminus\eta_D} W_i^x
\end{aligned} \tag{3.18}$$

$$\sum_{j:\ n_x\in\eta_D} E_j^x = \vec{g}_j^x$$

$$\sum_{i:\ n_x\in\eta\setminus\eta_D} W_i^x = \vec{W}^x$$

$$\sum_{j:\ n_x\in\eta\setminus\eta_D} E_j^x = \vec{E}^x$$

$$\vec{g}_i = \vec{g}_j^x + \vec{g}_j^y \qquad j = 1..N$$

$\vec{g}_j$ is a vector of size N that holds values of E on each Dirichlet boundary node for both x, and y components. Vector $\hat{g}_i^y$ follows from a similar procedure on E_y. It is of importance to say that even though the sets defined under the summation are subsets of the set of all nodes, when programming, their associate vectors will span the whole domain. In other words this notation stands for entries to indexes to vectors that may be defined full of zeroes. Or in a different approach: j in $j:\ n_x \in \eta_D$ can be indexes 1 and N meaning that the first and last nodes belong to Dirichlet boundary points on x. All other elements of the vector are undefined and hold their initiation value zero.

Changing the indexes i, j and using the definition of dot product in a matrix notation we can translate the abstract form as:

$$a\left(W_x, E_x\right) = \langle \mathbb{A}\vec{g}^x, \vec{W}^x \rangle + \langle \mathbb{A}\vec{E}^x, \vec{W}^x \rangle$$

Where $\mathbb{A}$ is a matrix whose elements contain the indexed integration over the basis functions h_i, and h_j, these are known values because we know the form of the functions and can calculate the integrals by numerical or analytic means. For the solution in the software routines we implemented numerical integration based on Gauss-Legendre quadrature as defined in [4].

Matrix $\mathbb{A}$ is known as the stiffness matrix in Mechanics, and here it relates to energy stored in form of electric field. Matrix $\mathbb{A}$ is built as a combination of matrices $\mathbb{A}^x$ and $\mathbb{A}^y$:

$$\mathbb{A} = \mathbb{A}^x + \mathbb{A}^y$$

where

$$\mathbb{A}^x = \sum_{i,j}^{n_x \in \eta \setminus \eta_D} a(h_i, h_j) + \sum_{i,j}^{n_y \in \eta \setminus \eta_D} a(h_i, h_j)$$

The product $\mathbb{A}^x \vec{g}^x = \vec{d}^x$ will be called the Dirichlet vector.

$$a\left(W_x, E_x\right) = \langle \vec{d}^x, \vec{W}^x \rangle + \langle \mathbb{A}^x \vec{E}^x, \vec{W}^x \rangle \tag{3.19}$$

In a very similar way but now substituting the approximate functions into the kinetic operator $m(\mathbf{W}, \mathbf{E})$ we get:

$$\begin{aligned} m\left(W_x, E_x\right) &= \langle \mathbb{M}^x \vec{g}^x, \vec{W}^x \rangle + \langle \mathbb{M}^x \vec{E}^x, \vec{W}^x \rangle \\ m\left(W_x, E_x\right) &= \langle \vec{b}^x, \vec{W}^x \rangle + \langle \mathbb{M}^x \vec{E}^x, \vec{W}^x \rangle \end{aligned} \tag{3.20}$$

With $\mathbb{M}$ as the mass matrix or inductance matrix. Being related to the time derivative of the electric field, we can associate this to energy stored in the form of magnetic fields like a sort of inductance matrix. The global mass matrix is also a combination of terms related to the degree of freedom x and y:

$$\begin{aligned} \mathbb{M} &= \mathbb{M}^x + \mathbb{M}^y \\ \mathbb{M} &= \sum_{i,j}^{n_x \in \eta \setminus \eta_D} m(h_i, h_j) + \sum_{i,j}^{n_y \in \eta \setminus \eta_D} m(h_i, h_j) \end{aligned}$$

And the same follows with operators q and f:

$$f(W_x) = \langle \vec{f}^x, \vec{W}^x \rangle \tag{3.21}$$

$$q(W_x) = \langle \vec{q}^x, \vec{W}^x \rangle \tag{3.22}$$

$$\vec{f}_i^x = ik_0 Z_0 \sum_{i:\ n_x \in \eta} \int_\Omega h_i J_i^x$$

$$\vec{q}_i^x = \sum_{i:\ n_x \in \eta_N} \int_{\Gamma_N} h_i K_i^x$$

Where we introduced a subset of the set $\eta \setminus \eta_D$ called η_N which represents nodes on Neumann boundaries.

Substituting definitions: 3.19,3.20, 3.22, 3.21 in equation 3.15, we get:

$$\langle \vec{d}^x, \vec{W}^x \rangle + \langle \mathbb{A}^x \vec{E}^x, \vec{W}^x \rangle - \langle \vec{b}^x, \vec{W}^x \rangle - \langle \mathbb{M}^x \vec{E}^x, \vec{W}^x \rangle = -\langle \vec{f}^x, \vec{W}^x \rangle - \langle \vec{q}^x, \vec{W}^x \rangle$$

$$\langle \mathbb{A}^x \vec{E}^x - \mathbb{M}^x \vec{E}^x, \vec{W}^x \rangle = \langle \vec{b}^x - \vec{d}^x - \vec{q}^x - \vec{f}^x, \vec{W}^x \rangle \tag{3.23}$$

Being $\vec{W}^x$ an arbitrary function, the following linear system of equations appear:

$$\left(\mathbb{A}^x - \mathbb{M}^x\right) \vec{E}^x = \vec{b}^x - \vec{d}^x - \vec{q}^x - \vec{f}^x \tag{3.24}$$

Similarly, and following exactly the same procedure:

$$\left(\mathbb{A}^y - \mathbb{M}^y\right) \vec{E}^y = \vec{b}^y - \vec{d}^y - \vec{q}^y - \vec{f}^y \tag{3.25}$$

These two systems can be solved simultaneously by intercalating rows and columns of $\mathbb{A}^x$ with $\mathbb{A}^y$, and keeping the formulation as one global matrix multiplying a vector that contains values of the field in both components

$$\left(\mathbb{A} - \mathbb{M}\right) \vec{E} = \vec{b} - \vec{d} - \vec{q} - \vec{f}$$

Something that is used in many references is to remove rows and columns associated to Dirichlet positions of E in the matrices and vectors of the equation. This is done because we already know the values of E for those points and their information is already saved in vector $\vec{d}$. We will add the symbol $\setminus D$ before the symbols to note that Dirichlet rows and columns should be deleted.

$$\setminus^D \left[\left(\mathbb{A} - \mathbb{M}\right) \vec{E} = \vec{b} - \vec{d} - \vec{q} - \vec{f}\right]$$

The solution of this linear system of equations is achieved by means of robust computational routines such as LAPACK. And they are finally put together with the known values in order to get a full representation of the field inside the domain.

3.5 Edge elements vs Node Elements

There is another way to represent vector fields in Finite Elements, and it is by using Edge Elements [1]. Up until now we defined the problem as a simultaneous solution of two scalar fields, one associated to the x coordinate and another to y, the advantage of doing that over solving for only one scalar field is that we have much more flexibility in the kinds of boundary conditions that can be formulated. We also achieve higher accuracy and can know more about the problem.

There is however another form to postulate vector fields, and that is by defining tangential values of the field on each edge and interpolating $\mathbf{E}$ inside the element by using vectorial basis functions, these functions are then not related to each node, but to each line at the element. In the jargon of Finite Element these are termed Nedelec elements [36, 37], some preferences on these elements over the nodal elements resides on the fact that the electric and magnetic fields are 1-manifolds. For example, the field inside a three node element (el) with nodes $1, 2, 3$ is then the combination of the vector shape functions $\mathbf{N}_{ij}$:

$$\mathbf{E}^{el}(x, y) = \mathbf{N}^{el}_{12} E^{el}_{12} + \mathbf{N}^{el}_{13} E^{el}_{13} + \mathbf{N}^{el}_{23} E^{el}_{23} \tag{3.26}$$

And from this point the formulation continues almost identically to what we did before.

There is however an ongoing discussion [38, 39, 40] over the advantages and disadvantages of using edge elements instead of nodal elements, and at the moment of implementation edge elements seemed more troublesome to implement than node elements. The formulation of a vector field solver using edge elements is pending as one possible alternative to our current implementation.

3.6 Time domain formulation

For analysis of wave propagation in finite lattices with defects we must be able to simulate the evolution of the fields along time and a stationary harmonic model is no longer suitable. For this we have to define construct a numerical solution that combines the finite element method and finite differences, the former for the solution of spatial derivatives and the later for time derivatives. The following procedures are summarized and have less comments because they are made in a very similar way than the process in the previous section.

3.6.1 Weak formulation in time dependent problem

Proceeding in the same way as in the time harmonic case we take equation 2.14, and rewrite it as:

$$\nabla \times \left[\bar{\bar{\mu}}^{-}1 \cdot \nabla \times \mathcal{E}(t)\right] + \bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathcal{E}(t)}{\partial t^2} = -\frac{\partial \mathcal{J}}{\partial t} \tag{3.27}$$

Multiplying by the test function $\mathbf{W}$ and integrating over the entire domain will give us a functional equivalent to 3.7 to be minimized:

$$\begin{aligned}
\int_\Omega \mathbf{W} \cdot \nabla \times \left[\bar{\bar{\mu}}^{-}1 \cdot \nabla \times \mathcal{E}(t)\right] + \mathbf{W} \cdot \bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathcal{E}(t)}{\partial t^2} &= -\int_\Omega \mathbf{W} \cdot \frac{\partial \mathcal{J}}{\partial t} \\
\int_\Omega \mathbf{W} \cdot \nabla \times \left[\bar{\bar{\mu}}^{-}1 \cdot \nabla \times \mathcal{E}(t)\right] + \int_\Omega \mathbf{W} \cdot \bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathcal{E}(t)}{\partial t^2} &= -\int_\Omega \mathbf{W} \cdot \frac{\partial \mathcal{J}}{\partial t} \\
\int_\Omega \bar{\bar{\mu_r}}^{-1} \nabla \times \mathcal{E}(t) \cdot \nabla \times \mathbf{W} - \int_\Omega \nabla \cdot \left(\mathbf{W} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathcal{E}(t)\right) + \int_\Omega \mathbf{W} \cdot \bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathcal{E}(t)}{\partial t^2} &= -\int_\Omega \mathbf{W} \cdot \frac{\partial \mathcal{J}}{\partial t} \\
\int_\Omega \bar{\bar{\mu_r}}^{-1} \nabla \times \mathcal{E}(t) \cdot \nabla \times \mathbf{W} - \oint_\Gamma \hat{n} \cdot \left(\mathbf{W} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathcal{E}(t)\right) + \int_\Omega \mathbf{W} \cdot \bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathcal{E}(t)}{\partial t^2} &= -\int_\Omega \mathbf{W} \cdot \frac{\partial \mathcal{J}}{\partial t} \\
\int_\Omega \bar{\bar{\mu_r}}^{-1} \nabla \times \mathcal{E}(t) \cdot \nabla \times \mathbf{W} + \int_{\Gamma_N} \mathbf{W} \cdot \left(\hat{n} \times \bar{\bar{\mu_r}}^{-1} \nabla \times \mathcal{E}(t)\right) + \int_\Omega \mathbf{W} \cdot \bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathcal{E}(t)}{\partial t^2} &= -\int_\Omega \mathbf{W} \cdot \frac{\partial \mathcal{J}}{\partial t}
\end{aligned} \tag{3.28}$$

This is achieved remembering that $\hat{n} \cdot \mathbf{W} = 0$ on Γ_D, and using all the vector identities invoked in the previous section.

3.6.2 Boundary conditions in time dependent problem

Boundaries are defined in the same way, but now they are time dependent:

$$\hat{n} \times \mathcal{E}(t) = \mathcal{P}(t) \quad on \quad \Gamma_D \tag{3.29}$$

$$\hat{n} \times \left(\bar{\bar{\mu}}^{-1} \nabla \times \mathcal{E}(t)\right) + Y \hat{n} \times \left(\hat{n} \times \frac{\partial \mathcal{E}(t)}{\partial t}\right) = \mathcal{K}_N(t) \tag{3.30}$$

Using Neumann boundary conditions applied to 3.28 we get:

$$\int_\Omega \bar{\bar{\mu_r}}^{-1} \nabla \times \mathcal{E}(t) \cdot \nabla \times \mathbf{W} + \mathbf{W} \cdot \bar{\bar{\epsilon}} \cdot \frac{\partial^2 \mathcal{E}(t)}{\partial t^2} = -\int_{\Gamma_N} \mathbf{W} \cdot \mathcal{K}_N(t) - \int_\Omega \mathbf{W} \cdot \frac{\partial \mathcal{J}}{\partial t} \tag{3.31}$$

3.6.3 Base functions and abstract form

To obtain a finite element solution the spatial domain has to be discretized in the same way as we did when obtaining the frequency domain solution. For this, we use compact support basis functions that in combination with nodal values will give a representation of the global field as the combination of spatial subdivisions called elements. In this case is better to invoke the definitions for the approximated functions first-hand:

$$\begin{aligned}\mathcal{E}(t) &= \mathcal{E}(x,y,t)_x\hat{a}_x + \mathcal{E}(x,y,t)_y\hat{a}_y \\ \mathcal{E}(x,y,t)_x &\approx \sum_{i:\ n_x\in\eta_D} h_i\mathcal{E}_i^x(t) + \sum_{i:\ n_x\in\eta\setminus\eta_D} h_i\mathcal{E}_i^x(t) \\ \mathcal{E}(x,y,t)_y &\approx \sum_{i:\ n_y\in\eta_D} h_i\mathcal{E}_i^y(t) + \sum_{i:\ n_x\in\eta\setminus\eta_D} h_i\mathcal{E}_i^y(t)\end{aligned}$$

Noting that when time derivatives are applied over the field, these will act only on the nodal values $\mathcal{E}_i^x(t)$ so that for example:

$$\frac{\partial\mathcal{E}(x,y,t)_x}{\partial t} \approx \sum_{i:\ n_x\in\eta_D} h_i\frac{\partial\mathcal{E}_i^x(t)}{\partial t} + \sum_{i:\ n_x\in\eta\setminus\eta_D} h_i\frac{\partial\mathcal{E}_i^x(t)}{\partial t}$$

W Keeps being the same as defined in the previous section.

$$\mathcal{J} = \mathbf{J}$$

$$\mathcal{K}_N = \mathbf{K}_N$$

Operators are almost the same as before. The main difference being that free space related material constants k_0 and Z_0 haven't been extracted from material matrices $\bar{\bar{\mu}}$ and $\bar{\bar{\epsilon}}$.

$$\begin{aligned}a(\mathbf{W},\mathbf{E}) &: \mathbb{V}\times\mathbb{V}\to\mathbb{R} \\ a(\mathbf{W},\mathbf{E}) &= \int_\Omega \bar{\bar{\mu}}^{-1}\nabla\times\mathbf{E}\cdot\nabla\times\mathbf{W}d\Omega\end{aligned} \tag{3.32}$$

$$\begin{aligned}m(\mathbf{W},\mathbf{E}) &: \mathbb{V}\times\mathbb{V}\to\mathbb{R} \\ m(\mathbf{W},\mathbf{E}) &= \int_\Omega \mathbf{W}\cdot\bar{\bar{\epsilon}}\cdot\mathbf{E}d\Omega\end{aligned} \tag{3.33}$$

$$\begin{aligned}q(\mathbf{W}) &: \mathbb{V}\to\mathbb{R} \\ q(\mathbf{W}) &= \textstyle\int_{\Gamma_N}\mathbf{W}\cdot\mathcal{K}_N\end{aligned} \tag{3.34}$$

$$\begin{aligned}f(\mathbf{W}) &: \mathbb{V}\to\mathbb{R} \\ f(\mathbf{W}) &= \textstyle\int_\Omega \mathbf{W}\cdot\frac{\partial\mathcal{J}}{\partial t}\end{aligned} \tag{3.35}$$

Note that all these are operators that act on spatial coordinates and as such are independent of

time derivatives. The abstract form for wave propagation in time domain formulation is then:

$$a\left(\mathbf{W},\mathcal{E}\right)+m\left(\mathbf{W},\frac{\partial^2\mathcal{E}}{\partial t^2}\right)=-f\left(\mathbf{W}\right)-q\left(\mathbf{W}\right) \tag{3.36}$$

If a process of substitution like that shown in equation 3.18 is performed for each of these operators, one gets a system of equations equivalent to the one in equation 3.4

$$\sum_{j:\ n_x\in\eta_D}\mathcal{E}_j^x=\{g^x\}$$

$$\mathbb{A}^x\left\{g^x\right\}=\left\{d^x\right\}$$

$$\mathbb{M}^x\left\{g^x\right\}=\left\{b^x\right\}$$

$$\sum_{i:\ n_x\in\eta\setminus\eta_D}W_i^x=\vec{W}^x$$

$$\sum_{j:\ n_x\in\eta\setminus\eta_D}\mathcal{E}_j^x=\left\{\mathcal{E}^x\right\}$$

$$\sum_{j:\ n_x\in\eta\setminus\eta_D}\frac{\partial^2\mathcal{E}_j^x}{\partial t^2}=\frac{\partial^2\left\{\mathcal{E}^x\right\}}{\partial t^2}$$

$$\sum_{i:\ n_x\in\eta}\int_\Omega h_i\frac{\partial\mathcal{J}_i^x}{\partial t}=\left\{f^x\right\}$$

$$\sum_{i:\ n_x\in\eta_N}\int_{\Gamma_N}h_i\mathcal{K}_i^x=\left\{q^x\right\}$$

$$\begin{aligned}&\langle\left\{d^x\right\},\vec{W}^x\rangle+\langle\mathbb{A}^x\left\{\mathcal{E}^x\right\},\vec{W}^x\rangle-\langle\left\{b^x\right\},\vec{W}^x\rangle-\langle\mathbb{M}^x\frac{\partial^2\left\{\mathcal{E}^x\right\}}{\partial t^2},\vec{W}^x\rangle=-\langle\left\{f^x\right\},\vec{W}^x\rangle-\langle\left\{q^x\right\},\vec{W}^x\rangle\\&\langle\mathbb{A}^x\left\{\mathcal{E}^x\right\}-\mathbb{M}^x\frac{\partial^2\left\{\mathcal{E}^x\right\}}{\partial t^2},\vec{W}^x\rangle=\langle\left\{b^x\right\}-\left\{d^x\right\}-\left\{q^x\right\}-\left\{f^x\right\},\vec{W}^x\rangle\end{aligned} \tag{3.37}$$

$$\mathbb{A}\left\{\mathcal{E}\right\}-\mathbb{M}\frac{\partial^2\left\{\mathcal{E}\right\}}{\partial t^2}=\left\{b\right\}-\left\{d\right\}-\left\{q\right\}-\left\{f\right\} \tag{3.38}$$

In the simulations that were performed as part of the project we didn't consider any sources or Neumann boundaries, so $\{q=0\}$ and $\{f\}$. The details about the finite difference formulation used to discretize the time derivative are explained in the section (3.7) where the explicit solution of 3.38 is explored .

3.7 Mass matrices

In numerical methods, and more specifically in Finite elements, mass matrices are a discrete representation of a continuous distribution of something, that multiplies the second time derivative of the solution function.

Mass matrices are cataloged as consistent or diagonal. Consistent mass matrices use the same shape functions as the ones used to generate the element stiffness matrix. All reference to mass matrices mentioned before, were consistent [41]. The other way to formulate mass matrices is the lumped mass matrix. Which is made by avoiding the interpolation and simply placing particle masses m_i at nodes i of an element such that $\sum m_i$ is the total element mass.

The great advantage of lumped mass matrices is that they are diagonal and the solution of the system of equations can be performed in a explicit manner convenient for big computational domains. However, precision is lost by not stating mass as functions to be interpolated.

To form a lumped mass matrix one can use the HRZ Lumping scheme which consist on the following algorithm [41]:

1. Compute the diagonal coefficients of the consistent mass matrix.
2. Compute the mass for each element m.
3. Compute the trace of the matrix s.
4. scale all the diagonal coefficients by multiplying them by the ratio $\frac{m}{s}$.

Lumped mass matrices are used in the implementation of an explicit formulation of Finite Elements in Time Domain FETD where equation 2.19 is solved instead of 3.4. The next section expands on the topic of explicit formulation.

3.8 Explicit formulation

The central-difference method is used to approximate the second order derivative of $\mathbf{E}$ in 3.38 by expanding $\mathbf{E}_{n+1}$ and $\mathbf{E}_{n-1}$ in taylor series about time $n \triangle t$:

$$\mathbf{E}_{n+1} = \mathbf{E}_n + \delta t\dot{\mathbf{E}}_n + \frac{\delta t^2}{2}\ddot{\mathbf{E}}_n + \frac{\delta t^3}{6}\dddot{\mathbf{E}}_n + O(\delta t^4) \tag{3.39}$$

$$\mathbf{E}_{n-1} = \mathbf{E}_n - \delta t\dot{\mathbf{E}}_n + \frac{\delta t^2}{2}\ddot{\mathbf{E}}_n - \frac{\delta t^3}{6}\dddot{\mathbf{E}}_n + O(\delta t^4) \tag{3.40}$$

Restricting to fourth-order accuracy ($O(\delta t^4)$) and adding 3.39 to 3.40 one can easily solve for $\mathbf{E}_{n+1}$:

$$\ddot{\mathbf{E}}_n = \frac{1}{\delta t^2}\left(\mathbf{E}_{n+1} - 2\mathbf{E}_n + \mathbf{E}_{n-1}\right) \tag{3.41}$$

subscript n denotes time $n\delta t$ with δt being the selected timestep.

Replacing into our general formulation one gets (fix details):

$$\frac{1}{\delta t^2}\mathbb{M}\mathbf{E}_{n+1} = -\mathbb{A}\mathbf{E}_n + \frac{1}{\delta t^2}\mathbb{M}\left[2\mathbf{E}_n - \mathbf{E}_{n-1}\right] \tag{3.42}$$

The equation 3.42 can be solved by an implicit solver that takes matrices and solves in one pass the system of equations for one time n as we do with 3.4. This however is not very practical for interesting problems with complex domains when running under workstations that have low memory such as my laptop. For the solution of 3.42 is best to solve for each node at a time and this is done by using lumped matrices instead of consistent ones. Having a diagonal mass matrix $\mathbb{M}$ decouples the nodal values in time and makes it possible to solve for $\mathbf{E}_{n+1}$ without solving simultaneous systems of equations. The details of how the algorithm computes the solution of 3.42 explicitly are treated in the documentation of the software online.

Chapter 4

Implementation

This chapter explores some key concepts about the implementation of the simulation platform that was built. All the code and routines that were produced, and that as a whole form the platform have been named PeyeQM. In the first section an overview of the Object Oriented Paradigm will be presented to contextualize some of the notions used in further sections. Then we will present a simplified structure of what a Finite Element solver should have, and from that, the implementation of PeyeQM will be treated in a rather simplified form. Finally, a brief explanation of what is Python (the language we used to program) and other two tools is given. The reader is encouraged to see the documentation for routines available online in the following repository.

4.1 Object Oriented Paradigm

During developments of FEM software, there have been changes in programming paradigms, from the traditional procedure-oriented to the object-oriented. Being the traditional procedure usually tied to Fortran and C programming languages, and the object-oriented to higher level languages such as C++, Java, and Python.

Generally speaking, under procedure oriented paradigms when software platforms grow, necessary changes in the code (say, addition of new kinds of elements) require changes in the whole program. Inter dependencies in the program architecture are often hidden and difficult to determine. Most of the time, in procedure oriented paradigm a high degree of knowledge of the entire program is necessary to modify small routines [42]. Extensibility and flexibility are thus two important qualities that the procedure oriented paradigm lacks. Flexibility and abstraction are the keys to manage complexity [43], and they are two of the requirements that a FE platform must meet. The following is a list of general requirements that PeyeQM was meant to meet:

Modularity Construction of a software by almost independent or loosely coupled components

[43]

Reusability The likelihood that a segment of source code can be used again to add new functionalities with slight or no modification.

Extensibility A design principle where the implementation takes into consideration future growth.

Multi platform capabilities Computing methods and concepts that are implemented and interoperate on multiple computer platforms [44].

Readability A human judgment of how easy a text is to understand. "The readability of a program is related to its maintainability" [45].

Now, one programming paradigm that satisfies these requirements is the Object-Oriented Paradigm (OOP). A program implemented under OOP may be viewed as a collection of interacting objects, as opposed to the conventional model, in which a program is seen as a list of tasks (subroutines) to perform [46]. The objects that form a program under OOP are representations of things with inherent qualities and attributes. They are capable of receiving messages, processing (or storing) data, and sending messages to other objects. **Modularity** is then obtained by abstracting the process of a FE computation as the interaction between a set of objects, each of which represents a stage or process. **Reusability** and **extensibility** are a consequence of the objects being independent one of the other, defining their relations as parsing of messages that are sufficiently general, allows programers that work on different sections of the code to treat objects they are not working on as **black boxes**. Now, as long as the message doesn't change, work on one object does not affect other parts of the program, so new functionalities can be easily added. In order to achieve a good implementation of a OOP program one needs to start from the beginning with a good idea of the general structure of objects and their interrelations, so that the strengths of OOP qualities get well-exploited. The main characteristics of Object Oriented programming are:

Data abstraction In OOP data abstraction is obtained by definition of classes. A class is the set of common attributes that a general object of a certain kind or *class* has. For example, "Banana" class would represent the properties and functionality of bananas in general, such as being edible, or having potassium. Usage of data abstraction allows us to generalize the behavior of objects independently of what is their current state (name, value of its attributes).

Encapsulation Is understood as the possibility of a language or paradigm of bundling data with the methods (or functions) that act on it, or have to do with it. So, in OOP objects not only contain information, but are capable of performing tasks because they have the functions for that built in their definition. In our example a Banana of Class Banana not only would have potassium but would also have the possibility of making someone fall if step upon.

Inheritance In OOP is the possibility of the paradigm to define classes of classes. Something like nesting of abstractions. For example a Banana is a class derived from a more general abstraction known as "Fruit", and has things (grows from a plant) in common with other Fruit classes such as "Orange" and "Apple".

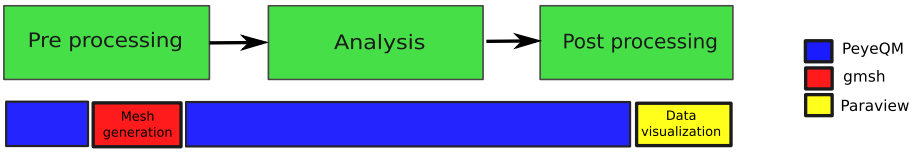

Figure 4.1: Representation of how much of the simulation process is performed by using PeyeQM's routines.

Polymorphism is a characteristic of operators that can act over objects without caring about its precise type. OOP allows the overloading of operators such that depending on the definitions of the class, certain operators perform different actions. For example in Python the + sign is polymorphic in the sense that it can add integers, if acting on integers, and concatenate strings if acting on strings.

All of this characteristics make the Object Oriented Paradigm a suitable framework from which to design FE software platforms such as PeyeQM according to the requirements listed above.

4.2 General stages in a simulation

There are three main stages in a finite element algorithm:

1. Pre-Processing
2. Analysis
3. Post-Processing

With the aim of focusing on problem solving and reduce coding of a very complicated platform, we have designed PeyeQM to perform only the critical parts of these stages and have resorted to external software to do Meshing and visualization as illustrated in figure 4.1. The following is a brief description of what is done in each stage:

4.2.1 Pre-Processing

Pre-Processing is the stage where we describe the problem that will be solved, in terms that are readily interpreted by the analysis (processing) stage. In other words, is what we need to do before solving. Comparing simulating to cooking this would be the moment when we choose and chop the ingredients as well as heating the stove. Preparing the ingredients in FEM means to:

- Define the geometry or domain of the simulation, with its regions and associated materials.

- Define what kind of physics rule the problem, and what kind of solution we want.
- Discretize or divide the domain in nodes and elements by using a meshing algorithm.
- Set boundary and initial conditions.
- Based on the physics of the problem, build a system of equations that relate all of the above.

4.2.2 Analysis

In the Processing or Analysis stage, we take the system of equations that has been constructed in the first stage and solve it using the respective numerical methods. If we have a stationary problem then we have to solve a system of linear equations, if it is an eigenvalue problem, then we have find the eigenvalues and vectors of a certain matrix, and if it is an explicit solution then we must loop over unknowns assigning values. All in all, what we will do in this stage is the equivalent to the actual cooking of the ingredients.

4.2.3 Post-Processing

Post processing, is to get the solution produced by the solver and serve it in a portioned and digestible way. In this step we will generally produce graphs, plots, and animations that represent the characteristic of the simulated phenomenon. If Pre-Processing was a sort of codification from human language to algorithm, then Post-Processing is a way of de-codification, from numbers to "human".

4.3 Classes, Diagrams and flow charts of PeyeQM

PeyeQM follows this basic Scheme but divides the Pre-Processing stage in two, one part is the assembling of the simulation and the second is the codification or interpretation of that simulation. Figure 4.2 shows the interpretation that we have made of the steps to follow.

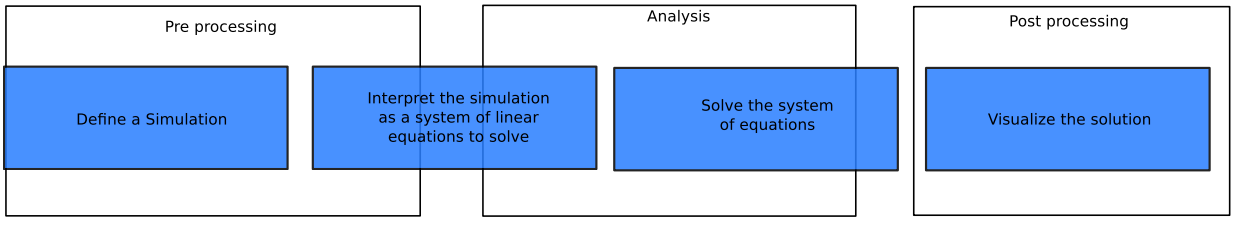

Figure 4.2: Diagram of general simulation stages as abstracted by PeyeQM.

The way in which we formulated the program in the context of OOP was to use three main classes that deal with the flow of chart 4.2.

- A *Simulation* class,
- *Interpreter* class, and
- *Solver* class.

In that way, a simulation can interpreted as a kind of conversation between the user and members (instances or objects) of those classes. The first part of that conversation would be the user telling what he wants to a member of class *Simulation*. Doing so the object "simulation" gets informed of the parameters needed to solve the problem. The solver however does not understand either human language or the simulation language, and it needs some intermediary to hear what the simulation has to say and translate it as systems of equations which are its language. This new character is called the Interpreter and is a member of class *Interpreter*. The interpreter is capable of reading the simulation and depending on the kind of physical problem, conditions and domain, will build the matrices and vectors mentioned in chapter 3, and assemble them in the form of an equation. At this point of the conversation, the Solver, who is a member of class *Solver* will listen to the interpreter and proceed to diligently solve the problem. Whenever the solution is ready it will then return the solution as a file that is in a predefined format, this file will then be passed to the Post-Processor, who will paint the solution for the user closing the loop of this conversation[1].

A diagram showing the main classes that form a Simulation in PeyeQM can be seen in figure 4.3.

4.4 PeyeQM usage

The best way to explain how to use the platform is with an example. As it is right now, the platform works by using Python scripts[2] that can be run by a computer that has Python console installed. In this section we will explain each line of a script file (called Waveguide.py) used to simulate the field inside a wave guide[3], hoping that with that the reader gets an idea of how to use the libraries and encouraging him-her to explore the examples that have been uploaded in the repository.

Starting from the first two lines we define what to use in order to execute the file and the kind of coding:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

The next part is the doc string of the script, this is what appears if from inside python you type `help(Waveguide.py)`. The documentation will include the file types that are needed and will also give a brief summary of what the file does.

[1] It is important to remember now that we are using an external tool called Paraview for the visualization of data.
[2] Programs that can interpret and automate the execution of tasks which could alternatively be executed one-by-one by a human operator.
[3] A proper explanation of what is a waveguide is made in chapter 5

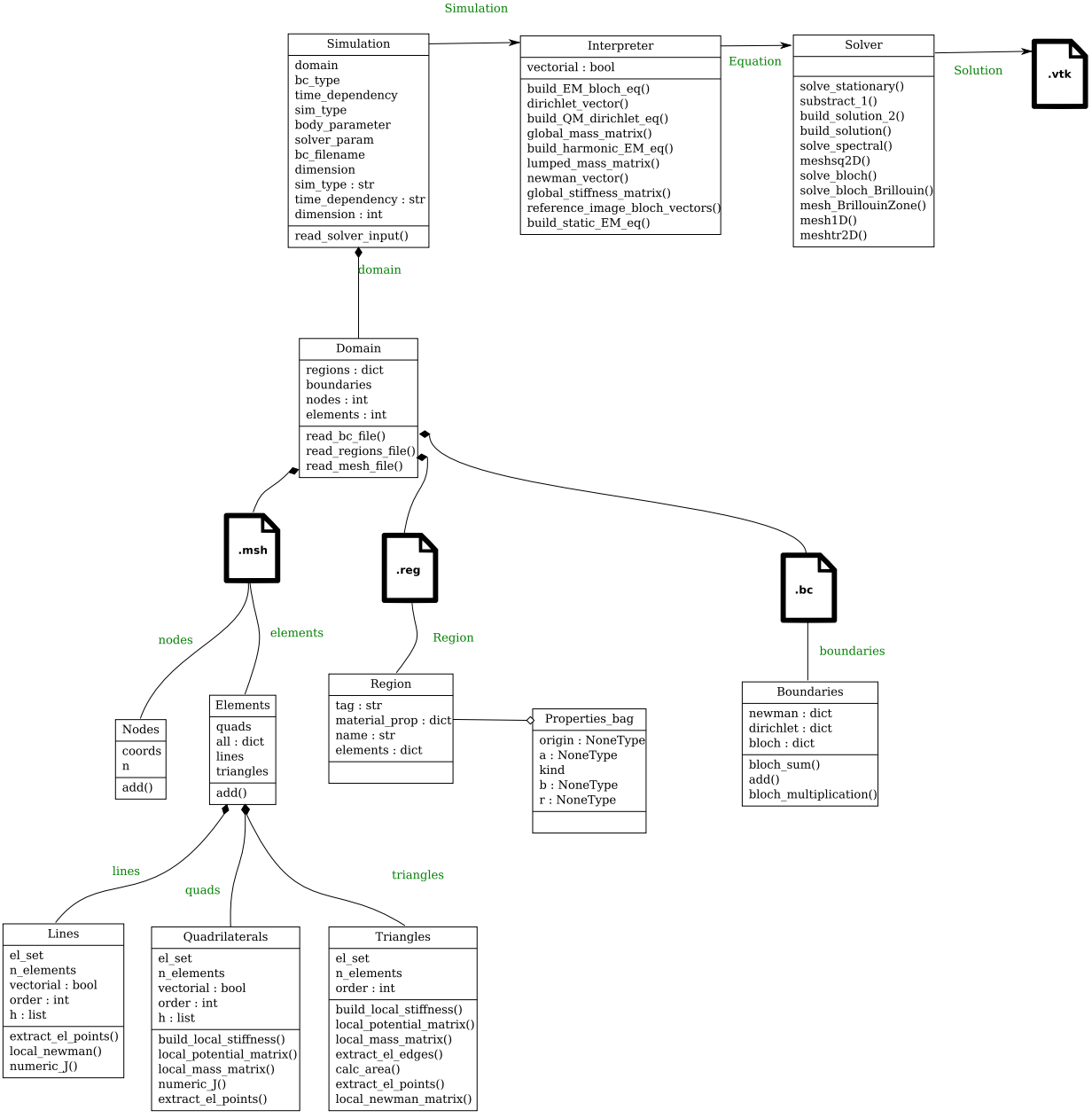

Figure 4.3: This diagram shows the main classes defined PeyeQM. Classes that are called by a method are connected with lines that end with a black diamonds, and classes that are attributes of other classes are connected by white diamonds.

```
"""
Created on Sat Apr  6 18:58:29 2013

This script solves the harmonic wave equation for electromagnetic fields in
a rectangular waveguide.

It reads the files with name stated in variable filename.

The necessary files are files with following extensions:

    .msh    Mesh file has information about nodes and elements
    .reg    Information about regions and material properties
    .bc     Boundary conditions pertinent for the simulation

The output will be a set of .vtk files written with the same name of write_solver_input.
The number of results depends on the number of eigenvalues and eigenvectors you wish
to compute.

Pseudo:
_____
 - Define path of main libraries
 - Import classes and functions
 - Define filename of input files
 - Define all the initial conditions that make the simulation and save them
     into the mesh file.
- Instantiate and populate the simulation
     - add a domain by reading from the mesh
     - add boundary conditions from .bc
     - add regions definitions from .reg
- Instantiate  the Interpreter and build the equation for harmonic problem
- Instantiate the solver and solve the equation for spectral problem
- fit the solution for a three component array, and export it to vtk files

@author: Santiago Echeverri Chacon
"""
```

Following that, (and as mentioned in the docstring) we have the definition of absolute path for the libraries that contain classes and functions, this is needed if the script is in a folder and not in the root directory of the libraries:

```
import os, sys
```

```
lib_path = os.path.abspath('../..')
sys.path.append(lib_path)
```

And then, we import the classes in order to use them afterwards:

```
from Classes import Simulation
from Interpreter import Interpreter
from Solver import Solver
from write import write_vtk, write_solver_input
from numpy import zeros
```

For convenience all the input files share the same name. We define the name and also write in the `.msh` file the conditions of the simulation. In the second statement we are saying that this will be a 2D Electromagnetic problem with Dirichlet conditions, and that it can find for boundaries if it looks in file with name filename and extension `.bc`:

```
filename = 'square_waveguide'
write_solver_input(filename +'.msh',dimension = 2, bc_type = 'Dir', \
parameter = [], eq = 'EM', sol_type = 'Stationary',analysis_param \
= ['y', 'y', 15, 15, 20, 20, 2], bc_filename = filename +'.bc')
```

Afterwards, we initialize an instance of class simulation and read the information that was written in the mesh with the method `read_solver_input()`. The nodes and elements are assigned to subclass Domain, by using the function `read_mesh_file()`. Boundary conditions are assigned to the domain by invoking the method `read_bc_file()`, and regions of the domain are defined by reading the `.reg` input file with method `read_regions_file()`.

```
simu = Simulation()
simu.read_solver_input(filename +'.msh')
simu.domain.read_mesh_file(filename +'.msh', True)
simu.domain.read_bc_file(simu.bc_filename)
reg_filename = simu.bc_filename.split('.bc')[0]
simu.domain.read_regions_file(reg_filename)
```

At this point the simulation is fully defined and ready to be interpreted. So, in the next lines we instantiate the interpreter and call the routine that builds the equation for harmonic fields. Notice that the argument of the function that assembles the equation is the instance of the simulation, so the interpreter has full access to everything we defined before.

```
inter = Interpreter()
eq = inter.build_harmonic_EM_eq(simu)
```

The next step is to call the solver and solve with the routine for harmonic simulations `solve_spectral()`. The output of the solver is an array with the eigenvalues and another array with the respective eigen functions.

```
my_solver = Solver()
value, fields = my_solver.solve_spectral(simu, eq)
print len(fields), 'value', value
```

After this command the eigenvalues are printed in the console. The following code snippet is not very well organized, but what it does is to lower the numbering of the elements by one because `.vtk` files use numbering that starts in 0. After that we format the fields that have two components into an array in which the third component z is assumed to be zero for every node. This is because `.vtk` requires that all components of a vector field are defined.

```
quads = my_solver.substract_1(simu.domain.elements.quads.el_set)
quads = quads[:,1:]
for i in range(len(fields)):
    field3 =  zeros((simu.domain.nodes.n,3))
    field3[:,0:2] = fields[i]
    fields[i] = field3
```

The final line of code is for writing into `.vtk` format. We use function `write_vtk()` and give it the nodes, elements and vector fields as arguments.

```
        write_vtk(filename +str(i)+'.vtk', 'MyTitle', 'UNSTRUCTURED_GRID' \
         ,simu.domain.nodes.coords,\
        quads, ['VECTORS', ['sol'], [fields[i]]])
```

4.5 Python

The software platform developed as part of this project was entirely written in Pyhon. Numerical routines for handling of matrices and the solution of systems of equations, or eigenvalue problems where taken from python based scientific packages: scipy and numpy

Python is a high level programing language oriented towards dynamic objects. Moreover it has been designed such that it emphasizes code readability, Python's syntax allows programmers to express concepts in fewer lines of code than would be possible in low-mid level compiled languages such as C. It is thus similar to Matlab and JavaScript in the sense that it is an interpreted programming language that does not need compiling, and is supported in multiple platforms. Being so, it sacrifices some performance with optimization of developing time in mind[47]. Python is often used as a scripting language, it can call and execute other programs and it allows wrapping of

routines written in low level languages like C. All of these properties make it a relevant tool for teachers, scientists and engineers whose priority is development speed and clarity over machine level high performance [25, 48, 49, 50, 51, 52].

4.6 Gmsh

> "Gmsh is a 3D finite element grid generator with a build-in CAD engine and post-processor. Its design goal is to provide a fast, light and user-friendly meshing tool with parametric input and advanced visualization capabilities" [53].

We used Gmsh as both a Computer Aided Design (CAD) tool and mesh generator for the modelling of simulation domains used in most part of the results presented in 5. It could be said that Gmsh is the main pre-processing tool used by PeyeQM.

Gmsh CAD and meshing modules are design such that its input and output can be parsed using ASCII text files written in Gmsh's own scripting language. In this project we took advantage of this, and developed the input format of the simulator over the same syntax as Gmsh otuput `.msh` mesh files. Moreover, PeyeQM's module for the creation of geometries (`gmsh_library.py`) used to define finite crystals with defects, exports geometry files `.geo` that are proper inputs for Gmsh. Using that module we can bypass the CAD module of Gmsh and construct arbitrary crystal-like geometries that are be interpreted and meshed by the meshing tool of Gmsh. The main classes used to define geometrical objects that are then translated to gmsh syntax are shown in the diagram of figure 4.4

4.7 Paraview

> "ParaView is an open-source, multi-platform application designed to visualize data sets of size varying from small to very large" [54, 55].

It is built on top of the Visualization Tool Kit libraries(VTK), so given a *.vtk* file format that contains data one can generate all sorts of graphs and plots from it, and interactively extract quantitative and qualitative information. Paraview is used externally in this implementation as the principal post processing tool. The solution of FE simulations is saved to *.vtk* files by means of writing with routine `write_vtk()` found in library write.py. All the images from chapter 5 (except for dispersion plots) were generated using Paraview.

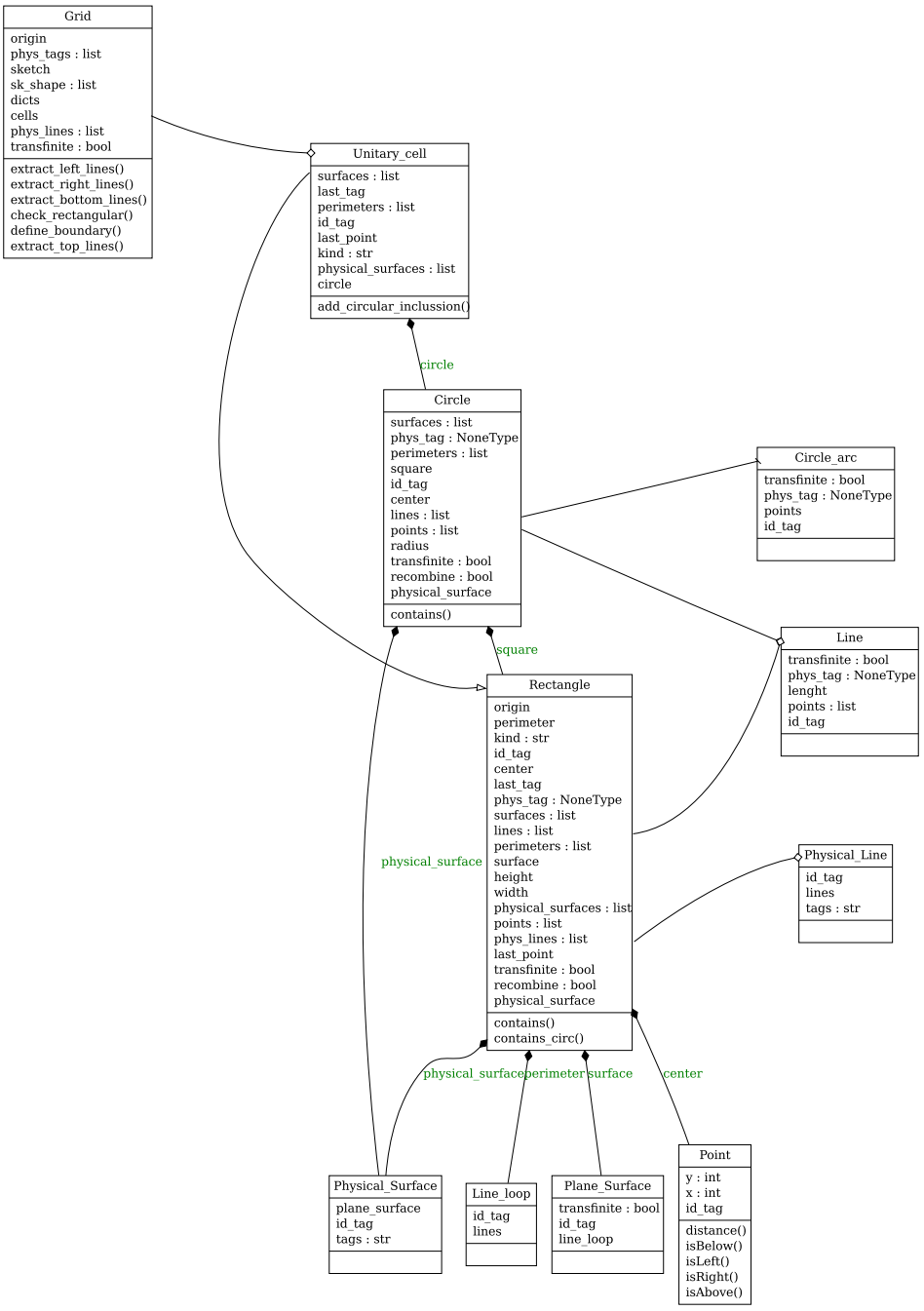

Figure 4.4: This diagram shows the classes defined in the module gmsh_library.py. Classes that are called by a method are connected with lines that end with a black diamonds, inheritance is shown with a white arrow, and classes that are attributes of other classes are connected by white diamonds.

Chapter 5

Results

As mentioned in the introduction, we developed a software platform able to simulate propagation of electromagnetic waves in Photonic Crystals. In chapter 2 we stated the equations that rule the problem, and discussed background behind one method of solution using FEA in chapter 3. In this chapter the solutions obtained by applying these concepts are presented and compared with either their analytic solutions or numerical solutions from the literature.

5.1 Electrostatic benchmark tests

Now, platform PeyeQM was built in a bottom-up process that started by making a FE solver that could handle static scalar problems of electron confinement in 2D potential wells for application in Quantum Mechanics[1]. This project involved an upgrade of that platform that began with the implementation of support for vector formulations, and a transition to Object Oriented Paradigm that could improve flexibility and maintenance.

The first section of this chapter shows the results obtained for Electrostatic benchmark tests, and were made in order to assert the accuracy of the static vectorial solver, and routines for the construction of stiffness matrices.

5.1.1 Electric field due to charged elements

Parallel plate capacitor

The first experiment was to simulate the most basic field we could think of, this is, that inside a parallel plate capacitor, where field lines go straight from one plate to the other.

[1] At first the PeYeQM was able to solve time independent Poisson and Schrödinger equations in 1D and 2D.

Electric fields satisfy equation 2.5. If there are no sources or sinks ($-\frac{\partial \mathbf{B}}{\partial t} = 0$), then the field $\mathbf{E}$ is said to be *irrotational* and by an identity of vector calculus we know that there must exist a scalar field V whose gradient is a valid Electrostatic vector field:

$$\nabla \times \nabla V = 0 \tag{5.1}$$

Scalar equations are easier to solve and are very useful when the medium is homogeneous and isotropic, this is the reason why many books stick with scalar problems and leave the process of obtaining $\mathbf{E}$ as performing on V the derivations in operator ∇. Using the scalar potential as an intermediate step for the solution of $\mathbf{E}$ stops being a good idea when interfaces between mediums and sharp geometries appear, and a direct solution using vector fields gains relevance.

In a parallel plate capacitor like that in figure 5.1, we consider two (infinite, parallel, plane, perfectly conducting, charged) plates that occupy the planes $x = 0$ and $x = d$ kept at potentials $V = 0$ and $V = V_0$ respectively. With plates of very large dimensions compared to the spacing between them, the potential becomes a function of x only, and equation 5.1 becomes:

$$\frac{d^2V}{dx^2} = 0$$

Integrating twice, we get:

$$V(x) = Ax + B \tag{5.2}$$

Where A and B are constants of integration solved by using boundary conditions.

$$V(0) = A0 + B = 0 \quad or \quad B = 0$$
$$V(d) = Ad + B = V_0 \quad or \quad A = \frac{V_0}{d}$$

Making the particular solution for the potential:

$$V = \frac{V_0}{d}x \quad for \quad 0 < x < d$$

The field is then obtained by taking the gradient of V, and we add a $-$ sign taking into account that $\mathbf{E}$ is a conservative field.

$$\mathbf{E} = -\nabla V = -\frac{dV}{dx}\hat{a}_x = -\frac{V}{d}\hat{a}_x \tag{5.3}$$

This means that the field is uniform and directed from the higher potential plate to the lower potential plate as shown in 5.1 [56].

A rectangular region was meshed and Dirichlet boundary condition were applied in the bottom and top plates as a field pointing down. Separation d is taken unitary and the result is shown in figure 5.2.

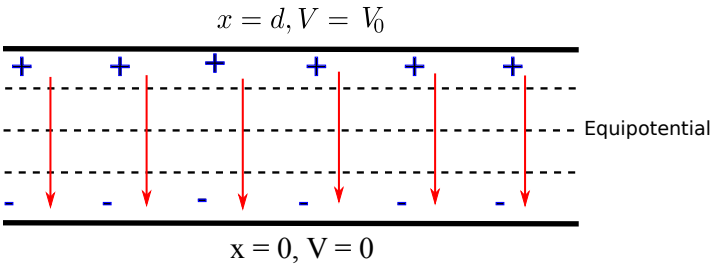

Figure 5.1: Cross sectional view of parallel plate capacitor.

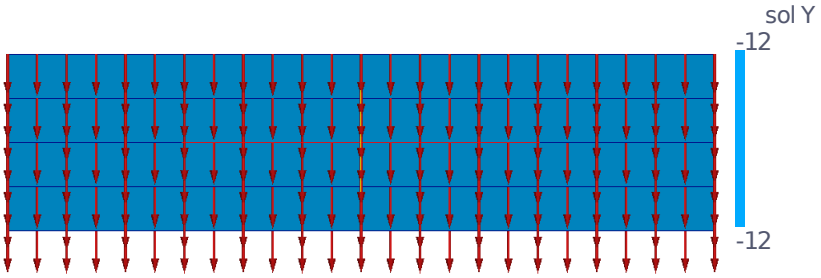

Figure 5.2: Solution of electric field in parallel plate capacitor.

It can be seen that the lines don't change in either magnitude or direction, and go from the charged plate wit potential V_0 to the bottom plate as expected.

This test showed that the routines for assembling stiffness matrices are capable of continuing a simple field over a distance between to boundaries.

The script for producing this simulation and the necessary input files can be seen in the *repository.*

Charged cylinder

The next field problem to solve in order to test PeYeQM was one with circular geometry. The solution for the field due to a point charge or a charged cylinder was used to compare the precision of the method for representing fields with two components.

Gauss law 2.3 in spherical coordinates, assuming a homogeneous medium with scalar permittivity is:

$$\epsilon \int_S \mathbf{E} \cdot d\mathbf{S} = \int_V \rho dV \tag{5.4}$$

Where $\int_V \rho dV = q$ is the charge of the particle. If that charge is distributed uniformly, then there are no preferred directions and the electric field must be radial [57] and have constant intensity

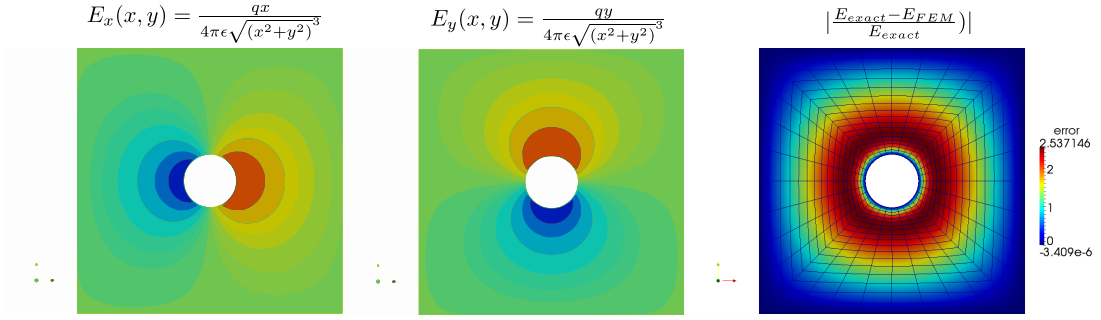

Figure 5.3: Whole simulation of Electric field due to a charged cylinder. Left, Numerical result for the x component of the field. Middle, x component of the solution, and, right, mesh and error calculated between analytic formula and results.

over spherical shells. So for an arbitrary spherical shell of radius r:

$$\epsilon \mathbf{E} \int_\theta \int_\phi r^2 sin(\theta) d\phi = q$$
$$\mathbf{E}_r(r) = \frac{q}{4\pi\epsilon r^2}\hat{r}$$

This has to be transformed into Cartesian coordinates in order to input boundary conditions. If we consider r as the magnitude of vector $\vec{r}$ in the unitary direction $\hat{r}$ then we can transform it to a linear combination of unitary vectors $\hat{x}$ and $\hat{y}$ by the following:

$$\vec{r} = rcos(\theta)\hat{x} + rsin(\theta)\hat{y} \tag{5.5}$$

And by trigonometry we know that $cos(\theta) = \frac{x}{r}$ and $sin(\theta) = \frac{y}{r}$. Making the necessary substitutions we get to:

$$\mathbf{E}_r(r) = \mathbf{E}_x(x,y) + \mathbf{E}_y(x,y) = \frac{qx}{4\pi\epsilon\sqrt{(x^2+y^2)}^3}\hat{x} + \frac{qy}{4\pi\epsilon\sqrt{(x^2+y^2)}^3}\hat{y} \tag{5.6}$$

So, a plot the solution after simulating should match a representation of this analytic result. We initially proposed a model where the charged cylinder or point source is modelled as a whole circle inside a square domain, the results of the simulation (figure 5.3) showed that for this kind of problem where the field decays in the form $\frac{1}{r^2}$ it was better to use finer meshes and take advantage of symmetries to reduce the computational cost. Taking just an eight of the model, and reshaping the outer boundary to a circle improved the quality of the elements and allowed us to use a finer mesh. figure 5.4 shows the y and x components of the field as well as the error between the simulation and the analytic solution. For the simulation factor $\frac{q}{2\pi\epsilon}$ has been normalized to 1, and the mesh was done done using 1996 QUAD8 elements with 5525 nodes, thus 11050 degrees of freedom. As a comparison, the simulation of the whole circle involved 2432 elements and 7040 nodes with 14080 degrees of freedom, but it solved for a domain that was eight times bigger.

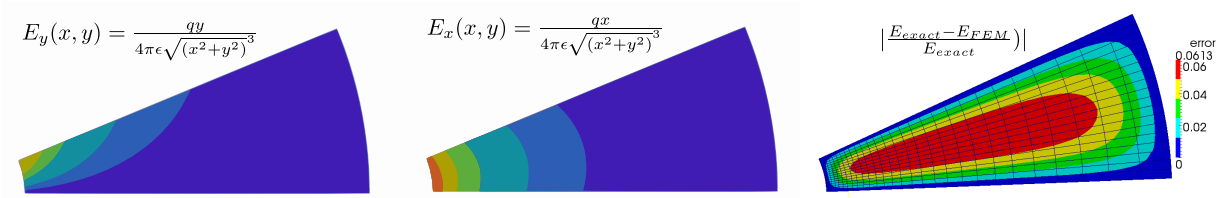

Figure 5.4: Segment of the simulation for Electric field due to a charged cylinder. a) Numerical result for the y component of the field, b) x component of the solution, c) Mesh and error calculated between analytic formula and results.

Electric Dipole, two charged spheres.

A similar simulation was performed using two semicircular surfaces, taking advantage of inversion symmetry along x axis. Boundary conditions are Dirichlet conditions over the circles and the rectangular boundaries on top, and Neumann conditions set to 0 on bottom lines. The analytic solution is simply the overlapping of the field due to charge 1 and charge 2, having one of them centered in the origin, and the other displaced by a distance d. The field everywhere can be expressed as:

$$\mathbf{E}_1(x,y) = x\left(x^2+y^2\right)^{-\frac{3}{2}}\hat{x} + y\left(x^2+y^2\right)^{\frac{3}{2}}\hat{y} \tag{5.7}$$

$$\mathbf{E}_2(x,y) = -(x-4)\left((x-4)^2+y^2\right)^{-\frac{3}{2}}\hat{x} - y\left((x-4)^2+y^2\right)^{\frac{3}{2}}\hat{y} \tag{5.8}$$

$$\mathbf{E}_x = \left[x\left(x^2+y^2\right)^{-\frac{3}{2}} - (x-4)\left((x-4)^2+y^2\right)^{-\frac{3}{2}}\right]\hat{x} \tag{5.9}$$

$$\mathbf{E}_y = \left[y\left(x^2+y^2\right)^{\frac{3}{2}} - y\left((x-4)^2+y^2\right)^{\frac{3}{2}}\right]\hat{y} \tag{5.10}$$

Where again $\frac{q}{2\pi\epsilon} = 1$ for the left charged circle and -1 for the one on the right at a distance $d = 4$. Figure 5.5 shows components x and y of the resulting field as obtained from the FE solver. The direction of the field lines can be observed by means of arrow glyphs in figure 5.6, where field lines go outwards in the positively charged cylinder and inwards in the negative. In the same way as before, a comparison between analytic and numerical solutions was performed (figure 5.7) and error appeared to be magnified in the region between cylinders. This value keeps being higher than expected, and different meshing schemes were tested in order to improve accuracy. Less error was observed using adaptative meshing methods than structured grids. The current mesh for this simulation has 510 elements and 1622 degrees of freedom, a future simulation is proposed with finer meshes using a PC with bigger RAM.

$$\mathbf{E}_x = \left[x\left(x^2+y^2\right)^{-\frac{3}{2}} - (x-4)\left((x-4)^2+y^2\right)^{-\frac{3}{2}}\right] \qquad \mathbf{E}_y = \left[y\left(x^2+y^2\right)^{\frac{3}{2}} - y\left((x-4)^2+y^2\right)^{\frac{3}{2}}\right]$$

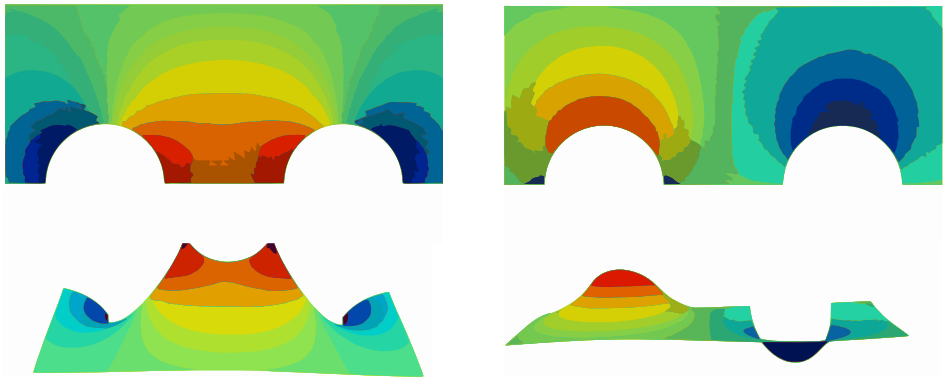

Figure 5.5: Components of the electric field after simulating a problem of two cylinders with opposite charges.

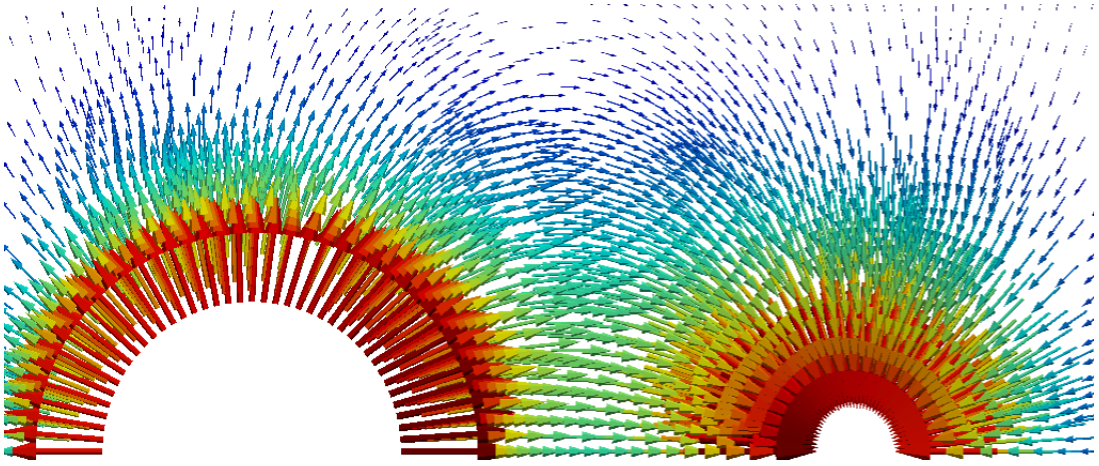

Figure 5.6: Arrow glyphs of the solution for the dipole problem. Length and color of the arrow indicate magnitude of the field. As expected, the field points from the positively charged cylinder to the negative.

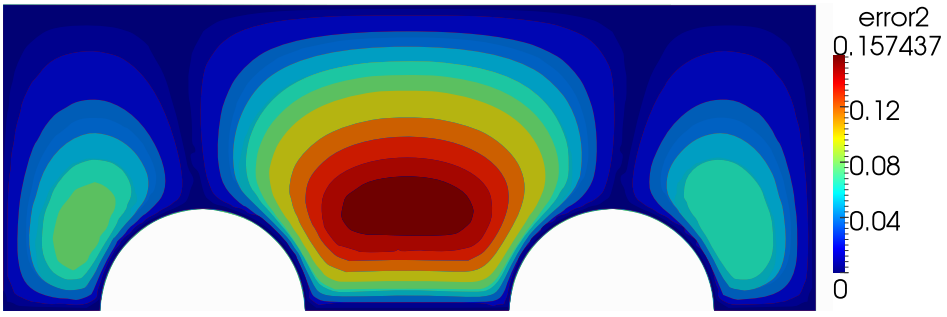

Figure 5.7: Error calculation of simulation against exact solution. The method reproduces the nature of the problem, but a finer mesh is needed to obtain precise results.

5.2 Harmonic benchmark tests

The second section regards solutions of eigenvalue problems, either for closed domains or infinite and periodic ones. The first part will deal with simulation of resonant modes in wave-guides, and the second with free waves and periodic crystals. In this stage of PeYeQM we knew that the program was capable of assembling stiffness matrices and solving systems of linear equations, what the following tests with wave-guides would give was the certainty that we were able to form mass matrices as well as solve the eigenvalue problems associated with a formulation of the form:

$$\mathbb{A}\tilde{\mathbf{E}} = \mathbb{M}\tilde{\mathbf{E}}$$

5.2.1 Eigenvalues and modes in wave-guides

Ideal wave-guides can be modelled by a source-less version of the wave equation for time harmonic fields 2.20 in arbitrary 2D domains closed by a metallic boundary that guarantees Dirichlet condition $\mathbf{E} = 0 \quad at \quad \Gamma$. This problem is relevant in electromagnetism and telecommunications because it is the principle behind many applications such as data transmission in fiber optics, and microwaves. Here, the 2D domain represents a cross section of a metallic structure that guides waves along its axis. We are particularly interested in knowing two things:

1. The modes or wave functions that are solution to the equation for a particular shape.
2. The frequencies associated to those modes or shapes.

And will start by introducing the analytic solution behind simple wave-guides such as rectangular and circular guides.

Rectangular wave-guides

In transverse magnetic fields (TM) that travel inside a wave-guide whose axis is over z, we have $\mathbf{H}_z = 0$ and solve for $\mathbf{E}$ components. Solutions of equation 2.29 where the medium inside the wave guide is a perfect dielectric, and for eigenvalues $\vec{k}^2 = \omega^2\mu\epsilon$ is a lengthy but straightforward process that involves separation of variables and use of boundary conditions:

$$\mathbf{E}_x = 0 \quad for \quad y = 0,\ 0 < x < a \tag{5.11}$$

$$\mathbf{E}_x = 0 \quad for \quad y = b,\ 0 < x < a \tag{5.12}$$

$$\mathbf{E}_y = 0 \quad for \quad x = 0,\ 0 < y < b \tag{5.13}$$

$$\mathbf{E}_y = 0 \quad for \quad x = a,\ 0 < y < b \tag{5.14}$$

A description of the solution process is left for the reader to look up in references such as Rao [56] or Jianming [1]. What is found in the process is that only certain discrete frequencies are allowed inside the wave-guide, and they are given by:

$$\omega_{n,m} = \frac{1}{\sqrt{\mu\epsilon}}\sqrt{\left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2} \qquad m,n = 0,1,2,\cdots \quad (m = n \neq 0) \tag{5.15}$$

Where ω is indexed by integers n, m and the trivial solution $\omega = 0$ is excluded. a is the width of the rectangular cross section and b is its height. The wave functions that are solution to this eigenproblem for each component of the field are in the following general form:

$$\mathbf{E}_z = A \sin\frac{m\pi x}{a}\sin\frac{n\pi x}{b}e^{\mp jk_z z} \tag{5.16}$$

$$\mathbf{E}_x = \mp jB\frac{m\pi}{a}\cos\frac{m\pi x}{a}\sin\frac{n\pi x}{b}e^{\mp jk_z z} \tag{5.17}$$

$$\mathbf{E}_y = \mp jC\frac{n\pi}{b}\sin\frac{m\pi x}{a}\cos\frac{n\pi x}{b}e^{\mp jk_z z} \tag{5.18}$$

For our comparisons we will only look at how closely the shapes of the simulated solution represent $\mathbf{E}$ in 5.16, and how similar are the analytic and numeric natural frequencies of the guide ω.

The numerical solution for rectangular wave-guide has less degenerate modes, and the shapes look better.

Comparison of ω^2 for square wave-guide				
m,n	1,1	1,2	2,2	3,1
FEM	4.93480857	12.33721117	19.73961759	24.6763011
Anlytic	4.93480220	12.33700550	19.73920880	24.67401100

Comparison of ω^2 for rectangular wave-guide $a = 2b$				
m,n	1,1	2,1	3,1	1,2
FEM	3.08425459	4.93480795	8.0190859	12.33717191
Anlytic	3.08425137	4.93480220	8.0190535	12.33700550

Circular wave-guides

For a circular wave-guide the modes are

Comparison of $\frac{\omega}{a}$ for circular wave-guide				
m,n	0,1	1,1	2,1	0,2
FEM	2.40482634	3.83171343	5.13564189	5.52012378
Anlytic	2.40482555	3.83170597	5.13562230	5.52007811

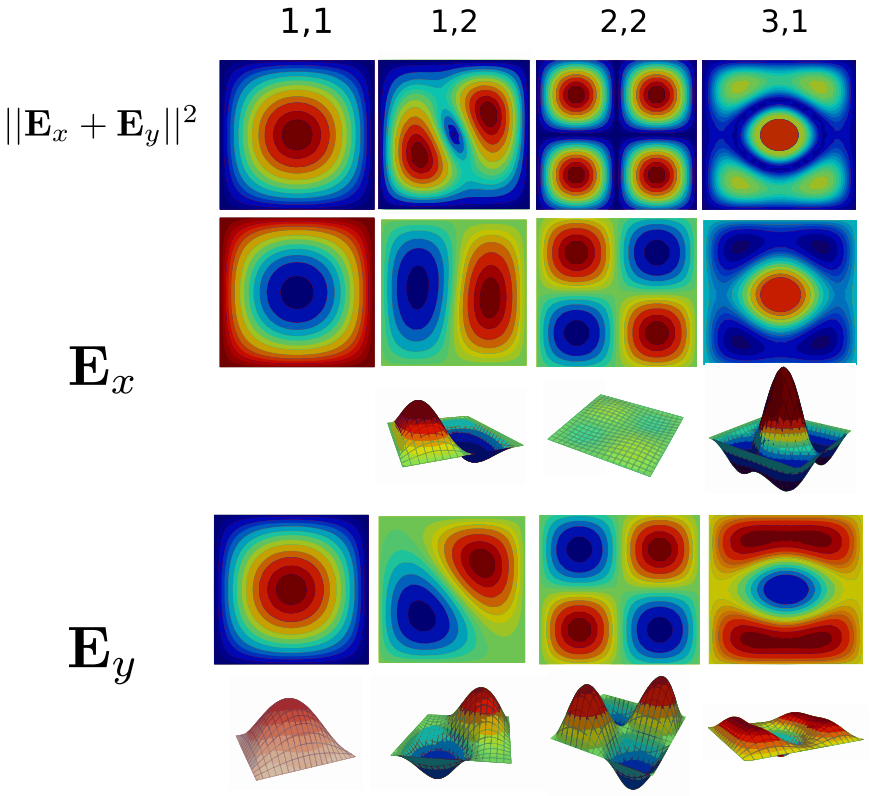

Figure 5.8: Results for three different modes from the simulation of a square shaped wave-guide with side $a = b = 2$

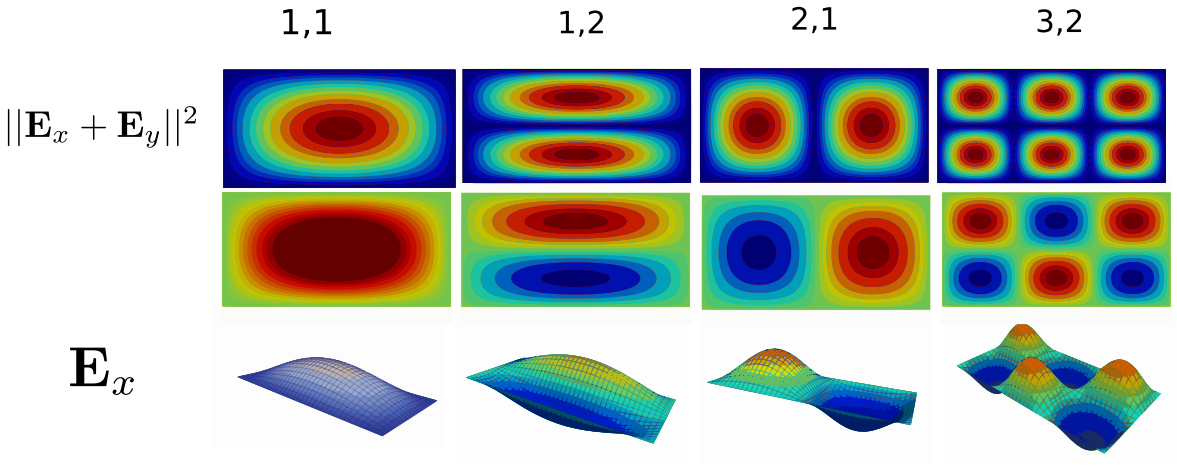

Figure 5.9: Results for three different modes from the simulation of a rectangle shaped waveguide with sides $a = 4$ and $b = 2$.

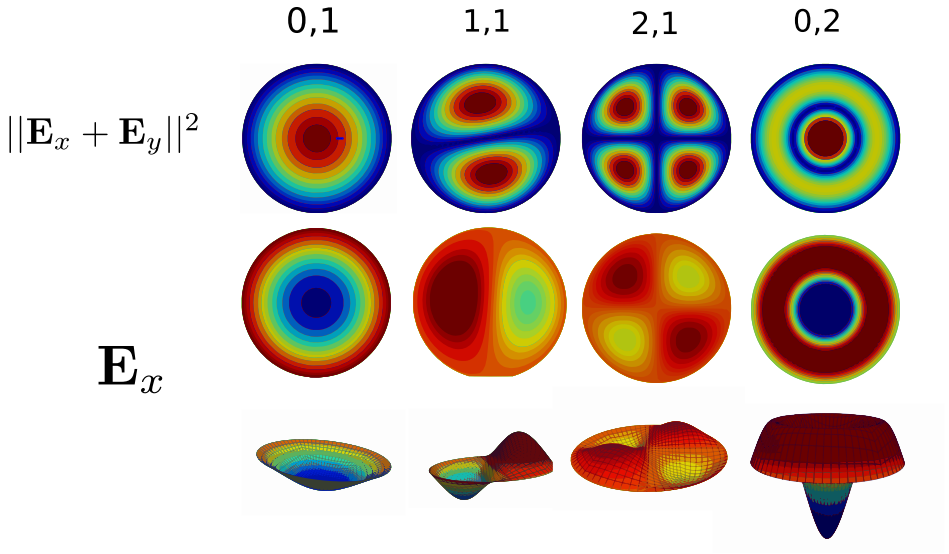

Figure 5.10: Results for three different modes from the simulation of circular wave-guide with radius $a = 2$.

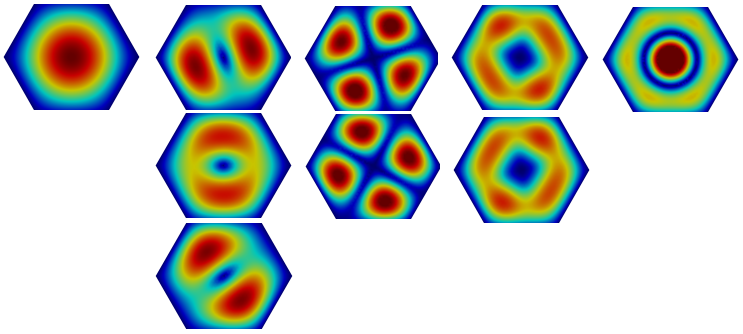

Figure 5.11: Degeneracy of modes from the simulation of hexagonal wave-guide.

Hexagonal wave-guide

Threefold degeneracy is seen in modes of hexagonal guides due to symmetries.

Comparison of ω^2 for hexagonal wave-guide				
m,n	0,1	1,1	2,1	0,2
FEM	5.34990939	8.51630317	11.39340788	12.2461672
Anlytic				

Elliptical waveguide

The closest to a reference from which to compare the solutions of elliptical waveguides was a study of oscillation in an elliptic membrane by J. Gutiérrez Vega [58]. Shapes shown in figure 5.12 are

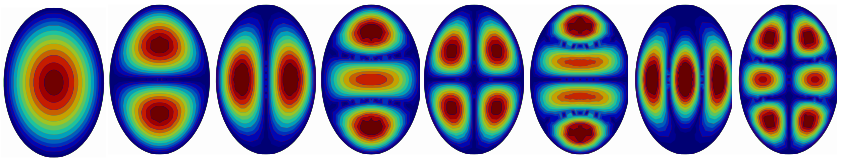

Figure 5.12: Results for the first 8 modes from the simulation of an elliptical wave-guide with minor axis $a = 2$ and mayor axis $b = 3$.

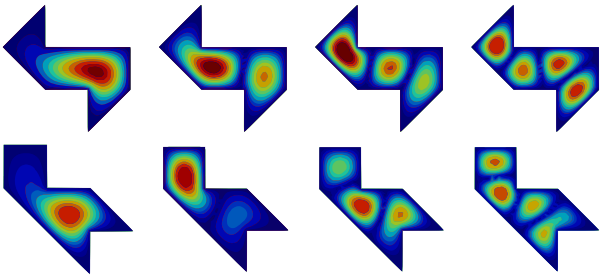

Figure 5.13: First 4 modes from the simulation of two isoespectral shapes as taken from [5].

in agreement with the images in the reference.

First four frequencies ω for elliptical wave-guide				
FEM	2.0195	2.4149	2.6627	2.9457

Isoespectral shapes

We also tested the solver by performing the computation of isoespectral shapes as proposed by Chapman in [5]. The idea behind this was to obtain the same eigenvalues from different shapes by taking advantage of possible mappings between figures that share certain properties. A comparison between eigenvalues is presented in table 5.2.1, and the first 4 non-degenerate modes are shown in figure 5.13.

Comparison of ω^2 for two isoespectral wave-guides				
Shape 1	6.38705867	7.65676809	9.11294713	10.23292227
Shape 2	6.3850403	7.65926504	9.11794073	10.23394057

Finite lattices

Now that the software was tested for the solution of eigenvalue problems, it needed to prove its ability to handle simulations of heterogeneous domains, like in Photonic Crystals. The following results are tests where the harmonic wave equation is solved for a finite grid of 4×4 circular dielectric slabs with ratio $\frac{r}{a} = 0.2$ and dielectric constant $\epsilon_r = 8.9$ surrounded by air. Boundary

conditions are as in 5.14, this is, inside a rectangular metallic boundary like if in a wave-guide. The expected result is to see concentration of field inside regions with higher ϵ. As well as lower natural frequencies than those produced by homogeneous medium due to the denominator $\sqrt{\mu\epsilon}$ in 5.15.

Comparison of ω for finite grid of dielectric rods vs air for the same mesh.				
m,n	1,1	1,2	2,2	3,1
Embedded rods	0.77475487	1.19190369	1.46064033	1.58328716
Air	1.11074305	1.75637223	2.22176045	2.48458998

Note in table 5.2.1 that if frequencies found for the air filled case are multiplied by two (this domain is twice as big) and then squared, we get the same result as in 5.2.1. Figure 5.14 illustrates the difference between solving for an homogeneous and non-homogenous problems. The flexibility to easily model interfaces and to couple regions is one of the great advantages of using both the Finite Element Method and a vectorial formulation. Notice that these are solutions that are hard to solve analytically.

5.2.2 Bloch periodic lattices and dispersion curves

Finally, we solved the harmonic equation for infinite crystals by defining Bloch periodicity boundary conditions over a unitary cell. Looping through combinations of phase changes at boundaries, allowed us to see the spectrum for the crystal given different incident plane waves defined by varying wave number $\vec{k}$. Dispersion plots shown in figures 5.16, 5.17, and 5.18 are made by taking only the perimeter of the irreducible Brillouin zone which is the minimal representation of the domain in frequency space. Figure 5.19 shows the complete representation of natural frequencies for the lattice given every combination of incident plane waves with wave number $\vec{k} = \vec{k}_x + \vec{k}_y$ inside a unitary cell. The results presented in figure 5.17 were made with the same conditions as those used by Joannopolous in page 68 of the book: **Photonic Crystals: molding the flow of light** [2] and agree with their findings. Apart from great similarity between the curves, our value of 31.41% for the gap-midgap ratio is close to the one reported by them of 31.4%.

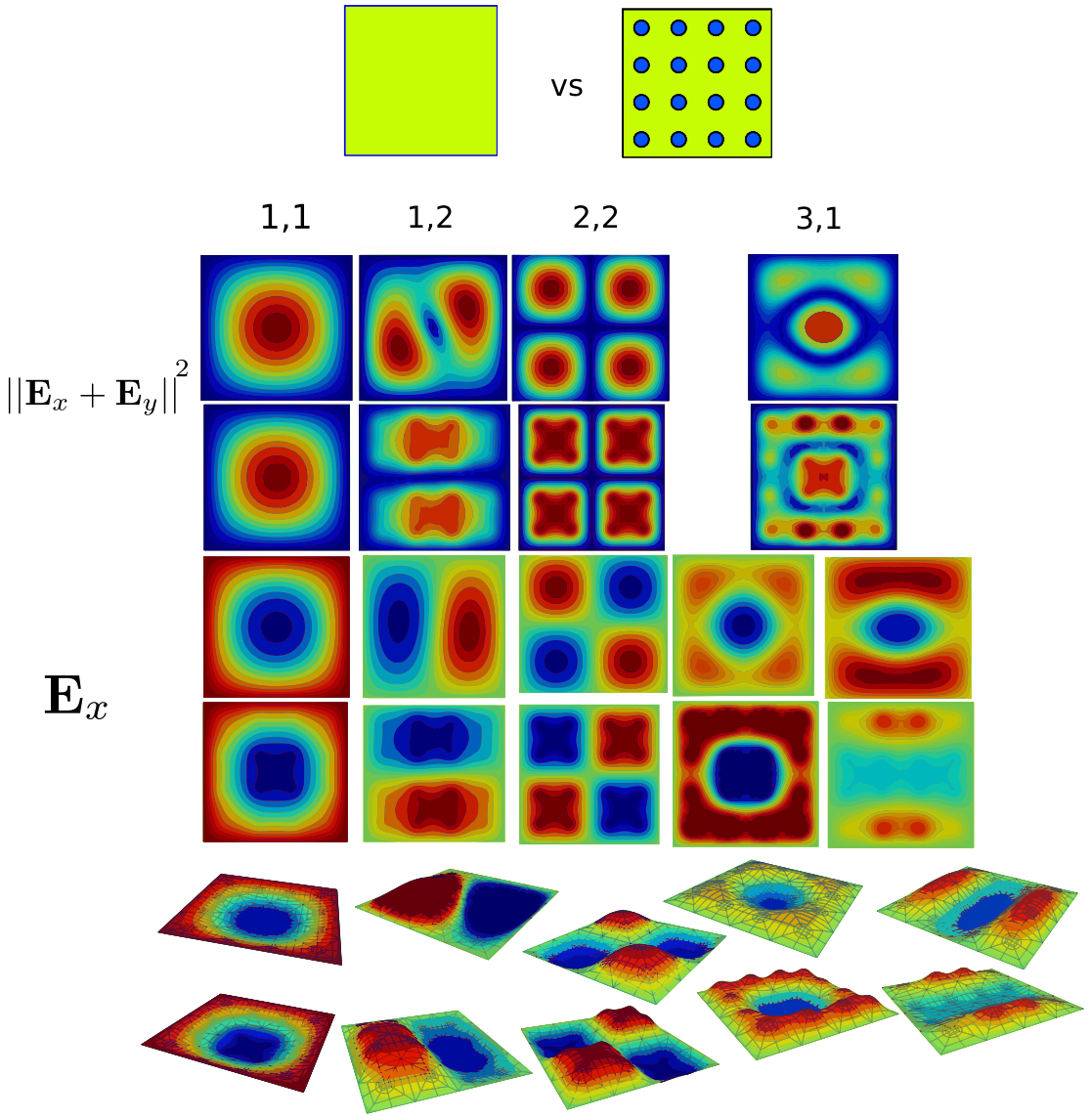

Figure 5.14: A lattice of dielectric rods embedded in a domain causes peaks of field magnitude in the places with higher dielectric constant.

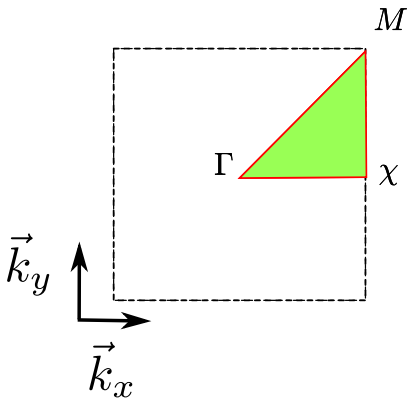

Figure 5.15: Illustration of reduced Brillouin zone in spectral domain, and reference points Γ, χ and M.

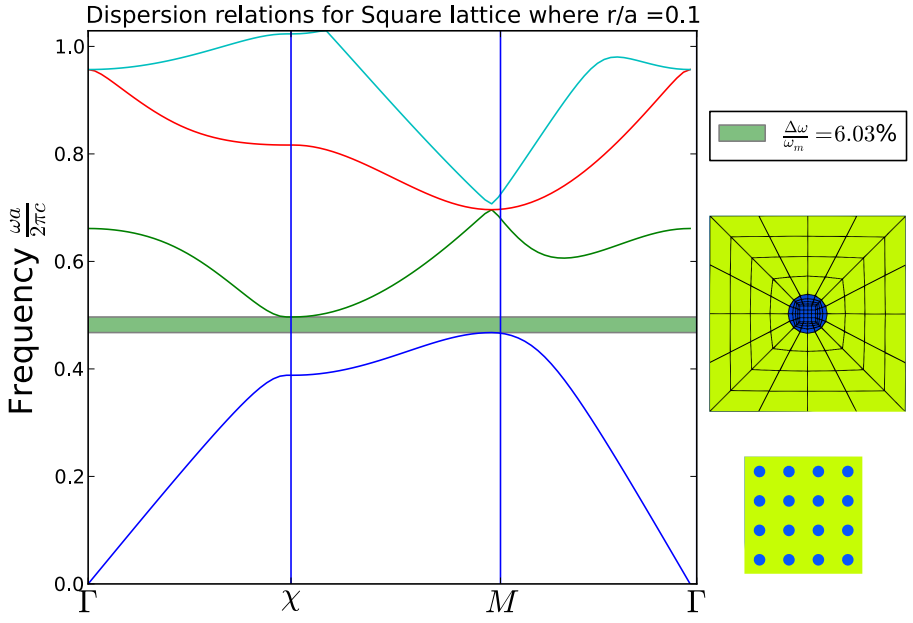

Figure 5.16: Dispersion plot for a square lattice of r/a = 0.1 and $\epsilon_r = 8.9$.

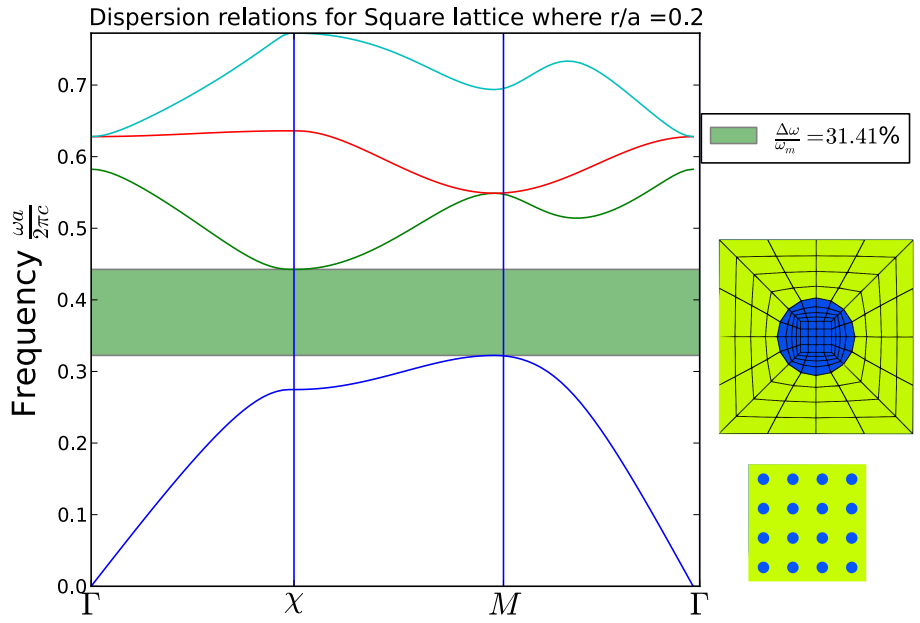

Figure 5.17: Dispersion plot for a square lattice of r/a = 0.2 and $\epsilon_r = 8.9$.

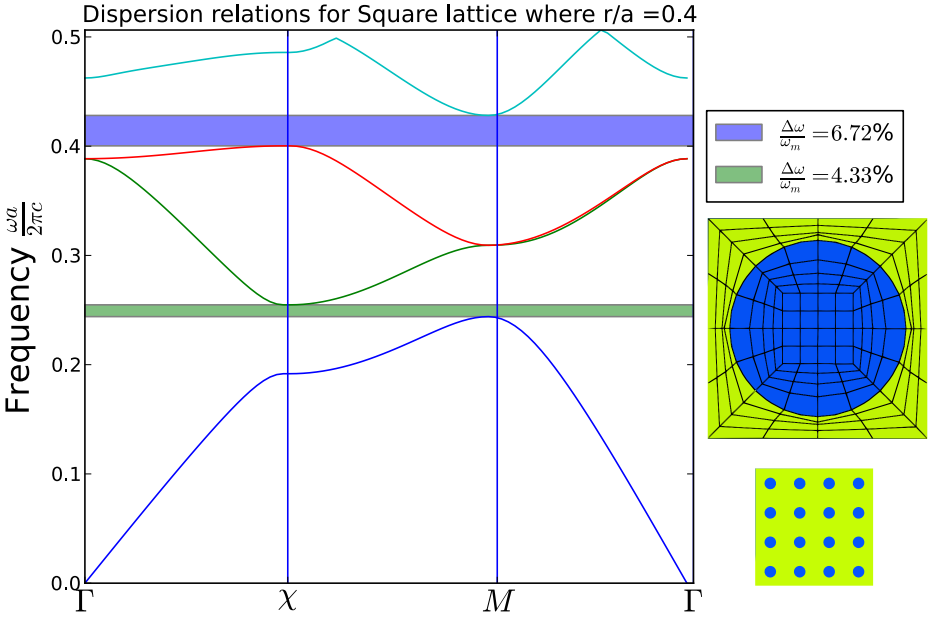

Figure 5.18: Dispersion plot for a square lattice of r/a = 0.4 and $\epsilon_r = 8.9$.

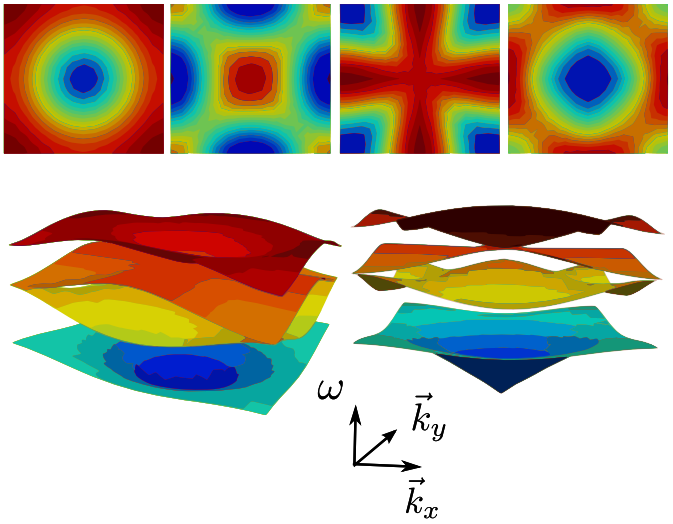

Figure 5.19: Frequency surfaces for square lattice of 5.17, where each point in a surface i represents the eigenvalue i of the solution, given a pair of $\vec{k}_x$, $\vec{k}_y$.

Chapter 6

Conclusions and future work

6.1 Conclusions

We developed the necessary computational tools for the Finite Element modelling, simulation and analysis of EM wave propagation in the context of Photonic Crystals. This was done as an ongoing open source software platform based on Python and structured around Object Oriented Paradigm. In order to build this platform, the necessary concepts and mathematical tools associated to the modelling of electromagnetic waves in periodic media were appropriated, specifically Maxwell's equations, and periodicity formulations taken from solid state physics.

Moreover, the finite element method was explored as an alternative to other methods commonly used in the analysis of PCs such as the Plane Wave Expansion (PWE) [30] method and Finite Differences Time Domain (FDTD)[21]. FEM was chosen mainly because it has proven to be a great general purpose method for investigating PCs [59] being its flexibility to model interfaces and defects in both frequency and spatial domains one of its greatest advantages. FEM is also well known for its stability and robustness and has proven its versatility in many fields of engineering [4, 60, 33, 61].

The platform was designed using the Object Oriented Paradigm having in mind the objective of having a well structured and flexible architecture, capable of being upgraded to future capabilities. Even though the architecture is in a developing stage and a proper manual for its use its pending, the modules, classes and methods that conform it are functional for production of results, and are documented for user reference. The reader is encouraged to visit the **online repository** and explore its examples.

We proved that PeyeQM is capable of accurately simulating 2D electromagnetism stationary, and harmonic vector fields as well as fields inside in-homogeneous domains and infinite crystals. Time domain simulations (FDTD) of finite lattices with point and line defects are still in developing

phase but look very promising. The need of CAD tools for the construction of finite lattices of dielectric rods with arbitrary size and placement was a factor that was underestimated in the beginning. And some time was spent on defining routines for the definition of clusters of dielectric regions within a rectangular matrix.

In the present time, we are working on fixing bugs, cleaning the code, and making a proper documentation of the software in order to achieve a product that can be shared and published as an article and or a downloadable application. This is a work that has great potential for academic and educational use by students of engineering and physics interested in diving into computational physics in the context of open source collaborative software.

6.2 Future work

The following is a list of items that are worth exploring in future work on PeyeQM platform:

- Definition of vectorial triangular elements for improving meshing flexibility and linear quadrilateral elements for computation speed. Triangular elements are simpler than quadrilaterals and are very often used by meshing algorithms in order to fill regions where quadrilateral elements get distorted. On the other hand, a comparison between convergence and speed between linear and quadratic elements would improve the understanding of the problem and ease calculations.
- Consider adding support for infinite boundary conditions for the simulation of unbounded domains [62, 63]. One popular candidate very often used in electromagnetism is the use of regions with Perfectly Matched Layers (PML)[1], where the domain is surrounded with a material that has artificial energy dissipation and the properties are fitted to minimize reflection between the interfaces. Other possibilities are:
  - to use a Robin boundary condition when intrinsic impedance is known, or to map the unbounded region to a known geometry (e.g, with a conformal mapping), both approaches used in the open source code FEMM (**Finite Element Method Magnetics**) [64];
  - hybrid FEM and Boundary Elements(BEM) as presented by Nicolás Guarín in [3];
  - Infinite Elements, that are special finite elements with an assumed decaying behavior ([33]).
- Evaluate convergence of the method for the problems described in chapter 5 using finer spatial and spectral ($\vec{k}$ domain) meshes.
- Perform more simulations of time dependent problems involving defects in finite crystals. And compare the results of Q factor and efficiency with publications.
- Integrate the CAD and solver into an optimization algorithm that maximizes transmission or confinement by inducing variations in position and shape of defects in the lattice.

- Consider parallelization schemes to be implemented in order to make bigger simulations plausible. Algorithms that split the computation into threads and take advantage of the parallel architecture of modern computers are the key to obtaining results of enormous simulation faster than before. Making research on things like weather and seismic simulation, molecular dynamics or drug compounds testing practical and realizable. With a parallel implementation of PeyeQM bigger domains and finer meshes would be able to be modelled in a robust workstation. It is to notice that nowadays parallel computation capacity is available for rent in places such as Amazon's ec2 service.

- Possible integration of (now external) processes such as meshing and visualization, by means of scripting that uses gmsh or Paraview Python libraries. This would make the process of solving more autonomous and free the user from external tasks.

Bibliography

[1] Jian-Ming Jin. *Theory and computation of electromagnetic fields*. Wiley, 2010. 5, 11, 12, 15, 17, 41, 65, 75

[2] John D. Joannopoulos. *Photonic Crystals, Molding the Flow of Light*. Princeton University Press, 2008. 5, 12, 14, 22, 25, 28, 69

[3] Nicolás Guarín. Simulación Numérica de Problemas de Propagación de Ondas: Dominios Infinitos y Semi-infinitos. Master's thesis, Universidad EAFIT, 2012. 5, 16, 26, 27, 28, 34, 75

[4] Klaus-Jürgen Bathe. *Finite element procedures*, volume 2. Prentice hall Englewood Cliffs, 1996. 5, 35, 39, 74

[5] S. J. Chapman. Drums that sound the same. *The American Mathematical Monthly*, 102:124–138, 1995. 7, 68

[6] Salah Obayya. *Computational Photonics*. Wiley, 2011. 14

[7] Nader Engheta. *Metamaterials: physics and engineering explorations*. Wiley Inter Science, 2006. 14

[8] Filippo Capoliono. *Theory and Phenomena of Metamaterials*. CRC Press, 2009. 14

[9] Lawrence Cowsar Gang Bao. *Mathematical Modeling in Optical Science*. Society for Industrial and Applied Mathematics, 1987. 15

[10] F.X. Kartner, S. Akiyama, and G. Barbastathis. Electronic Photonic Integrated Circuits for High Speed, High Resolution, Analog to Digital Conversion. *Proc. of SPIE*, 6125:612503–1, 2006. 15

[11] A.M. Apetrei and J.M. Moison. Electromagnetic field confined and tailored with a few air holes in a photonic-crystal fiber. *Applied Physics B: Lasers and Optics*, 2005. 15

[12] Nadia K. Pervez. Photonic crystal spectrometer. *Optics Express*, 18, 2010. 15

[13] Eli. Yablonovich. Inhibited spontaneous emission in solid-state physics and electronics. *Physical*, 58:2059–2062, 1987. 15

[14] E. Yablonovich. Photonic band structure: The face-centered-cubic case employing nonspherical atoms. *Physical Review Letters*, 67, 1991. 15

[15] Pierre R. Villeneuve. Microcavities in photonic crystals: Mode symmetry, tunability, and coupling efficiency. *Physical Review B*, 54, 1996. 15

[16] Hatice Altug. Ultrafast photonic crystal nanocavity laser. *Nature Physics*, 2, 2006. 15

[17] S.A. Moore. Photonic crystal laser with mode selective mirrors. *Optical Society of America*, 16, 2008. 15

[18] Janik Wolters. Enhancement of the zero phonon line emission from a single nitrogen vacancy center in a nanodiamond via coupling to a photonic crystal cavity. *Applied Physics Letters*, 2010. 15

[19] J. D. Steven G. Johnson, Joannopoulos. Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis. *Optical Society of America*, 2001. 15, 28

[20] Yhefferson Gutierrez Loaiza. Estructura de bandas en un cristal fotónico bidimensional. Master's thesis, Universidad EAFIT, 2011. 15, 16, 28

[21] Ardavan F. Oskooi, David Roundy, Mihai Ibanescu, Peter Bermel, J.D. Joannopoulos, and Steven Johnson G. Meep: A flexible free-software package for electromagnetic simulations by the FDTD method. *Computer Physics Communications*, 2009. 15, 74

[22] Vitaly Félix Rodriguez. Finite-Element Analysis of Photoinc Crystal Cavities: Time and Frequency Domains. *Journal of Lightwave Technology*, 23:1514–1521, 2005. 15

[23] Koshiba Masanori. Time-domain beam propagation method and its application to photonic crystal circuits. *Journal of Lightwave Technology*, 18:102–110, 2001. 15

[24] Sara E. Rodriguez. Numerical Analysis of the Modal Coupling at low resonances ina a Colombian Andean Bandola in C using the Finite Element Method. Master's thesis, Universidad EAFIT, 2012. 16

[25] Edward Villegas. Modelización y Simulación de Efectos de Confinamiento en el Grafeno. Master's thesis, Universidad EAFIT, 2011. 16, 56

[26] Santiago Echeverri. Implementación del método de elementos finitos para la ecuación de Schrodinger en estructuras periódicas. Unpublished project report, 2011. 16

[27] Daniel E. Sierra. Desarrollo de un módulo computacional para la síntesis y procesamiento de campos ópticos con aplicaciones en holografía digital y speckle. Master's thesis, Universidad EAFIT, 2010. 16

[28] Charles Kittel. *Introduction to Solid State Physics*. Joh, 2005. 17, 26

[29] JD Jackson. *Classical Electrodynamics*. John Wiley & Sons: New York, 1998. 17

[30] John Joannopoulos Steven Johnson. Block-iterative frequency-domain methods for maxwell's equations in a planewave basis. *Optics Express*, 8:173–190, 2001. 22, 74

[31] Steven G. Johnson. Notes on the algebraic structure of wave equations, 2007. 28

[32] Herbert Goldstein, Charles P. Poole Jr., and John L. Safko. *Classical Mechanics*. Addison-Wesley, 3 edition, 6 2001. 30

[33] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, and Jian Z Zhu. *The Finite Element Method: Its Basis and Fundamentals: Its Basis and Fundamentals*. Butterworth-Heinemann, 2005. 30, 74, 75

[34] J.N. Reddy. *Applied Functional Analysis and Variational Methods in Engineering*. Krieger Publishing, 1 edition, 1991. 30

[35] Manuel Julio García. *Lecture Notes on Numerical Analysis*. 2011. 32

[36] Jean-Claude Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35(3):315–341, 1980. 41

[37] Fumio Kikuchi. Theoretical analysis of Nedelec's edge elements. *Japan journal of industrial and applied mathematics*, 18(2):321–333, 2001. 41

[38] Gerrit Mur. Edge elements, their advantages and disadvantages. *IEEE Transactions on magnetics*, 30:3552–3557, 1994. 41

[39] J.P. Webb. Edge elements and what they can do for you. *IEEE Transactions on magnetics*, 29:1460–1465, 1993. 41

[40] Gerr. The fallacy of edge elements. *IEEE Transactions on magnetics*, 34:3244–3247, 1998. 41

[41] Michael E. Plesha Robert D. Cook, David S. Malkus. *Concepts and applications of Finite Element Analysis Third edition*. John Willey and Sons, 1989. 45

[42] Jaroslav Mackerle. Object-oriented programming in fem and bem: a bibliography (1990-2003). *Advances in Engineering Software*, 35:325–336, 2004. 47

[43] C. Lage. Concept oriented design of numerical software. Technical report, Eidgenössische Technische Hoschule Zürich, 1998. 47, 48

[44] The Computer Language Company Inc. 48

[45] P.L Raymond. A metric for software readability. *International Symposium in Software Testing and Analysis*, 2008. 48

[46] Wikipedia. Object-oriented programming, 2013. 48

[47] Fotis Georgatos. How applicable is python as first computer language for teaching programming in a pre-university educational environment, from a teacher's point of view? Master's thesis, AMSTEL Institute, Faculty of Science, Universiteit van Amsterdam, June 2002. 55

[48] Jaan Kiusalaas. *Numerical Methods in Engineering with Python*. Cambridge University Press, 2005. 56

[49] A. Bäcker. Computational physics education with python. *Computing in Science and Engineering*, pages 30–47, 2007. 56

[50] EuroScipy tutorial. *Python Scientific lecture notes*. team, July 2010. 56

[51] P.H. Borcherds. Python: a language for computational physics. *Computer Physics Communications*, 177:199–201, 2007. 56

[52] Hans Petter Langtangen. *Python Scripting for Computational Science*. University of Oslo, 2004. 56

[53] C. Geuzaine and J.F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. 56

[54] A. Henderson. *ParaView Guide, A Parallel Visualization Application*. Kitware Inc, 2007. 56

[55] Paraview documentation. Accessed 2 October 2011. 56

[56] Nannapaneni Narayana Rao. *Elements of engineering electromagnetics*. Pearson Prentice Hall, 2004. 59, 65

[57] David Keung Cheng. *Fundamentals of Engineering Electromagnetics*. Addison-Wesley, 1993. 60

[58] Ramón Rodriguez-Dagnino J. Gutierrez-Vega, S. Chávez-Cerda. Free oscilations in an elliptic membrane. *Revista Mexicana de Física*, 45:613–622, 1999. 67

[59] Imanol Andonegui. The finite element method applied to the study of two-dimensional photonic crystals and resonant cavities. *Optics Express*, 21:4072–4092, 2013. 74

[60] L Ramdas Ram-Mohan. *Finite element and boundary element applications in quantum mechanics*, volume 5. Oxford University Press on Demand, 2002. 74

[61] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The fenics book*, volume 84. Springer, 2012. 74

[62] Xavier Antoine, Anton Arnold, Christophe Besse, Matthias Ehrhardt, and Achim Schädle. A review of transparent and artificial boundary conditions techniques for linear and nonlinear schrödinger equations. 2009. 75

[63] Daniel Appelö. *Non-reflecting Boundary Conditions for Wave Propagation Problems*. PhD thesis, KTH, 2003. 75

[64] David Meeker. Finite element method magnetics. *User's Manual*, 4:158, 2010. 75