UNIVERSITY OF
Southampton

Algorithms and Technologies for Photonic Crystal Modelling

by

Elizabeth E. Hart

Thesis submitted for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Engineering Sciences
Computational Engineering Design Group

November 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ENGINEERING SCIENCES
COMPUTATIONAL ENGINEERING DESIGN GROUP

Doctor of Philosophy

by Elizabeth E. Hart

In this thesis an investigation into the behaviour of light when passing through photonic crystals was carried out using numerical methods. Photonic crystals are expensive and difficult to fabricate so there is a requirement for computer simulations that can quickly and accurately model how the crystal structure will affect the behaviour of light. A finite difference method was written to model two-dimensional photonic crystals. The results from the finite difference method modelling agreed with known results for standard photonic crystal structures created by the plane wave expansion method. Once validated the finite difference method was used in a genetic algorithm optimisation. It found that novel shaped rods can increase the size of photonic band gaps when compared with cylindrical rods.

A new meshless method algorithm was developed to solve Maxwell's equations. Simulations were carried out using an equation with known analytical solutions; how the accuracy of the results was affected by different designs of experiment and different radial basis functions was recorded. The meshless method was developed further to model photonic crystals. The meshless method requires the creation of large dense matrices and then forms a generalised eigenvalue problem. A new set of algorithms were developed that can model photonic crystals accurately. Exploration of alternative technologies was carried out to try to obtain a speed up in the modelling process. A graphics processing unit was used to do general purpose computation. Graphics processing units generally show significant speed up when compared to central processing unit for filling the matrices required for the meshless method. For accelerating numerical methods a heterogenous approach is preferable to a strict graphics processing unit implementation.

Nature has evolved complex nanostructures that provide very specific and often very special optical effects, at present these are not well understood and cannot be replicated. In this thesis a new meshless method has been developed which will enable the development of complex crystal geometries.

Contents

List of Figures

List of Tables

Listings

Declaration Of Authorship

I, *Elizabeth E. Hart*, declare that the thesis entitled *Algorithms and Technologies for Photonic Crystal Modelling* and the work presented in it are my own. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as journal papers [5].

Signed: .... Date: ....

Acknowledgements

I would like to thank my supervisors Prof. Simon Cox and Dr Kamal Djidjeli for there help and guidance over the last 3 years. I would also like to thank Dr Marc Molinari, Dr András Sóbester and Dr Ken Thomas for their guidance and expertise. I am greatful to Microsoft for their financial support.

Thanks for my friends and family for all their support. A big thanks to my parents for being there when things got tough. Thanks to my friends Emma Fosbrook and Jennie Langham-Brown for listening and making tea in a cup. Thanks to Dr Steven Johnston, Dr Gang Xue, Greg Jasion, Neil O'Brien, Alex Purdue and JJ Valletta for being good people to share the office with and always willing to help. Thanks also to the other people that have always had time for a chat and made the last 3 years fun Dr Richard Boardman, Dr Giles Endicott, Dr Allan Burke-Veliz, Dr David Toal and Nicholas Reed. A special thanks to Dr Stuart Boden and Michael Pollard from the Nano Reaserach Group, ECS, Asa Asadollahbaik from Materials Research Group and Slava Kubytskyi from the Kyiv National Taras Shevchenko University for their support and knowledge.

Thanks to Dr Matthew Fairman for always being there and offering encouragement, I could not have done it without you.

Nomenclature

ϵ	The dielectric constant
Γ	The centre of the irreducible Brillouin zone
$\mathscr{L}$	The linear differential operator
μ	The magnetic permeability
∇	The vector differential operator
ϕ	RBF
Ψ	The wave function of Schrödinger's equation
$\mathbf{B}$	The magnetic field density
$\mathbf{D}$	The electric flux density
$\mathbf{E}$	The electric field intensity
$\mathbf{H}$	The magnetic field intensity
$\mathbf{R}$	The translational vector
$\mathbf{r}$	The position vector
Θ	The Hermitian linear operator
a	The lattice constant
c_s	The velocity of light
k	The quasimomentum vector
M	The face edge of the irreducible Brillouin zone
n	The refractive index
v_p	The phase velocity
X	The corner of the irreducible Brillouin zone

Chapter 1

Introduction

1.1 Photonic Crystal Overview

It can be seen from nature, that it is possible to create amazing optical effects using photonic band gap (PBG) structures e.g. the iridescence in the wings of a Morpho butterfly. Since 1987 the term “Photonic Crystals” (PhCs) [6, 7] has been used to describe periodic dielectric structures that prevent the propagation of certain wavelengths of electromagnetic radiation. Figure 1.1 shows a simple photonic crystal structure that causes photons to have frequency bands and frequency gaps, which are comparable to a semiconductor crystal lattice where electrons have energy bands and band gaps. The PhC shown is made of silicon pillars that are arranged in a square lattice and surrounded by air. Suitable PhCs could be used to filter, focus and disperse light to create many desired specialist optical effects. There is considerable interest in these crystals due to their potential application in, for example, lasers, optical circuits (computers), light sources and optical communications. Fabrication of PhCs is difficult and expensive, and therefore it is vital that an accurate method of modelling is used to ensure a suitable PBG is created by the structure. Moreover, as the complexity of the geometric configuration increases to mimic, for example structures found in nature, new modelling methods maybe required. In the context of this thesis, a computationally expensive algorithm is an algorithm that has a run time which is to long to be used in an optimisation process (less than a minute is good, less that 5 minutes is acceptable, greater than 10 mins is no good). This is because if an individual run time is to long the optimisation will not be able to do a large amount of runs and therefore will not cover the whole design space.

1.1.1 The Wigner-Seitz Cell

PhCs have discrete translational symmetry as the structure repeats after a set distance e.g. $\mathbf{r} = \mathbf{r} + \mathbf{R}$ where $\mathbf{R}$ is the translational vector $\mathbf{R} = s_1\mathbf{a}_1 + s_2\mathbf{a}_2 + s_3\mathbf{a}_3$. Here $\mathbf{a}_{1-3}$ are

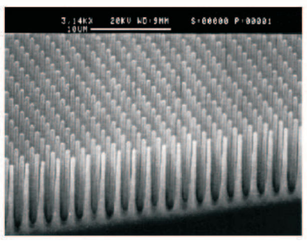


FIGURE 1.1: Single crystal silicon pillars. Courtesy of Martin Charlton, ECS, University of Southampton.

the lattice vectors and s is an integer. The smallest repeatable cell of the periodic PhC lattice is know as a Wigner-Seitz cell. The cell is constructed by extending lines between a lattice point and its nearest neighbours. The perpendicular bisectors of these lines are then taken and the area or volume enclosed by these bisectors is the Wigner-Seitz cell. Figure 1.2 shows a unit cell for a square lattice of air cylinders in a solid substrate, the reverse of the crystal in Figure 1.1.

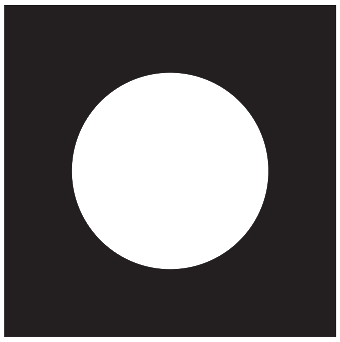


FIGURE 1.2: Unit cell for a square array of air columns drilled into a material substrate.

1.1.2 Maxwell's Equations

Modelling the behaviour of light in a PhC requires the use of Maxwell's equations. Assuming the material has no free charges or current Maxwell's equations are stated as:

$$\begin{aligned}
\nabla \cdot \mathbf{B}(\mathbf{r},t) &= 0, \\
\nabla \cdot \mathbf{D}(\mathbf{r},t) &= 0, \\
\nabla \times \mathbf{E}(\mathbf{r},t) &= -\frac{\partial}{\partial t}\mathbf{B}(\mathbf{r},t,) \\
\nabla \times \mathbf{H}(\mathbf{r},t) &= \frac{\partial}{\partial t}\mathbf{D}(\mathbf{r},t), \qquad (1.1)
\end{aligned}$$

where **B** is the magnetic field density, **E** is the electric field intensity, **D** is the electric flux density and **H** is the magnetic field intensity. The refractive index of a material is a measure of how much the speed of light is reduced when travelling through the material. The refractive index is defined by:

$$n = \frac{c_s}{v_p} \tag{1.2}$$

where c_s is the velocity of light and v_p is the phase velocity. The dielectric constant (ϵ) of a material is equal to the square of the refractive index in a non-magnetic medium.

For modelling the behaviour of electromagnetic waves in two-dimensional systems the following standard assumptions will be made [2]:

1. The fields strengths are very small.
2. The material is macroscopic, isotropic and has regions of homogenous dielectric material.
3. The dielectric constant does not vary with frequency and low-loss dielectric materials are used, meaning the imaginary part of the dielectric constant can be ignored.
4. The magnetic permeability is close to unity so the material is non-magnetic.

Then Maxwell's equations can be rewritten in terms of the magnetic field as [1]:

$$\nabla \times \left(\frac{1}{\epsilon(\mathbf{r})} \nabla \times \mathbf{H}(\mathbf{r}) \right) = \left(\frac{\omega}{c_s} \right)^2 \mathbf{H}(\mathbf{r}). \tag{1.3}$$

This can then be rewritten as an eigenvalue problem:

$$\Theta \mathbf{H}(\mathbf{r}) = \left(\frac{\omega}{c_s} \right)^2 \mathbf{H}(\mathbf{r}), \tag{1.4}$$

where

$$\Theta = \nabla \times \left[\frac{1}{\epsilon(\mathbf{r})} \nabla \times \right], \tag{1.5}$$

which is a Hermitian linear operator. The solution of equation (1.4) gives eigenvectors that correspond to the field patterns of the harmonic modes and eigenvalues which are proportional to the squared frequencies of these modes.

1.1.3 Bloch-Floquet Theorem

In 1928, Bloch showed that electrons in a conductor are not scattered by the periodic ions but only by imperfections [8]. Bloch showed that the wave function $\Psi_{\mathbf{k}}(\mathbf{r})$ of Schrödinger's equation can be written as the product of a periodic envelope function $U_{\mathbf{k}}(\mathbf{r})$ multiplied a planewave $e^{\imath \mathbf{k} \cdot \mathbf{r}}$. This work extended a one-dimensional theorem by Floquet from 1883 [9]. By taking equation (1.4) as analogous to Schrödinger's equation this technique can be applied to modelling photonic crystals. Now the magnetic field can be rewritten as a Bloch state of the form:

$$\mathbf{H}(\mathbf{r}) = e^{\imath\mathbf{k}\cdot\mathbf{r}}u_{\mathbf{k}}(\mathbf{r}), \tag{1.6}$$

where

$$u_{\mathbf{k}}(\mathbf{r}) = u_{\mathbf{k}}(\mathbf{r}+\mathbf{R}). \tag{1.7}$$

Equation (1.7) shows an important feature of Bloch states that different values of k don't necessarily give different modes. The first Brillouin zone is a Wigner-Seitz cell in the reciprocal lattice space where all $\mathbf{k}$ vectors are unique. If there is rotational symmetry in the Brillouin zone then a smaller section of the zone may be used, known as the irreducible Brillouin zone. The first Brillouin zone (shaded yellow) and irreducible Brillouin zone (shaded blue) are shown for a square lattice in Figure 1.3. Critical or special points are found at the centre $(0, 0)$, corner $(\pi/a, 0)$ and face edge $(\pi/a, \pi/a)$ and are known as Γ, M and X respectively, where a is the lattice constant.

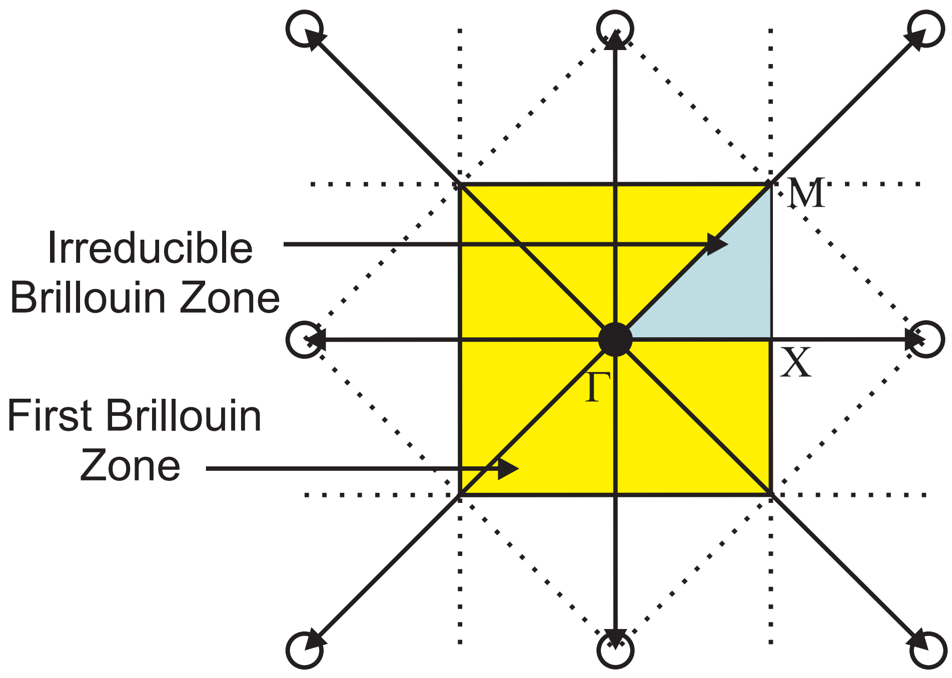


FIGURE 1.3: Brillouin zone construction for a square lattice.

1.1.4 Photonic Band Gaps

In 1887, Lord Rayleigh studied the propagation of waves through a medium endowed with a periodic structure [10]. He found that all one-dimensional PhCs have a small band gap. Figure 1.4 helps to explain his findings. The dashed lines show the dispersion relation for a material in which the dielectric is constant. Here an artificial periodicity a has been defined so that the region of interest can be limited to the irreducible Brillouin zone from 0 to π/a. As the medium is homogenous the dispersion relation is just the light-line given by:

$$\omega = \frac{c_s \mathbf{k}}{\sqrt{\epsilon}}. \tag{1.8}$$

As periodic boundary conditions have been imposed, the lines fold back into the Brillouin zone when they get to the edges. The solid lines represent the dispersion line for a one-dimensional PhC, which therefore has a real periodicity. The degeneracy of the lines only holds when the periodicity is artificial so for all one-dimensional PhCs there is a PBG. The size of the gap depends on the contrast between ϵ_1 and ϵ_2. The low order gap occurs at the edges of the brillouin zone at $\mathbf{k} = \pi/a$. The modes in this case are standing waves, but there are two possible ways to position them. The first position means the wave's peaks and troughs are in the regions of high dielectric and the second position means the wave's peaks and troughs are in the regions of low dielectric. The electromagnetic variational theorem, which is analogous to the variational method of quantum mechanics, tells us that high-frequency modes concentrate their energy in the low dielectric regions and low-frequency modes concentrate their energy in the high dielectric regions [2]. The difference in frequency between the two modes causes the PBG to appear.

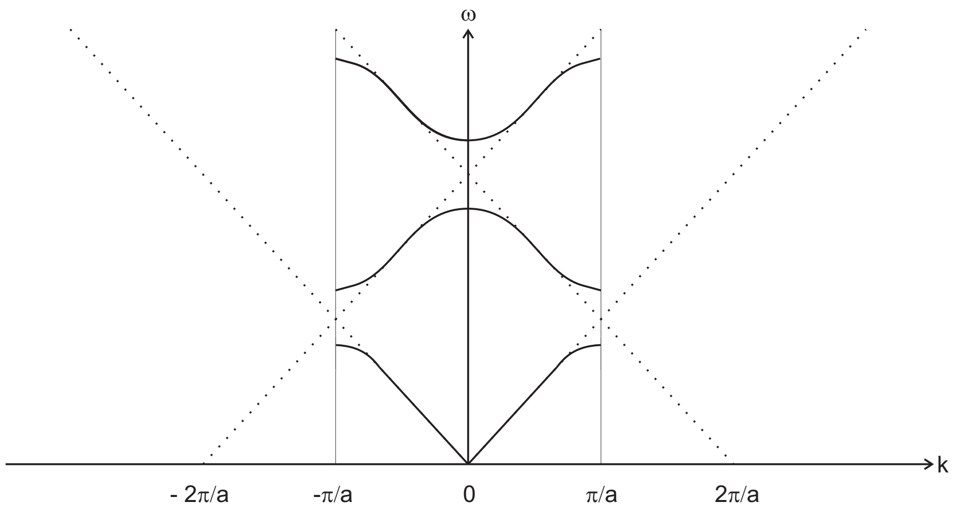


FIGURE 1.4: Dispersion relation for a one-dimensional photonic crystal (solid lines) and a uniform material (dashed lines). Adapted from [1].

The reason that there seems to be no electromagnetic modes in the gap, is that no purely real wave vectors exist. The wave vector in this case is complex $k + \imath\vartheta$ and the imaginary part causes the wave amplitude to decay exponentially into the crystal. These modes are known as evanescent modes and for light propagating in the z-direction have the form:

$$\mathbf{H}(\mathbf{r}) = e^{\imath \mathbf{k}_c \cdot z} \mathbf{u}_{\mathbf{k}c}(z) e^{-\vartheta z} \tag{1.9}$$

where $\mathbf{k}_c$ is the complex wave vector.

1.1.5 Two-Dimensional Photonic Crystals

A two-dimensional photonic crystal is periodic in the xy-plane and homogenous in the z-direction e.g cylindrical rods arranged in a square lattice surrounded by air. When looking at light propagation in only the xy-plane, mirror reflection symmetry in the z-direction allows separation into two distinct polarisations. In one, the electric field is parallel to the mirror plane and the magnetic field is perpendicular; here there is no electric field in the direction of propagation and these modes are known as transverse-electric (TE). In the other polarisation, the opposite happens and the magnetic field is parallel to the xy-plane and the electric field is perpendicular; here there is no magnetic field in the direction of propagation and these modes are known as transverse-magnetic (TM). Therefore we can solve equation (1.4) as two separate eigenvalue problems, one for each mode. The band structures for TE and TM are often totally different, so there may be a PBG in one but not in the other; or both may have a PBG but they do not overlap. In order for a structure to have a complete photonic band gap the gaps in the TE and TM modes must overlap. Band diagrams are used to illustrate the band structures of the modes. Figure 1.5 shows the band diagram for a hexagonal array of air columns drilled into a dielectric substrate ($\epsilon = 13$). The blue lines represent the TM bands, the red lines the TE bands and there is a complete PBG for this structure highlighted in yellow. The right-hand inset in the figure shows the crystal lattice layout and the left-hand inset the Brillouin zone with the irreducible brillouin zone shaded in light blue. The x-axis shows the in-plane wave vector $\mathbf{k}$, which goes along the edge of the irreducible Brillouin zone from Γ to M to K.

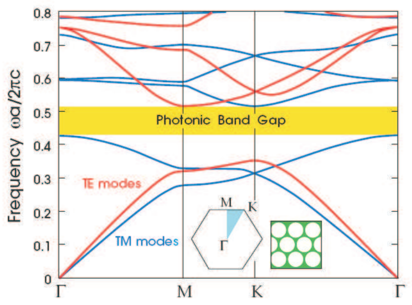


FIGURE 1.5: Reproduced from [2] the photonic band structure for the modes of a hexagonal array of air columns drilled into a dielectric substrate ($\epsilon = 13$). The blue lines represent the TM bands and the red lines the TE bands, there is a complete photonic band gap for this structure highlighted in yellow.

Gap-midgap ratio is used to measure a PBG. The absolute width of a PBG is not useful, as results in electromagnetics are scalable e.g if a PhC with a band gap of $\Delta\omega$ was expanded by factor a of s the resulting band gap would be $\Delta\omega/s$. However the gap-midgap is independent of the scale of the PhC. Where ω_0 is the frequency at the middle of the gap, the gap-midgap ratio is defined as $\Delta\omega/\omega_0$.

It is important to note that real two-dimensional PhCs (PhC Slabs) are finite in the z-direction and this leads to out-of-plane losses [11]. However they are already being used in commercial applications e.g. in LEDs for increased light extraction [12].

1.2 Focus of Thesis

The main aim of the thesis is to develop novel numerical algorithms for the efficient design of new PhC structures based on band diagram analysis.

The objectives of this thesis may be stated as follows:

1. To explore various ways to model two-dimensional PhCs by solving the generalised eigenvalue problems formed from Maxwell's equations.
2. To use the Finite Difference Method to model periodic PhCs.
3. To optimise PBGs using novel shape parameterisations.
4. To use a meshless method to solve periodic systems to validate their applicability.
5. To apply a meshless method to a two-dimensional PhC modelling problem.
6. To look into the use of novel accelerator technologies to help speed up the modelling process.

1.3 Structure of Thesis

This thesis is spilt into the following chapters:

- Chapter 1 - *Introduction.* This chapter introduces PhCs as an interesting modelling problem. It gives an overview of the history and theory of PhCs. Two-dimensional PhCs are then discussed as the main focus that will be modelled in this thesis.
- Chapter 2 - *Photonic Crystal Modelling - Finite Difference Method.* The finite difference method is introduced in this chapter as a way to model photonic crystals. The method is improved to work with crystal structures that have curved surfaces. A number of commonly modelled structures are simulated and compared with

results found in the literature produced by the plane wave expansion method. Finally, a novel structure is modelled and compared with results from a commercial software package.

- Chapter 3 - *Novel Two-Dimensional Photonic Crystals.* In this chapter a Genetic Algorithm is used as an optimisation technique to find larger gap-midgap ratios for novel PhC structures. Two different techniques are used for the parameterisation of the shape in the unit cell. The results are presented for the novel rods and compared with cylindrical rods.

- Chapter 4 - *Meshless Methods.* This chapter introduces a meshless method algorithm for solving partial differential equations; specifically the periodic elliptic Helmholtz equation. This well known partial differential equation was chosen to be solved because it has an analytical solution against which the implemented method may be verified. Computational experiments are carried out, to work out the optimum parameters to use in order to gain results of the required accuracy.

- Chapter 5 - *Photonic Crystal Modelling - Meshless Method.* The meshless strong form method from the previous chapter is built on to model the TM mode. The TE mode is then modelled using a more complicated meshless weak form method. The results are then compared with an analytical solution and two common results from the literature produced by the plane wave expansion method.

- Chapter 6 - *Technologies.* This chapter looks into using a range of accelerator technologies to speed up modelling algorithms. The chapter discusses different types of parallel architectures and the computer languages used to code them. It then goes into detail about how NVIDIA's and ATI's graphics hardware was used to accelerate the meshless method.

- Chapter 7 - *Conclusions and Further Work.* This chapter summarises the findings and provides an outlook for future research.

1.4 Contributions

The work in this thesis has been published:

- E. E. Hart, S. J. Cox, K. Djidjeli, and V. O. Kubytskyi. Solving an eigenvalue problem with a periodic domain using radial basis functions. *Engineering Analysis with Boundary Elements*, 33(2):258-262, 2009.

The following publication is under review:

- E. E. Hart, S. J. Cox and K. Djidjeli. Compact radial basis function meshless methods for photonic crystal modelling.

Chapter 2

Photonic Crystal Modelling - Finite Difference Method

2.1 Introduction

Combining the fundamental laws of electromagnetism and mathematics it is possible to create methods of numerical analysis for modelling PhCs. The traditional method used for modelling is the plane wave expansion method (PWEM) [13, 14, 15] as it is simple to implement and provides good results. However the method creates large dense matrices, is slow to converge, does not scale well to complex systems and is thus computationally expensive. The finite element method (FEM) has been used to model novel PhCs [16], which relies on elements that are connected by nodes in a predefined way. A finite difference discretisation method is suggested as an alternative. The finite difference method (FDM) is simple to implement, whilst taking into account the sharp discontinuities in the dielectric constant. The method presented in this chapter furthers the work done by [17] as it is suitable for curved surfaces within the unit cell e.g. circular rods. The method also creates sparse matrices that will require less computation and memory.

This chapter discusses the FDM to model PhCs. The work is compared to the standard results produced by the PWEM. This chapter is structured as follows: section 2.2 presents the formulation of the FDM for PhCs. The band diagrams produced by the FDM are show in section 2.3 before conclusions are drawn in section 2.4.

2.2 Finite Difference Method

The governing equations for the FDM are obtained by rearranging Maxwell's Equations (1.1). The TM and TE modes are respectively:

$$\begin{aligned}\nabla \cdot \left(\frac{1}{\mu}\nabla E\right) + \lambda\epsilon E &= 0,\\ \nabla \cdot \left(\frac{1}{\epsilon}\nabla H\right) + \lambda\mu H &= 0,\end{aligned} \tag{2.1}$$

where μ is the magnetic permeability, which for the derivation is deliberately not set to 1. When the unit cell is a square, the domain is:

$$\Omega = \{(x, y) \cdot 0 \leq x, y, \leq 1\}. \tag{2.2}$$

Using the method of Yang [17], which is based on the earlier work done by Bierwirth *et al.* [18], the finite differences for the problem on a square can be derived. The work of Varga [19] has also strongly influenced the derivation. Bierwirth et al, considers a graded finite difference mesh with spacings north (n), south (s), east (e) and west (w), and four material regions characterised by μ_k, ϵ_k, $k = 1, ..., 4$. An inner rectangle $ABCD$ is considered that lies midway in the mesh. Figure 2.1 shows part of a finite difference grid labelled with the notation of Bierwirth et al.

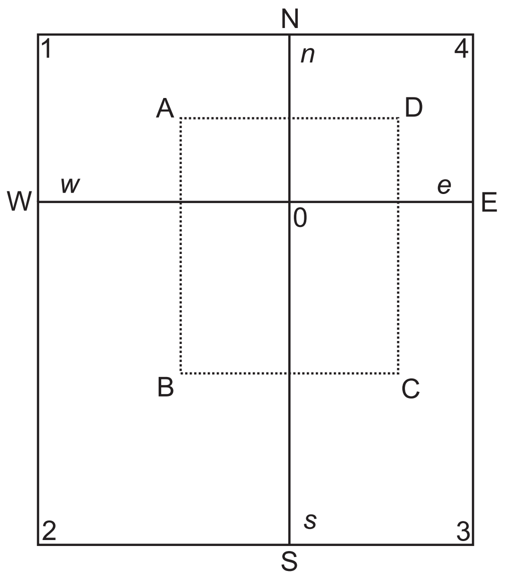


FIGURE 2.1: Part of a finite difference grid.

Consider the TE equation (2.1) and approximate the integral:

$$\int_{ABCD} \left[\nabla \cdot \left(\frac{1}{\epsilon} \nabla H \right) + \lambda \mu H \right] d\Omega. \tag{2.3}$$

Using Green's theorem to convert the first term into a surface integral gives:

$$\oint_{ABCD} \left[\frac{1}{\epsilon} \nabla H \right] \cdot \mathbf{n} ds + \lambda \oint \oint_{ABCD} \mu H d\Omega = 0. \tag{2.4}$$

Approximating the gradient on the line AB gives:

$$\nabla H \cdot \mathbf{n} = \frac{\partial H}{\partial x} \approx \frac{H_W - H_0}{w}. \tag{2.5}$$

Hence

$$\int_{AB} \frac{1}{\epsilon} \nabla H \cdot \mathbf{n} ds \approx \left(\frac{H_W - H_0}{w} \right) \left(\frac{n}{2\epsilon_1} + \frac{s}{2\epsilon_2} \right) \tag{2.6}$$

and the other contribution to the contour integral in equation (2.4) are derived in a similar manner. The second integral in equation (2.4) can be approximated to:

$$\oint \oint_{ABCD} \mu H d\Omega \approx \frac{1}{4} (nw\mu_1 + ws\mu_2 + se\mu_3 + en\mu_4) H_0. \tag{2.7}$$

Hence the finite difference approximation can be written as:

$$\begin{aligned}
& \frac{1}{2w} \left(\frac{n}{\epsilon_1} + \frac{s}{\epsilon_2} \right) H_W + \frac{1}{2s} \left(\frac{w}{\epsilon_2} + \frac{e}{\epsilon_3} \right) H_S \\
+ \quad & \frac{1}{2e} \left(\frac{s}{\epsilon_3} + \frac{n}{\epsilon_4} \right) H_E + \frac{1}{2n} \left(\frac{e}{\epsilon_4} + \frac{w}{\epsilon_1} \right) H_N \\
- \quad & \frac{1}{2} \left[\frac{1}{w} \left(\frac{n}{\epsilon_1} + \frac{s}{\epsilon_2} \right) + \frac{1}{s} \left(\frac{w}{\epsilon_2} + \frac{e}{\epsilon_3} \right) + \frac{1}{e} \left(\frac{s}{\epsilon_3} + \frac{n}{\epsilon_4} \right) + \frac{1}{n} \left(\frac{e}{\epsilon_4} + \frac{w}{\epsilon_1} \right) \right] H_0 \\
+ \quad & \frac{\lambda}{4} \left(nw\mu_1 + ws\mu_2 + se\mu_3 + en\mu_4 \right) H_0 = 0.
\end{aligned} \tag{2.8}$$

If $n = w = s = e = h$ and $\epsilon_j = \epsilon$, $\mu_j = \mu$, $j = 1, 2, 3, 4$ the equation can be rearranged to form the standard five point difference approximation.

2.2.1 Treatment of the Periodicity

As discussed in chapter 1 the Bloch-Floquet theory is used to treat the periodicity and taking a rectangle as a unit cell with sides a, b the first Brillouin zone is given by:

$$B = \mathbf{k} = s_1 \begin{pmatrix} \frac{1}{a} \\ 0 \end{pmatrix} + s_2 \begin{pmatrix} 0 \\ \frac{1}{b} \end{pmatrix}, -\pi < s_1, s_2 < \pi. \tag{2.9}$$

The solutions then take the form

$$H(\mathbf{x}) = exp(\imath \mathbf{k}.\mathbf{x}) U(\mathbf{x}), \tag{2.10}$$

where $U(\mathbf{x})$ is a periodic function.

Substituting equation (2.10) into equation (2.8) gives the modified finite difference formula:

$$\begin{aligned}
& \frac{e^{-k_1 w}}{2w}\left(\frac{n}{\epsilon_1}+\frac{s}{\epsilon_2}\right) U_W + \frac{e^{-k_2 s}}{2s}\left(\frac{w}{\epsilon_2}+\frac{e}{\epsilon_3}\right) U_S \\
+ & \frac{e^{-k_1 e}}{2e}\left(\frac{s}{\epsilon_3}+\frac{n}{\epsilon_4}\right) U_E + \frac{e^{-k_2 n}}{2n}\left(\frac{e}{\epsilon_4}+\frac{w}{\epsilon_1}\right) U_N \\
- & \frac{1}{2}\left[\frac{1}{w}\left(\frac{n}{\epsilon_1}+\frac{s}{\epsilon_2}\right)+\frac{1}{s}\left(\frac{w}{\epsilon_2}+\frac{e}{\epsilon_3}\right)+\frac{1}{e}\left(\frac{s}{\epsilon_3}+\frac{n}{\epsilon_4}\right)+\frac{1}{n}\left(\frac{e}{\epsilon_4}+\frac{w}{\epsilon_1}\right)\right] U_0 \\
+ & \frac{\lambda}{4}\left(nw\mu_1 + ws\mu_2 + se\mu_3 + en\mu_4\right) U_0 = 0.
\end{aligned} \tag{2.11}$$

2.3 Results

The results from the FDM can be validated by comparison with other theoretical results in the literature. The results are compared to the PWEM inline with the literature and will be the convention used in the rest of this thesis, i.e. the major feature for validation is the appropriate band shape without sharp oscillations. The computational results in this section were carried out using MATLAB. In section 1.1.5 we discussed that the two polarisation modes needed different crystal structures to produce large PBGs, therefore structures chosen from the literature for comparison were one that exhibits a TM PBG and another with a TE PBG. The first structure considered has isolated regions of high dielectric. Figure 2.2 shows the band diagram for a square array of alumina rods with radius $= 0.2a$ in air, that was produced by the PWEM. This figure is used for comparison with the FDM. Both the TE (red lines) and the TM (blue lines) band structures are shown. The x-axis shows the in-plane wave vector k, which goes along the edge of the irreducible Brillouin zone from Γ to X to M. Note for the band diagrams produced by the FDM the x-axis goes from Γ to X to M before returning to just before the Γ point.

Figure 2.3(a) shows the band structure for TM produced by the FDM, each of the four bands are shown in a different colour. In this band diagram there is a clear PBG between modes 1 and 2. Figure 2.3(b) shows the band structure for TE produced by the FDM. This band diagram shows that there is no PBG for this structure in the TE mode. Figure 2.3(c) shows TM and TE bands on the same diagram, there is no complete PBG as this is not possible when there is no gap in one of the modes.

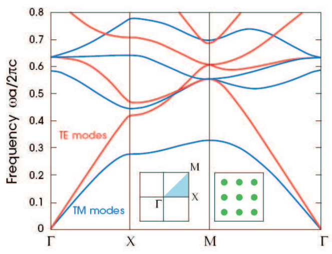
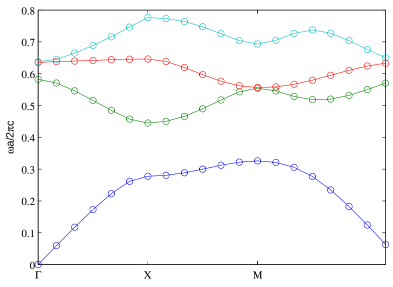
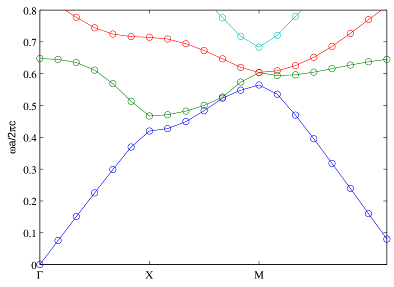


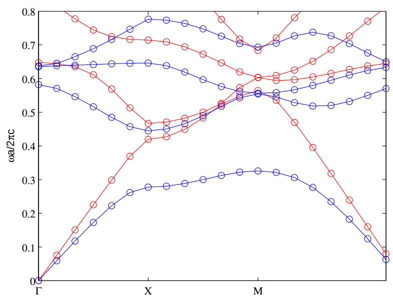
FIGURE 2.2: Reproduced from [2] a band diagram for a square array of dielectric columns ($\epsilon = 8.9$) with $r = 0.2a$ in air ($\epsilon = 1.0$). The blue bands represent the TM modes and the red bands represent the TE modes. The left inset shows the Brillouin zone, with the irreducible zone shaded light blue. The right inset shows a cross-sectional view of the dielectric.



(a) TM



(b) TE



(c) TM(blue) and TE (red)

FIGURE 2.3: Various band diagrams produced by the FDM showing the modes for a square array of dielectric columns ($\epsilon = 8.9$) with $r = 0.2a$ in air ($\epsilon = 1.0$)

The second structure to consider has a connected dielectric lattice. Figure 2.4 shows the photonic band structure for a square array of dielectric ($\epsilon = 8.9$) veins in air ($\epsilon = 1.0$). The veins have a thickness $= 0.165a$. Both the TE (red lines) and the TM (blue lines) band structures are shown. The x-axis shows the in-plane wave vector k, which goes along the edge of the irreducible Brillouin zone from Γ to X to M.

Figure 2.5(a) shows the band structure for TM produced by the FDM, each of the four bands are show in a different colour. This band diagram shows that there is no PBG for this structure in the TM mode. Figure 2.5(b) shows the band structure for TE produced by the FDM. In this band diagram there is a clear PBG between modes 1 and 2. Figure 2.5(c) shows TM and TE bands on the same diagram; there is no complete PBG.

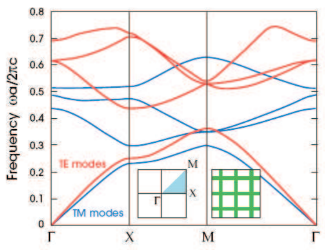
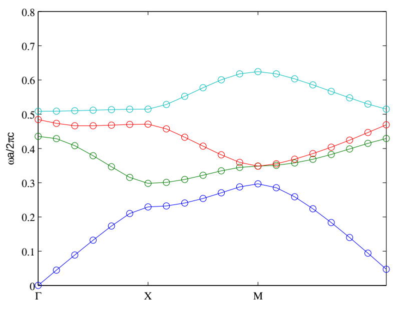
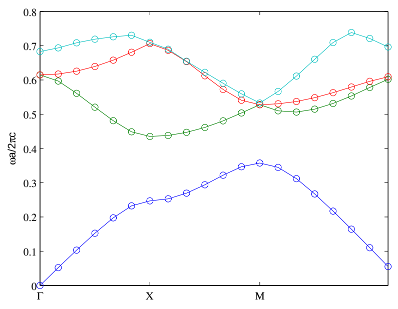


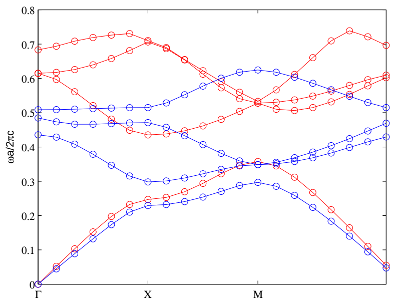
FIGURE 2.4: Reproduced from [2] a band diagram for the lowest frequency modes of a square array of dielectric ($\epsilon = 8.9$) veins (thickness $= 0.165a$) in air ($\epsilon = 1.0$). The blue bands represent the TM modes and the red bands represent the TE modes. The left inset shows the Brillouin zone, with the irreducible zone shaded light blue. The right inset shows a cross-sectional view of the dielectric.



(a) TM



(b) TE



(c) TM(blue) and TE (red)

FIGURE 2.5: Various band diagrams produced by the FDM showing the modes for a square array of dielectric ($\epsilon = 8.9$) veins in air ($\epsilon = 1.0$). The veins have a thickness $= 0.165a$.

2.3.1 Novel Structures

In 2003, Gielis proposed the Superformula as a single equation that can be used to describe a large variety of shapes [20]. In polar coordinates with r_s as the radius and ϕ_s as the angle, the Superformula can be stated as:

$$r_s(\phi_s) = \frac{1}{\{[(|\frac{1}{a_s}\cos(\frac{\phi m_s}{4})|)^{n_{s2}} + (|\frac{1}{b_s}\sin(\frac{\phi m_s}{4})|)^{n_{s3}}]^{\frac{1}{n_{s1}}}\}}. \tag{2.12}$$

Equation (2.12) has 6 parameters. The variable m_s defines the amount of rotational symmetry. The values of n_{s2} and n_{s3} determine whether the shape is inscribed or circumscribed in the unit circle. The value of n_{s1} further determines the shape, corners can be sharpened or flattened and the sides can be straight, convex or concave. a_s and b_s are the maximum height and width of the shape. Figure 2.6 shows a shape generated by the superformula using $a_s = 1.0$, $b_s = 1.0$, $m_s = 8$, $n_{s1} = 3$, $n_{s2} = 6$ and $n_{s3} = 6$, a scaling factor of 0.6 was also used to scale the shape to the unit cell.

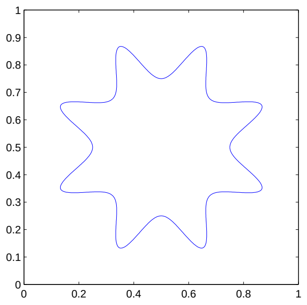
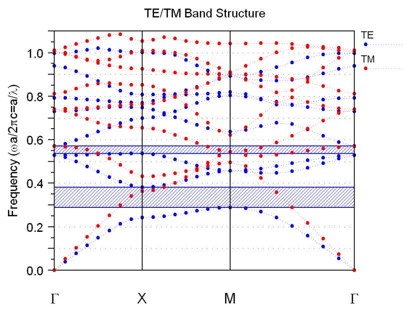
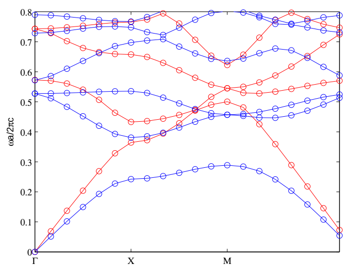


FIGURE 2.6: Cross-sectional area of a novel 'snowflake' rod. Shape generated using the superformula ($a_s = 1.0$, $b_s = 1.0$, $m_s = 8$, $n_{s1} = 3$, $n_{s2} = 6$ and $n_{s3} = 6$).

BandSolve [21] is a commercial software package released in 2002 by RSoft which uses a method based on the PWEM to model PhCs. BandSolve was used to validate the results of the FDM when a novel structure is used for the cross-sectional area of the rod. The 'snowflake' shape was chosen as a novel rod cross-sectional area as it is a concave polygon and will demonstrate the method's ability to deal with corners. Figure 2.7(a) shows a band diagram of both the TE (blue lines) and the TM (red lines) modes for a square lattice of 'snowflake' rods with $\epsilon = 8.9$ produced using 2^{18} plane waves. A large number of plane waves was chosen to give a very accurate result. It is worth noting that BandSolve names the TE and TM modes the opposite way around to the convention used in the rest of this thesis and in [2]. Figure 2.7(b) shows a band diagram of both the TE (red lines) and the TM (blue lines) modes for the same structure produced by the FDM. The two figures are in good agreement, both show the large PBG between modes 1 and 2 and the smaller PBG between modes 3 and 4 for the TM (BandSolve TE). Neither figure shows a PBG for TE (BandSolve TM).



(a) Courtesy of Michael Pollard, ECS, University of Southampton. BandSolve band diagram



(b) FDM

FIGURE 2.7: Various band diagrams showing the modes for a square array of 'snowflake' shaped dielectric columns ($\epsilon = 8.9$) in air ($\epsilon = 1$)

2.4 Conclusion

In this chapter the FDM was formulated and successfully used to model the standard results in the literature for both the TM and the TE modes. By comparing the method with a commercial software package we also know that the method can model 'novel' structures, which means it can be used to design new PhC structures using optimisation.

Chapter 3

Novel Two-Dimensional Photonic Crystals

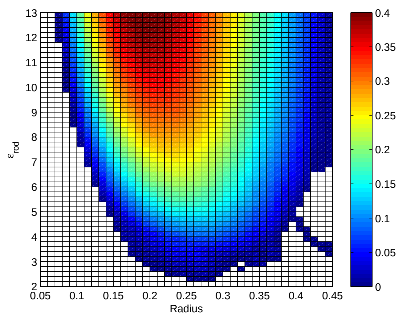
3.1 Introduction

Many studies have been carried out in order to find new two dimensional PhC structures that have larger complete PBGs. Traditionally trial and error was used [22, 23, 24, 25]. These studies looked at the affects of rod layout and cross-sectional area and concluded that high symmetry lattices and high filling factions give rise to larger complete PBGs. Padjen *et al.* [23] also concluded that rod shape could affect PBGs. They found that circular rods were better than square, rectangular or triangular in a hexagonal lattice. By using trial and error it is possible to never find the best answer so researchers moved on to optimisation. When optimising PhCs there are two main approaches taken, the first looks at a situation where only a localised search is required. This could occur when a know solution needs improving. Inversion optimisation was used in [26, 27] to design better PhC cavities. In [28, 29, 30, 31, 32] a generalised gradient ascent method was used to improve PBGs in various situations. Jensen *et al* [32] looked at improving the topology of the crystal structure in a high-bandwidth low-loss T-junction waveguide. Jao *et al.* [30] looked for a different structure to maximise each of the first ten PBGs for TM and then TE using GaAs. The biggest gap-midgap ratio they found for TM was 44.18% and for TE 21.04%. The second approach for optimisation looks at a problem where a global space needs to be searched. Genetic Algorithms were used by [33, 34, 35, 36] to design PhCs. Goh *et al.* allowed the optimisation to vary the size and location of cylindrical rods. In the other cases, the discretisation of the structure was done using a grid of small squares which could either be turned on or off to be the higher dielectric. Shen *et al.* again used GaAs and reported the largest complete PBG ratio of 20.1% for a square lattice.

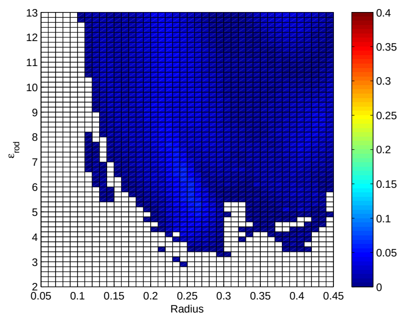
3.2 Cylindrical Rods

PhCs with cylindrical rods in air or substrates with air cylinders drilled though them are easy to manufacture, so are popular. An experiment was carried out that found the best gap-midgap ratios that could be produced by such photonic crystals, so that they could be compared with the novel rods found by the optimisation. Two computational experiments were run. In the first, the radius of a cylindrical rod was varied from $0.05 - 0.45$ and its dielectric was varied from $2 - 13$. In the second, the radius of cylindrical air hole in a substrate was varied from $0.05-0.45$ and the substrate's dielectric was varied from $2 - 13$. In both experiments the largest gap-midgap ratio was recorded for the TM mode and the TE mode. Both modes were also looked at together to see if there was a complete PBG.

Figure 3.1 shows the results from the experiment with solid rods. Figure 3.1(a) shows that for this PhC the TM mode produces a large gap-midgap ratio of 41%. As this is the largest gap produced by these experiments with cylindrical rods the colour bar from this figure is used on the other three figures for ease of comparison. Figure 3.1(b) shows that for the TE mode this PhC produces a smaller gap-midgap ratio of 6%. For the TM and the TE modes the best gap is found at the highest dielectric contrast of $13 : 1$. There is no figure shown for both modes together as this PhC structure does not produce a complete PBG, so should the figure have been shown it would have been completely white.



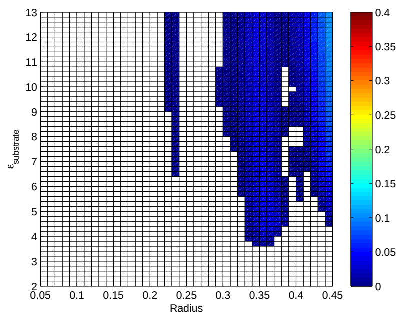
(a) TM



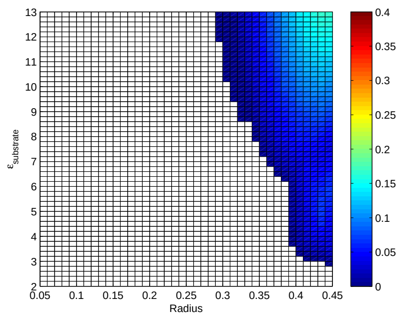
(b) TE

FIGURE 3.1: Largest gap-midgap ratio shown when the dielectric and the radius of a cylindrical rod is varied. The crystal lattice is square. White areas indicate when there is no PBG.

Figure 3.2 shows the results from the experiment with 'air' rods. Figure 3.2(a) shows that for this PhC the TM mode produces a gap-midgap ratio of 13%. Figure 3.2(b) shows that for the TE mode this PhC produces a larger gap-midgap ratio of 17%. Again for the TM and the TE modes the best gap is found at the highest dielectric contrast of $13 : 1$. This structure also has no complete PBG so the figure is not shown.



(a) TM



(b) TE

FIGURE 3.2: Largest gap-midgap ratio shown when the dielectric of the substrate and the radius of a cylindrical air rod is varied. The crystal lattice is square. White areas indicate when there is no PBG

From the results we can see that the PhCs made with circular rods do not produce complete PBGs and from the literature that more complicated structures are required to produce bigger gaps for the TE and TM modes.

3.3 Rod Parameterisation

Two new types of parameterisation were created for use in the optimisation. Motivation from nature, e.g the iridescent blue of a Morpho rhetenor butterfly's wings (figure 3.3), meant that parameterisations were chosen that could create concave polygons with sharp corners.

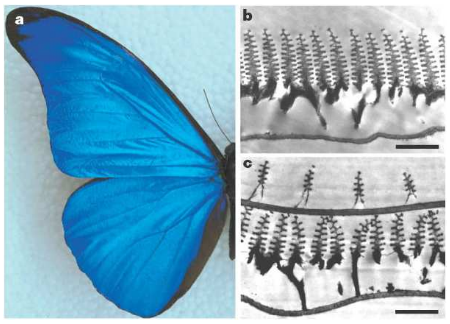


FIGURE 3.3: a, Real colour image of the blue iridescence from a M. rhetenor wing. b, Transmission electron micrograph (TEM) images showing wing-scale cross-sections of M. rhetenor. c, TEM images of a wing-scale cross-section of the related species M. didius. Bars, a, 1 cm; b, 1.8 m; c, 1.3 m. Image reproduced from [3].

3.3.1 Non-Uniform Rational B-Splines

A B-Spline is a sequence of Bezier curve segments that are connected together to form a single continuous curve. A Non-Uniform Rational B-Splines (NURBS) curve is defined by three things, its order, a set of weighted control points, and a knot vector [37]. There are two rules that must be followed when constructing a NURBS curve. Firstly the number of control points must be at least equal to the order of the curve and secondly the number of knots in the knot vector must be equal to the sum of the number of control points and the order of the curve. The order of the curve determines the number of neighboring control points whose location influences any given point of the curve. The knot vector determines how and where the control points affect the NURBS curve. There are several reasons why a NURBS curve was choose as a way to parameterise the rod cross-sectional area. Firstly, it is a well known way to represent a two-dimensional curve that does not require a large amount of parameters. Secondly, it offers the opportunity for the optimisation to be able to generate a large variety of shapes that can have very little symmetry. Thirdly, it is easy to understand how changing the value of one variable will affect the overall shape. Figure 3.4 shows two NURBS curves produced by parameterisation. The control points are shown in blue and the NURBS curves in red.

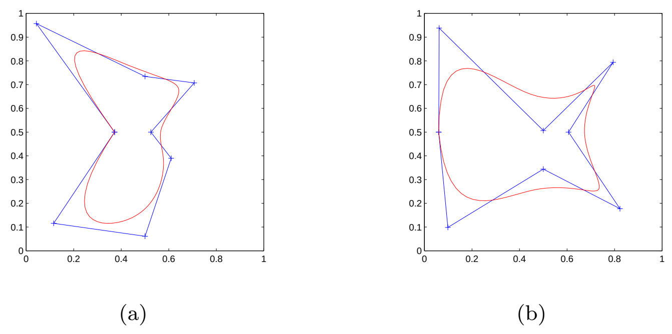


FIGURE 3.4: NURBS curves. The control points are shown in blue and the NURBS curves in red.

For the optimisation a fourth-order (cubic) curve with 8 control points that had equal weighting and a pinned uniform knot vector was chosen. Using a pinned uniform knot vector means that the first and last control points are pinned and the curve must start and end there. On its way from the first to the last control point, the curve passes close to the other control points but does not go though them. The uniform pinned knot vector used has the following format, as the order is 4^{th} the first 4 knots equal 0, then the knots increase in uniform steps until the last 4 knots which are all the same e.g. [0 0 0 0 1 2 3 4 5 6 6 6 6]. Equal weights for the control points were chosen to reduce the number of variables given to the optimisation. The eight control points were placed at each corner of the unit cell and in the middle of each side. They were then allowed to move along the line towards the centre of the unit cell. The eight control points and a scale parameter were the final variables used for the optimisation. As a comparison needed to be made between the novel rods and the cylindrical rods, it was important that the parameterisation could produce a circle. By moving the control points to the correct locations the NURBS curve can make a circle. The scale parameter was limited to 90% as this would ensure that only distinct shapes were produced.

3.3.2 Hicks-Henne Bump Functions

A Hicks-Henne bump function is a shape bumping function based on Bernstein polynomials that was first used by Hicks *et al.* in wing design optimisation [38]. A general form of the function can be written as:

$$f(x) = A_{hh} \left[sin \left(\pi x^{\frac{\log 0.5}{\log xp}} \right) \right]^t \tag{3.1}$$

where A_{hh} is the amplitude of the bump, xp locates the maximum point of the bump and t controls the width of the bump (sharp bumps are caused by large values of t) [39]. For the rod parameterisation an n sided polygon was taken and a circle drawn around it. Then n Hicks-Henne bump functions were attached to the edge of the circle. Figure 3.5 shows some rod shapes that can be created. The top two figures have 4 identical bumps, the first with a central xp and the second with an off-centre xp. The middle two figures have alternating bumps. In the first all bumps are set to be the same but in the second one set of bumps has a much smaller amplitude that the other. The bottom left figure shows that the parameterisation can produce a circle and the bottom right figure shows that it can produce n number of bumps. In this case there are 6 identical bumps but it is possible for them to all be different.

For the optimisation a family of shapes was chosen which had two sets of alternating bumps i.e. a total of 4 bumps. In total there were 7 variables xp_1, xp_2, A_{hh1}, A_{hh2}, t_1, t_2 and scale. Again the scale was limited to 90%.

3.4 Optimisation

3.4.1 Genetic Algorithms

A genetic algorithm (GA) is an optimisation technique based on Darwin's theory of evolution and the passing of genetic traits through genes. This concept was introduced by Holland [40] in 1975. Found in all cells, a chromosome is a structure made from strings of Deoxyribonucleic acid (DNA). DNA can be broken down into genes, which are used to pass genetic traits to offspring. During reproduction genes from the parents reform to create whole new chromosomes for the offspring and this is know as crossover. Errors with copying from the parents to the offspring some times occur. This results in the genes being changed slightly and is know as mutation. The fitness of an organism is measured as the succuss of its life.

In a GA an initial population is randomly created and then used to create the population in the first generation. The new population is made by evaluating the fitness of the initial population and then selecting a group parents from this. The pool of parents is selected using a tournament selection, 5 parents are randomly chosen from the last generation

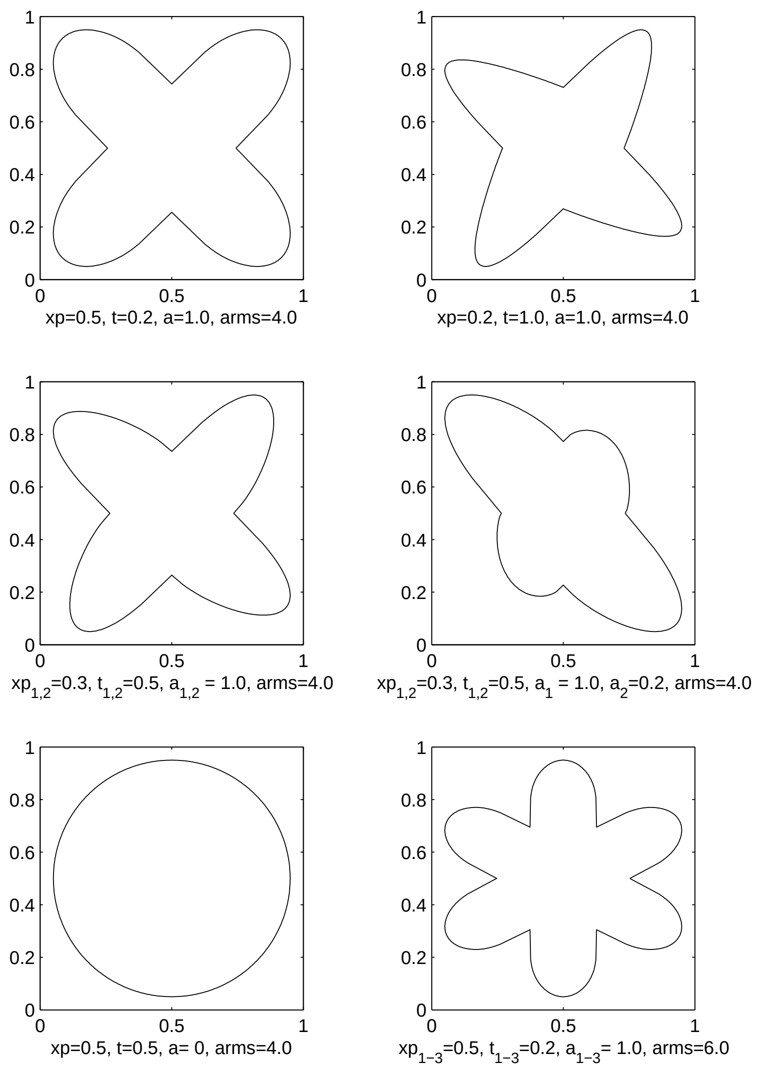


Figure 3.5: Various shapes produced by the Hicks-Henne parameterisation.

and the best one is placed into the pool. This processes is repeated until the pool is the correct size. The tournament selection ensures a wide range of parents are chosen so that the diversity of the population is kept large. Elitism is used to ensure that if it has not already been placed in the pool by the tournament selection, then the best individual must also be put in. Once the pool has been created, a set of parents is then randomly chosen to do one of three things to make the next generation; reproduce, crossover or mutate. Reproduction means both the parents are copied and then put into the new population. Crossover means part of one parent is combined with part of the other parent. Mutation means that one part of the parents is changed.

3.4.2 Experiments

A comparison of the results of the optimisation with the experiments done on the cylindrical rods needed to be made. Therefore the optimisations needed to have both high dielectric rods in air and high dielectric substrates with air holes. The high dielectric value was chosen to be 13 as this is value at which the cylindrical rods had the largest gap-midgap ratios. Table 3.1 shows the 12 different unit cell configurations that were used in the optimisations. There were 6 runs in total using the Hicks-Henne parameterisations; 3 runs for solid rods and 3 runs for air holes in a substrate. The first of the 3 runs looked at maximising the gap-midgap ratio for TM, the second for TE and the third maximising a complete gap-midgap ratio. The other 6 runs used the NURBS parameterisations but other than that, they were identical. All the runs had a large population size of 500 to allow the GA to do a good search.

Optimisation	Mode	Rods	Substrate	Shape
1	TM	$\epsilon = 1$	$\epsilon = 13$	Hicks-Henne
2	TE	$\epsilon = 1$	$\epsilon = 13$	Hicks-Henne
3	BOTH	$\epsilon = 1$	$\epsilon = 13$	Hicks-Henne
4	TM	$\epsilon = 13$	$\epsilon = 1$	Hicks-Henne
5	TE	$\epsilon = 13$	$\epsilon = 1$	Hicks-Henne
6	BOTH	$\epsilon = 13$	$\epsilon = 1$	Hicks-Henne
7	TM	$\epsilon = 1$	$\epsilon = 13$	NURBS
8	TE	$\epsilon = 1$	$\epsilon = 13$	NURBS
9	BOTH	$\epsilon = 1$	$\epsilon = 13$	NURBS
10	TM	$\epsilon = 13$	$\epsilon = 1$	NURBS
11	TE	$\epsilon = 13$	$\epsilon = 1$	NURBS
12	BOTH	$\epsilon = 13$	$\epsilon = 1$	NURBS

Table 3.1: The 12 unit cell configurations used in the Optimisation

3.5 Optimisation Results

The optimisations were given a large generation number with the idea that the outputs could be regularly reviewed to check if the GAs had converged. Figure 3.6 shows the outputs of each of the evaluations for the GA of the TE mode with a solid substrate with air hole and a NURBS parameterisation (optimisation 8). The GA has a large range of outputs which is good because it means it is sampling the whole population space. The largest gap-midgap ratio found has changed very little for the last 4000 evaluations.

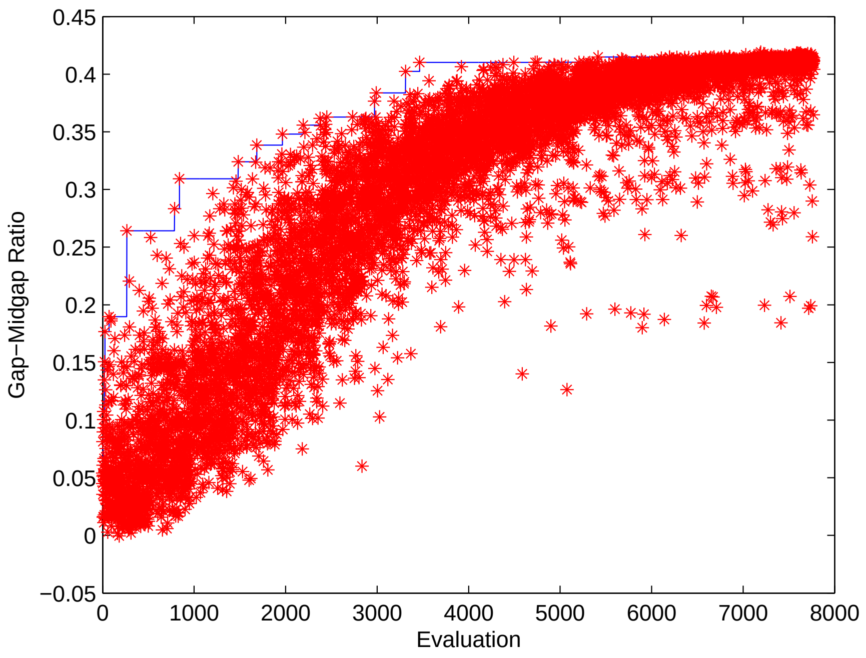


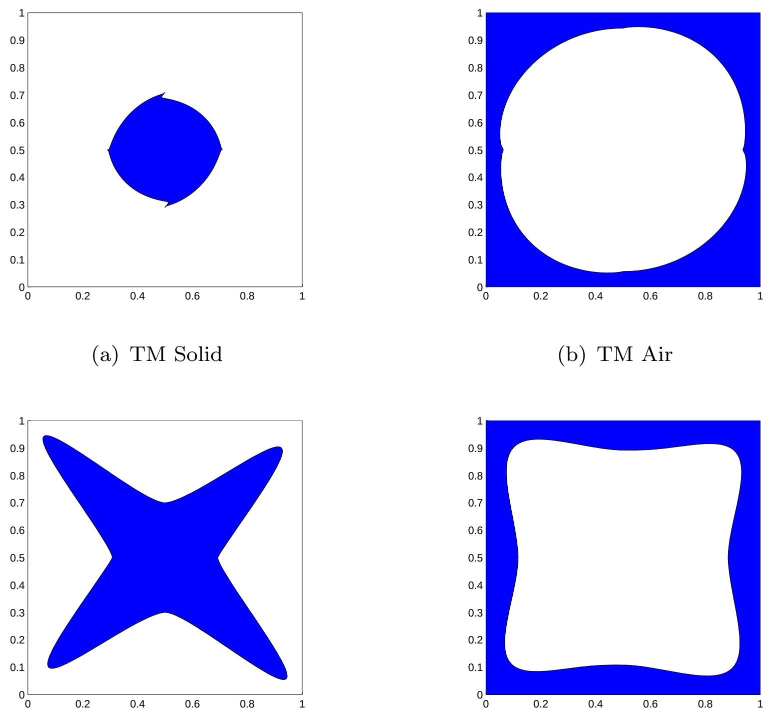
FIGURE 3.6: Output of each of the evaluations for the GA of the TE mode with a solid substrate with air hole and a NURBS parameterisation .

Table 3.2 shows the comparison between the results for the cylindrical rod experiment and the novel rods found by the GA . In all cases the GA found a larger gap-midgap ratio than the circular rods could produce. The largest gap-midgap ratios were found by the Hicks-Henne parameterisations, although the NURBS parameterisation results were close. Generally the GA found good improvements for the individual modes. The best improvement for the TE mode was with a high dielectric substrate and a Hicks-Henne shape which went from a 17% to a 42% gap-midgap ratio. The best improvement for the TM mode was for was for a high dielectric substrate with a Hick-Henne shape, which went from a 13% to a 28% gap-midgap ratio. For the TM mode with solid rods the optimisations found little improvement. For the complete gap-midgap ratios the GA only managed a small improvement of 1%.

		Gap-Midgap Ratio		
		Circle	Hicks-Henne	NURBS
Solid Rod	TE	0.06	0.21	0.21
	TM	0.41	0.42	0.42
	BOTH	0	0.01	0.01
Solid Substrate	TE	0.17	0.42	0.41
	TM	0.13	0.28	0.26
	BOTH	0	0.01	0.01

Table 3.2: Results for novel shapes compared with cylindrical rods

The best improvements were found by the Hicks-Henne parameterisations, the results for the shape of the novel rods are shown in figure 3.7. Figure 3.7(a) shows the rod cross-sectional area shape found for a solid rod for the TM mode. The result is very similar to a circle of radius 0.2, suggesting that for this mode a circle is a good rod shape. Figure 3.7(b) shows the rod cross-sectional area shape found for the air hole in a substrate for the TM mode. The result is trying to create an area of material in each of the corners of the unit cell, due to the periodicity this would then create one area of material. The GA is unable to create a completely isolated region due to the limit of the scale factor. In both cases the TM mode is behaving discussed in section 2.3 and favouring isolated regions of high dielectric material. Figure 3.7(c) shows the rod cross-sectional area shape found for a solid rod for the TE mode. The results shows the arms of the shapes trying to grow towards the corners of the unit cell. The GA is unable to grow the arms completely to the edges due to the scale factor. Figure 3.7(d) shows the rod cross-sectional area shape found for the air hole in a substrate for the TE mode. The 'air' rod has grown outwards to create a connected structure. In both case the TE mode is behaving as expected and favouring a connected lattice of high dielectric material.



(a) TM Solid

(b) TM Air

(c) TE Solid

(d) TE Air

Figure 3.7: Novel shapes produced by the optimisation

3.6 Conclusion

In this chapter a GA was used to do shape optimisation of a PhC. There were 12 runs in total which used two different shape parameterisations: NURBS and Hicks-Henne. An experiment with cylindrical rods was carried out to use as a comparison with the optimisation results. All the optimisation runs found larger gap-midgap ratios than the cylindrical rods they were compared with. The optimisation was unable to find any large complete gap-midgap ratios due to the simplicity of the parameterisations. The complexity of the parameterisation was limited by the FDM. A very complex structure would require a very large number of finite-differences to discretise it accurately and this would be computationally expensive. If each evaluation takes a long time then the GA will be very slow to converge. By introducing symmetry into the parameterisation it would be possible to reduce the time taken by each evaluation, as it would only need to sample k in the irreducible Brillouin zone rather than the whole k-space. But this would still limit the shapes that the novel structure could take. The new parametrisation technique is unique because it models whole shapes and can make concave polygons with sharp corners. Chapters 2 and 3 have helped to show the motivation for the development of a new method of modelling PhCs, that is fast and able to discretise complex shapes. In the next two chapters the development of a new method is presented.

Chapter 4

Meshless Methods

4.1 Introduction

Engineering problems often result in a set of partial differential equations (PDEs) along with a set of boundary conditions from the process of mathematical modelling. A number of numerical methods have been developed to solve these problems including the finite difference method (FDM), finite-volume method (FVM) and finite-element method (FEM). In recent years, new type of numerical methods without grids (meshless/gridless methods) have been suggested as an alternative to mesh-based methods (FDM, FVM, FEM), due to their potential in alleviating the mesh-generation complexities arising in the traditional methods such as, FDM, FVM and FEM [41]. Meshless methods are those in which the problem is represented by a discrete number of nodes/points (without any specified connections between points), and no grid or mesh is required for the simulation. In 2002 G. R. Liu defined a meshless method as: "A meshless method is a method used to establish system algebraic equations for the whole problem domain without the use of a predefined mesh for the whole domain". The goal of meshfree methods is to facilitate the simulation of increasingly demanding problems that require the ability to treat complex geometry, large deformations, and discontinuities. The emergence of meshless methods in engineering is in its early stages, but its suitability for a variety of problems (particularly fluid/structural problems) has been demonstrated [42, 43, 44].

For many years, radial basis functions [45, 46, 47] have been synonymous with scattered data approximation, especially in higher dimensions. However, in recent years there has been an increased interest in their use for solving PDEs. This approach, which approximates the whole solution of the PDEs directly using radial basis functions, is very attractive due to the fact that this method is a truly mesh or grid free technique. In 1990 Kansa [48, 49] introduced the radial basis function (RBF) collocation method for solving elliptical, hyperbolic and parabolic PDEs. This approach (Kansas method) was extended to solve various PDEs including nonlinear PDEs (see, e.g, Hon [50], Fedoseyer

[51] and Chinchapatnam [52]). In 1999 Rippa [53] carried out work on selecting the correct shape parameter for the RBFs. In 2003 Platte and Driscoll [54] showed how the global RBF collocation method could be adapted to compute eigenmodes of elliptic operators. Platte and Driscoll paid particular attention to the boundary regions and imposed Dirichlet and Neumann boundary conditions. Hardys multiquadratics (MQ and IMQ), Duchons Thin Plate Splines (TPS) and Gaussians are the Globally Supported (GS) RBFs which are commonly used in the literature, for solving PDEs [55]. MQ, IMQ and Gaussian RBFs include a shape parameter, whose numerical value can be varied to control the domain of influence of the basis function. For example, in the case of the Gaussian RBF, increasing the value of the shape parameter leads to flatter basis functions. Another class of RBF was introduced by Wendland [56], Wu [57] and Buhmann [58]. These functions are Compactly Supported (CS), i.e. the domain of influence extends over a finite region of the domain as opposed to the global RBFs whose influence extends over the entire domain. GSRBFs produce a dense collocation matrix $\boldsymbol{A}$, which tends to become ill-conditioned as the number of collocation points increases. The CSRBF kernels contain a support size parameter by which we can adjust the sparsity of the matrix, thus making $\boldsymbol{A}$ well-conditioned [59].

This chapter discusses a meshless method to solve an eigenvalue problem with periodic boundary conditions, arising from the Maxwell's equations. Using CSRBFs and restricting the shape parameter to a value less than or equal to half the length of the domain ensures the periodic boundary conditions are satisfied. The schemes are similar to generalised finite differences but with the advantage of arbitrary point locations. Due to the radial nature of the basis functions used, the meshless method also makes no distinction regarding the dimension of the problem. In this work, numerical results are presented for several RBFs and compared to the analytical solutions for the problem. The chapter is structured as follows: section 4.2 explains the classification of meshless methods used in this thesis. Section 4.3 introduces RBFs, section 4.4 presents the formulation of the meshless method and section 4.5 derives the analytical solutions to the problem. The results of the computational experiments are shown in section 4.6 before conclusions are drawn in section 4.7.

4.2 Classification of Meshless Methods

Using the formulation procedures meshless methods can be placed into three groups: weak-form, strong-form or collocation and a combination of weak-form and strong-form [43]. Meshless weak-form methods (MWFMs) take the governing PDEs with derivative boundary conditions and change them into a set of weak-form integral equations. This is often done using a variational or weighted residual method. A set of background cells is then constructed within the domain of the problem. A set of system equations can then be derived from the weak-form equations and used to integrate over the background cells. In 1992 Nayroles et al. [60] published an important paper on creating the diffuse element method (DEM) by applying the moving least squares (MLS) [61] method to the Galerkin weak form.

MWFMs can be further divided into global methods that span the whole domain and local methods that span a sub-domain. The element free Galerkin (EFG) method is a meshless global weak-form method (MGWFM) that was introduced in 1994 by Belyschko *et al* [62]. MGWFMs use the global Galerkin weak-form and the meshless shape functions. Other MGWFM include the radial point interpolation method (RPIM) [63] and the reproducing kernel particle method (RKPM) [64]. The meshless local Petrov-Galerkin method is a meshless local weak-form method (MLWFM) that was developed by Atluri *et al* [65]. Another MLWFM is the local radial point interpolation method (LRPIM) [63].

Meshless strong-form methods (MSFMs) use a collocation procedure to turn the strong-form governing equations and boundary condition equations into a set of discretised system equations. Some MSFMs include the general finite difference method (GFDM) [66] and the meshless collocation method [48].

Meshless methods that combine weak-form and strong form (MWS) were developed by Liu and Gu [67]. In MWS methods nodes near/on the boundaries with derivative conditions use local-weak form equations and all other nodes use the strong-form equations. This method is good because it uses the fewest background cells for the integration.

4.3 Radial Basis Functions Overview

4.3.1 Radial Basis Functions

A continuous function $\phi : \mathbb{R}^d \to \mathbb{R}$ is called radial basis function if $\phi(x) = \phi(y)$ whenever $||x|| = ||y||$ where $||.||$ donates the Euclidean norm and $\mathbb{R}^d$ denotes the d-dimensional space on $\mathbb{R}$ and $x, y \in \mathbb{R}$. A continuous function $\phi : \mathbb{R}^+ \to \mathbb{R}$ is strictly positive definite of order m if for every set of distinct data point $x_1, \ldots, x_N \subset \mathbb{R}^d$:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \phi(||\mathbf{x}_i - \mathbf{x}_j||) > 0, \tag{4.1}$$

for all $\alpha_1, \ldots, \alpha_N$ satisfying

$$\sum_{i=1}^{N} \alpha_i p(x_i) = 0, \tag{4.2}$$

for all polynomials p of degree less than m.

Table 4.1 lists some of the commonly used GSRBFs in the literature, $r = \|.\|$ denotes the Euclidean norm and c is the shape parameter. The domain of GSRBFs extends from $-\infty$ to ∞.

$\phi(r) = e^{-\frac{r^2}{c}}$	Gaussian
$\phi(r) = (c^2 + r^2)^{1/2}$	Multiquadratic (MQ)
$\phi(r) = (c^2 + r^2)^{-1/2}$	Inverse multiquadratic (IMQ)
$\phi(r) = r^{2c}\log(r)$	Thin Plate Spline (TPS)

TABLE 4.1: Globally Supported Radial Basis Functions.

Figure 4.1 shows a number of GSRBFs. In can been seen from the first row of the figure (4.1(a), 4.1(b), 4.1(c)) that for the Gaussian RBFs increasing value of the shape parameter leads to flatter basis functions. Rows 2 (4.1(d), 4.1(e), 4.1(f)) and 3 (4.1(g), 4.1(h), 4.1(i)) in the figure shows similar trends for the MQ and IMQ RBFs. The last row (4.1(j), 4.1(k), 4.1(l)) of the figure shows the TPS RBFs .

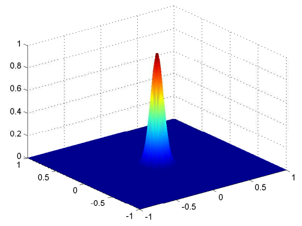
The existence of positive definite CSRBFs has been established by Wendland [56] and Wu [57]. The central idea of CSRBFs is to use a polynomial as a function of r with a local support c. The basic definition of the CSRBF $\phi_{l,\kappa}(r)$ have the form:

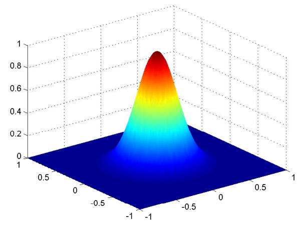
$$\phi_{l,\kappa}(r) = (1-r)_+^n p(r), \text{ for } \kappa \geq 1 \tag{4.3}$$

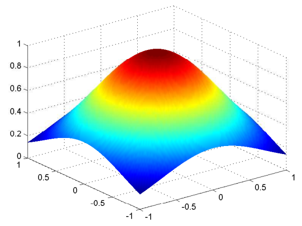
with the following conditions:

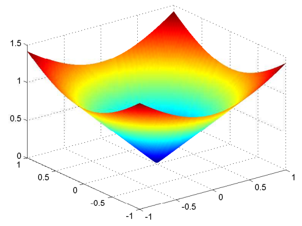
$$(1-r)_+^n = \begin{cases} (1-r)^n & \text{if } 0 \leq r < 1 \\ 0 & \text{if } r \geq 1, \end{cases}$$

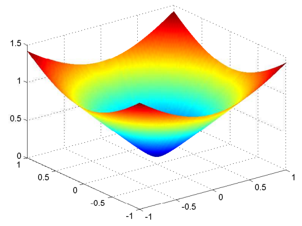
where $l = [\frac{d}{2}] + \kappa + 1$ is a dimension number, 2κ is the smoothness of the function and $p(r)$ is a prescribed polynomial. A function is said to have smoothness C^m if all its derivatives up to order m are continuous functions. Unlike GSRBFs, the influence of CSRBFs is local in $[0, 1]$ and the influence vanishes on $[0, \infty)$. Table 4.2 lists some of the Wendland and Wu CSRBFs (generally used in the literature) for $r < c$. Figure 4.2 shows Wendland and Wu's C2 and C4 functions when the shape parameter $c = 0.5$. It also shows their derivatives with respect to x and y and their their second derivatives $(\frac{\partial^2\phi(r)}{\partial x^2} + \frac{\partial^2\phi(r)}{\partial y^2})$.

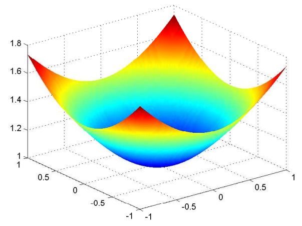

(a) GAU: c=0.01

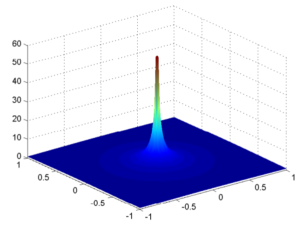

(b) GAU: c=0.1

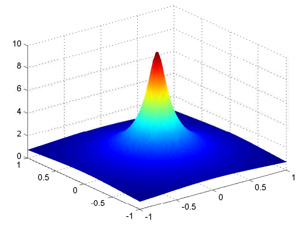

(c) GAU: c=1.0

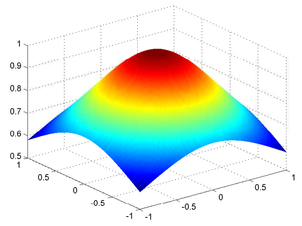

(d) MQ: c=0.01

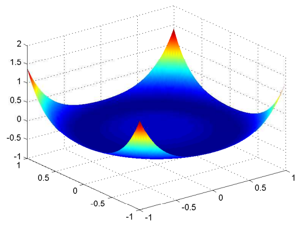

(e) MQ: c=0.1

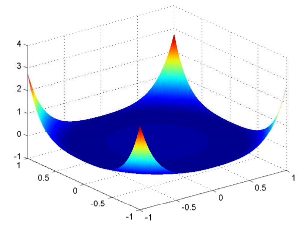

(f) MQ: c=1.0


(g) IMQ: c=0.01


(h) IMQ: c=0.1


(i) IMQ: c=1.0


(j) TPS: c=2


(k) TPS: c=3

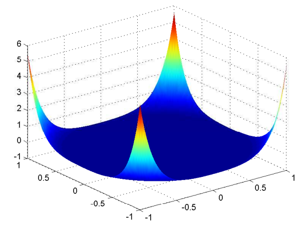
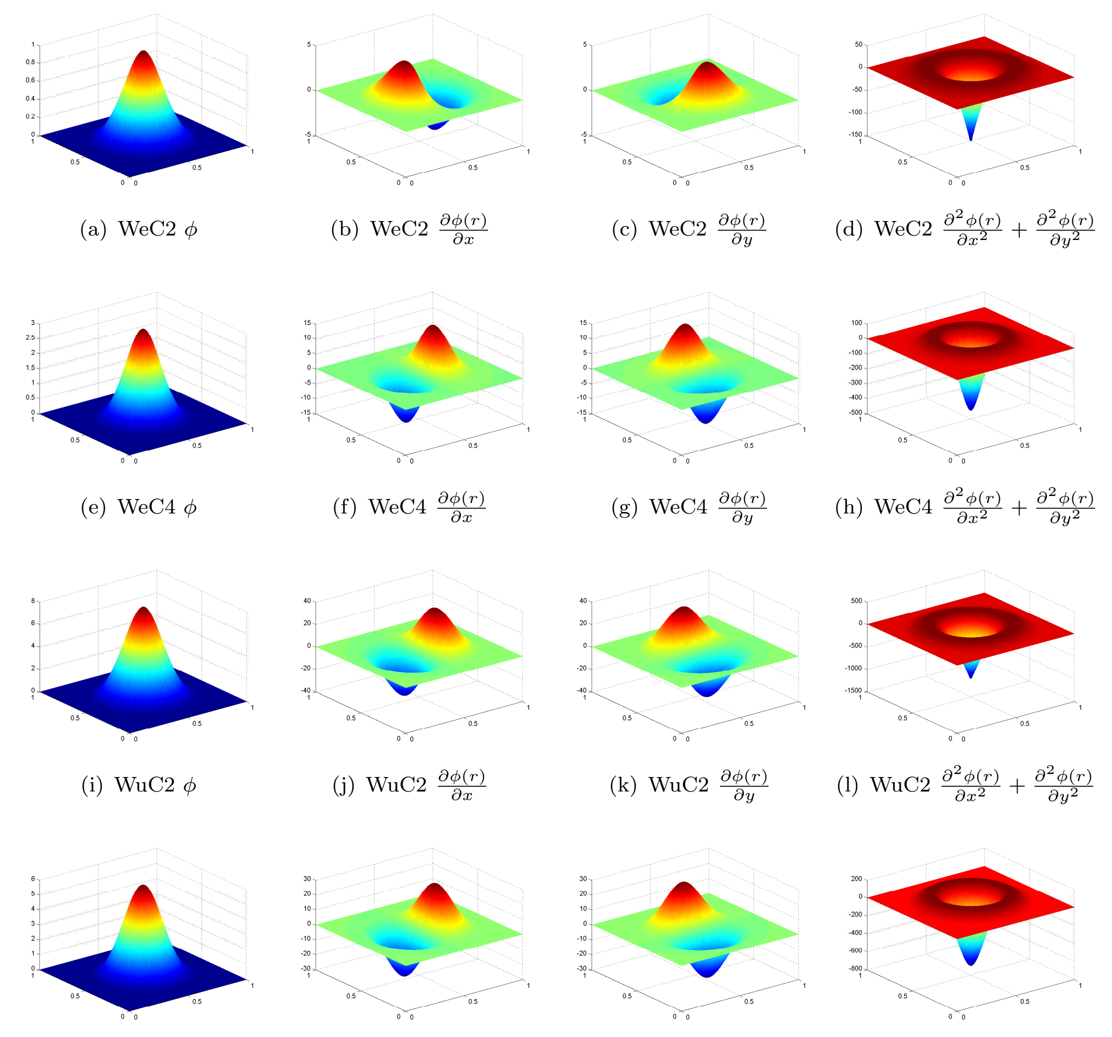

(l) TPS: c=4

FIGURE 4.1: Globally Supported Radial Basis Functions



(a) WeC2 ϕ (b) WeC2 $\frac{\partial \phi(r)}{\partial x}$ (c) WeC2 $\frac{\partial \phi(r)}{\partial y}$ (d) WeC2 $\frac{\partial^2 \phi(r)}{\partial x^2} + \frac{\partial^2 \phi(r)}{\partial y^2}$

(e) WeC4 ϕ (f) WeC4 $\frac{\partial \phi(r)}{\partial x}$ (g) WeC4 $\frac{\partial \phi(r)}{\partial y}$ (h) WeC4 $\frac{\partial^2 \phi(r)}{\partial x^2} + \frac{\partial^2 \phi(r)}{\partial y^2}$

(i) WuC2 ϕ (j) WuC2 $\frac{\partial \phi(r)}{\partial x}$ (k) WuC2 $\frac{\partial \phi(r)}{\partial y}$ (l) WuC2 $\frac{\partial^2 \phi(r)}{\partial x^2} + \frac{\partial^2 \phi(r)}{\partial y^2}$

(m) WuC4 ϕ (n) WuC4 $\frac{\partial \phi(r)}{\partial x}$ (o) WuC4 $\frac{\partial \phi(r)}{\partial y}$ (p) WuC4 $\frac{\partial^2 \phi(r)}{\partial x^2} + \frac{\partial^2 \phi(r)}{\partial y^2}$

FIGURE 4.2: Compactly Supported RBFs and their derivatives with $c = 0.5$

4.3.2 Radial Basis Function Interpolation

For scattered data $(\mathbf{x}_i, f_i) \in \mathbb{R}^{d+1}$, $1 \leqslant i \leqslant N$, the approximation $F(\mathbf{x})$ to a function $f(\mathbf{x})$ can be written as:

$$F(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi\left(\|\mathbf{x} - \mathbf{x}_j\|\right), \tag{4.4}$$

where $\|\mathbf{x} - \mathbf{x}_j\|$ is the Euclidean distance between points $\mathbf{x}$ and $\mathbf{x}_j$, N is the total number of points, $\phi(\|.\|)$ is a RBF with centre $\mathbf{x}_j$ and $\mathbf{x}$ is a point in $\mathbb{R}^d$. The unknown coefficients α_j, $j = 1, 2, ..., N$ can be determined by setting $F(\mathbf{x}_i) = f_i$, $i = 1, 2, ..., N$.

This gives the system of linear equations:

$$\mathbf{A}\boldsymbol{\alpha} = \mathbf{F}, \tag{4.5}$$

where $\boldsymbol{A} = [\phi(\|\mathbf{x}_i - \mathbf{x}_j\|)]$ is an $N \times N$ matrix, $\boldsymbol{\alpha} = [\alpha_j]$ and $\mathbf{F} = [F(\mathbf{x}_i)]$ are $N \times 1$ matrices. If ϕ is positive definite, the matrix $\boldsymbol{A}$ is non-singular, and thus the above linear system has a unique solution that can be solved, for example, by Gaussian elimination or an iterative (pre-conditioned) method. Although this condition guarantees the uniqueness of some particular RBFs interpolants if ϕ is positive definite, the RBFs generally can satisfy the condition of non-singularity.

4.3.3 Radial Basis Function Collocation

Powell [68] showed that the non-singularity of the RBFs approximations can be achieved by adding a finite number of polynomials into the system (4.4). Extending this to a general steady-state problem $\mathscr{L}[u] = f$, where $\mathscr{L}$ is some arbitrary linear differential operator (in our case $\mathscr{L} \equiv \nabla^2 + \lambda^2$), then an approximation $\tilde{u}$ to the solution u of $\mathscr{L}[u] = f$ can be obtained by letting:

$$\tilde{u}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi(\|\mathbf{x} - \mathbf{x}_j\|), \tag{4.6}$$

where α_j are obtained by letting

$$\mathscr{L}[\tilde{u}](\mathbf{x}_i) = f_i, \quad 1 \leq i \leq N. \tag{4.7}$$

If the matrix $\boldsymbol{A}_{\mathscr{L}} = [\mathscr{L}\phi(\|\mathbf{x}_i - \mathbf{x}_j\|)]$ is non-singular, then the linear system (4.7) has a unique solution and $[\alpha_j]$ can be obtained by Gaussian elimination or a iterative method. In general the function $\mathscr{L}\phi(r)$ is not positive definite, even if $\phi(r)$ is, and the theoretical proof of the solvability of (4.7) is still an open question. However, numerous numerical studies have shown that in many cases the matrix $\boldsymbol{A}_{\mathscr{L}}$ is invertible and that $\tilde{u}$ can provide an accurate approximation to u for sufficiently large N.

4.4 Meshless Method Formulation

By assuming the magnetic permeability μ and the dielectric constant ϵ to be 1, the TM and TE modes (2.1) can be cast into the form:

$$\nabla^2 u + \gamma^2 u = 0, \tag{4.8}$$

where ∇^2 is the Laplace operator, γ is a constant and the unknown function u (E or H) is defined on n-dimensional Euclidean space $\mathbb{R}^n$ (typically $n = 1, 2,$ or 3, when the solution to this equation makes physical sense). Equation (4.8) is commonly known as the elliptic Helmholtz equation, and arises in many physical applications, in particular in acoustic and electromagnetic waves.

In two dimensions, the scalar Helmholtz equation (4.8) takes the form:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \gamma^2 u = 0. \tag{4.9}$$

Substituting equation (4.6) into (4.9), leads to:

$$\sum_{j=1}^{N} \alpha_j \left(\frac{\partial^2 \phi(\|\mathbf{x}-\mathbf{x}_j\|)}{\partial x^2} + \frac{\partial^2 \phi(\|\mathbf{x}-\mathbf{x}_j\|)}{\partial y^2} \right) =$$

$$-\gamma^2 \sum_{j=1}^{N} \alpha_j \phi(\|\mathbf{x}-\mathbf{x}_j\|) \ , \quad i = 1, \ldots, N. \tag{4.10}$$

Equation (4.10) can be written in matrix form as:

$$L\boldsymbol{\alpha} = -\gamma^2 G\boldsymbol{\alpha}, \tag{4.11}$$

where,

$$G = \begin{bmatrix} g_1(\mathbf{x}_1) & \cdots & g_N(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ g_1(\mathbf{x}_N) & \cdots & g_N(\mathbf{x}_N) \end{bmatrix} \tag{4.12}$$

and

$$L = \begin{bmatrix} l_1(\mathbf{x}_1) & \cdots & l_N(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ l_1(\mathbf{x}_N) & \cdots & l_N(\mathbf{x}_N) \end{bmatrix}, \tag{4.13}$$

with

$$g_j(\mathbf{x}) \equiv \phi\left(\|\mathbf{x}-\mathbf{x}_j\|\right), l_j(\mathbf{x}) \equiv \nabla^2 \phi(\|\mathbf{x}-\mathbf{x}_j\|). \tag{4.14}$$

Equation (4.11) is a generalised eigenvalue problem. The expansion coefficients α are found by solving equation (4.11) (for the interior points of the domain) and by enforcin) the periodic equations (4.15) and (4.16) at the boundary points. The domain of the system can be seen in Figure 4.3.

Assuming periodic boundary conditions, the domain in figure 4.3 is made periodic by imposing the conditions:

$$u\,(x,0) = u\,(x,b) \tag{4.15}$$

and

$$u\,(0,y) = u\,(a,y)\,, \tag{4.16}$$

where a and b are the length of the edges of the domain. This was calculated in the method as:

$$\Delta_x = min(|x - x_i|\,, a - |x - x_i|) \tag{4.17}$$

and

$$\Delta_y = min(|y - y_i|\,, b - |y - y_i|) \tag{4.18}$$

where Δ_x and Δ_y are the difference in distances between the points in the x and y direction respectively and min selects the smallest number from the two answers calculated to give the minimum distance, taking into account the periodic boundary conditions. Combining equations (4.17) and (4.18) the Euclidean distance was then calculated as:

$$r = \sqrt{\Delta_x^2 + \Delta_y^2}. \tag{4.19}$$

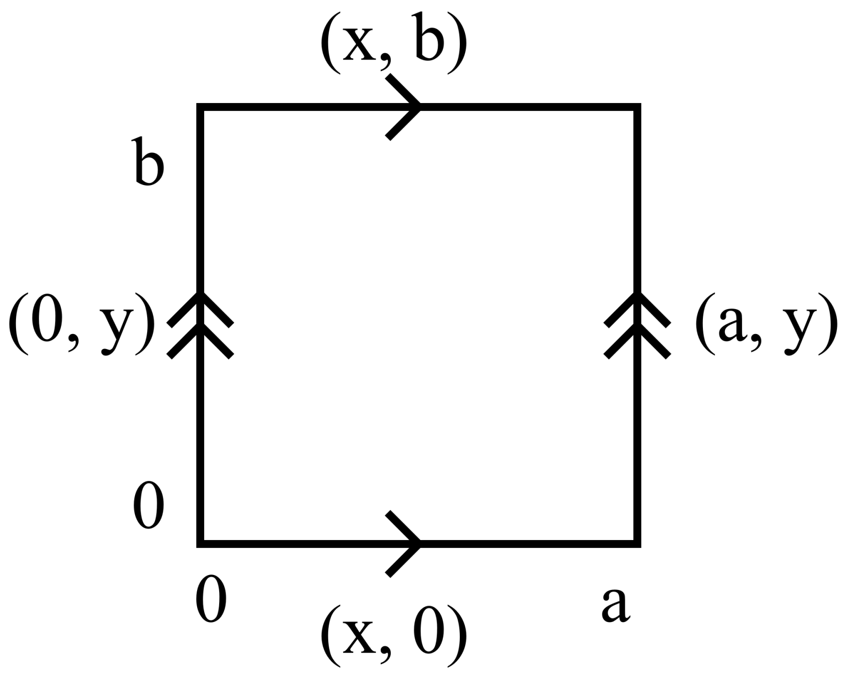


FIGURE 4.3: Domain of the periodic system.

This meshless method uses only CSRBFs as they are continuous at the boundaries of the domain (along with $\nabla^2\phi$) if the correct shape parameter is selected. For a square domain, the shape parameter must be less than or equal to half the value of the length of the domain to ensure that the periodic boundary conditions are satisfied.

The four CSRBFs (together with the corresponding $\nabla^2\phi$) used in the validation of the meshless method (for $r < c$) are shown in Table 4.2.

Wendland-C2
$\phi(r) = (1-\frac{r}{c})^4(1+4\frac{r}{c})$ $\nabla^2\phi = -\frac{20}{c^5}(c-r)^2(2c-5r)$
Wendland-C4
$\phi(r) = (1-\frac{r}{c})^6(3+18\frac{r}{c}+35\frac{r^2}{c^2})$ $\nabla^2\phi = -\frac{112}{c^8}(c-r)^4(c^2+4cr+20r^2)$
Wu-C2
$\phi(r) = (1-\frac{r}{c})^5(8+40\frac{r}{c}+48\frac{r^2}{c^2}+25\frac{r^3}{c^3}+5\frac{r^4}{c^4})$ $\nabla^2\phi = \frac{9}{c^9}(c-r)^3(45r^4+135r^3c+123r^2c^2+9rc^3-32c^4)$
Wu-C4
$\phi(r) = (1-\frac{r}{c})^6(6+36\frac{r}{c}+82\frac{r^2}{c^2}+72\frac{r^3}{c^3}+30\frac{r^4}{c^4}+5\frac{r^5}{c^5})$ $\nabla^2\phi = \frac{11}{c^{11}}(c-r)^4(55r^5+220r^4c+307r^3c^2+128r^2c^3-64rc^4-16c^5)$

TABLE 4.2: Wendland and Wu CSRBFs (with $\nabla^2\phi \equiv \frac{\partial^2\phi(r)}{\partial x^2} + \frac{\partial^2\phi(r)}{\partial y^2}$).

4.5 Analytical Solutions

The analytical solutions of the elliptic Helmholtz equation (4.8) with periodic boundary conditions can be found by imposing the following conditions:

$$u(x,0) = u(x,b),\ u(0,y) = u(a,y) \tag{4.20}$$

and

$$u'(x,0) = u'(x,b),\ u'(0,y) = u'(a,y), \tag{4.21}$$

where a and b are the lengths of the x and y domain boundaries respectively. Using the separation of the variables method it is possible to determine solutions of (4.9) of the form:

$$u(x,y) = X(x)Y(y). \tag{4.22}$$

Substituting equation (4.22) into equation (4.9), leads to two ordinary differential equations given by:

$$X'' + l^2 X = 0 \tag{4.23}$$

and

$$Y'' + (\gamma^2 - l^2)Y = 0, \tag{4.24}$$

where l is some constant. The general solutions are given by:

$$X(x) = A\cos lx + B\sin lx \tag{4.25}$$

and

$$Y(y) = C\cos(y\sqrt{\gamma^2 - l^2}) + D\sin(y\sqrt{\gamma^2 - l^2}). \tag{4.26}$$

Using the boundary conditions (4.20) and (4.21) it can be shown that:

$$\gamma_{nk}^2 = \left(\frac{2\pi n}{b}\right)^2 + \left(\frac{2\pi k}{a}\right)^2, \tag{4.27}$$

where n and k are 0, 1, 2... with $n = k = 0$ being the trivial solution. Also when the domain is square i.e. $a = b$:

$$\gamma_{nk} = \frac{2\pi}{a}\sqrt{n^2 + k^2}. \tag{4.28}$$

4.6 Numerical Results

The computational experiments in this section were carried out using MATLAB. All results are shown for eigenvalues scaled by π i.e. $\gamma_s = \gamma/\pi$ and the domain is $a = b = 1$. Figure 4.4 shows the layouts of the original points that are used in the generalised eigenvalue problem to calculate the expansion coefficient α: uniform, random and from a Sobol sequence [69].

Figure 4.5 shows how the actual relative error in the solution varies as a function of the total number of points (N). The actual relative error ϵ_r is defined at the ratio of the L_∞ norm of the difference between the analytical and the numerical solutions to the L_∞ norm of the analytical solution, i.e.

$$\epsilon_r = \frac{\|\gamma_{analytical} - \gamma_{numerical}\|_\infty}{\|\gamma_{analytical}\|_\infty}. \tag{4.29}$$

This figure also shows how the layout of the original points in the domain affects the accuracy of the meshless method results. The plots are for the first eigenvalue and are compared to the first analytical answer $\gamma_s = 2$. It can be seen from the figure that the best results come from the uniform layout of points but that even the random layout of points still gives accurate enough results for practical use. The Sobol sequence adds points by placing the next point at the maximum distance from the other points in the

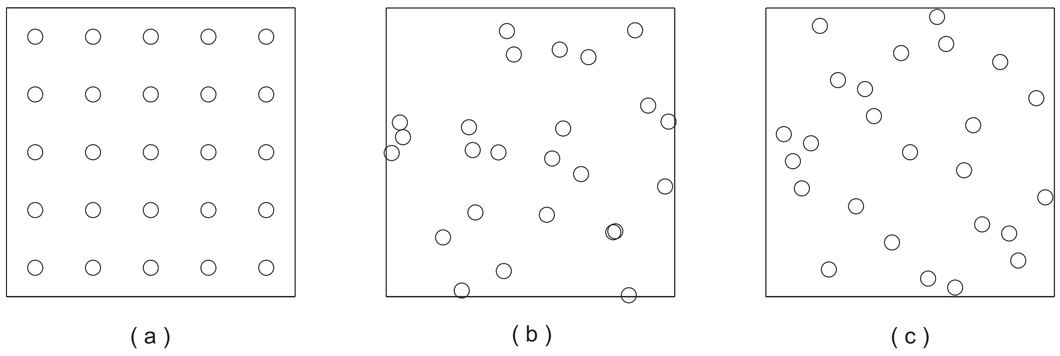


FIGURE 4.4: Design of experiment: (a) Uniform layout, (b) Random layout and (c) Sobol Layout.

domain. It is therefore possible to add just one extra point at a time making it more flexible than the uniform points, which requires going up to the next grid to increase accuracy. All of the point layouts show that as the number of points is increased, the accuracy of the method increases.

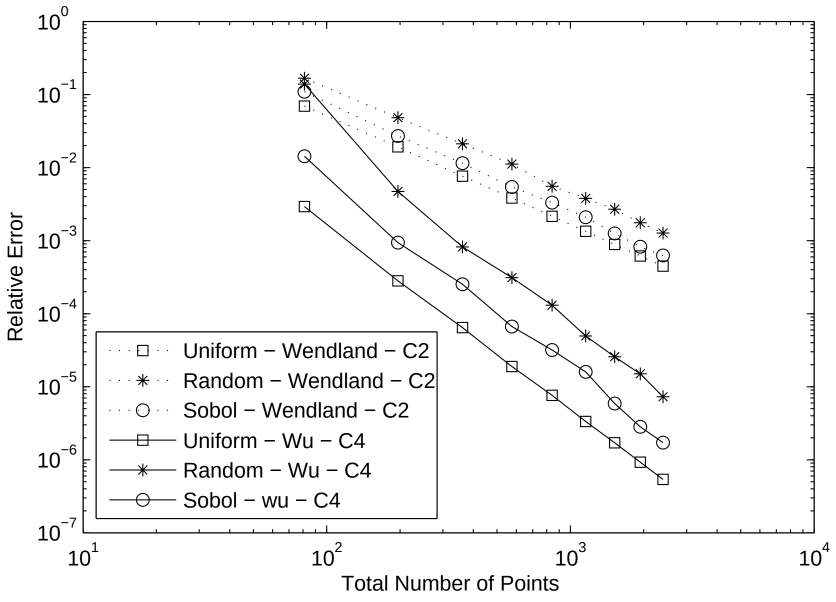


FIGURE 4.5: Relative error between analytical answer $\gamma_s = 2$ and meshless method result for various design of experiments.

Figure 4.6 shows the results from the experiment to obtain the convergence rate of the method, where h is the maximum mesh size. Using a straight line fit between the points, the method is found to converge at a rate of $O(h^5)$ for the C4 functions and at a rate of $O(h^3)$ for the C2 functions. In comparison to the central difference method, which is of order $O(h^2)$, the C4 functions will converge at a much faster rate than the FDM.

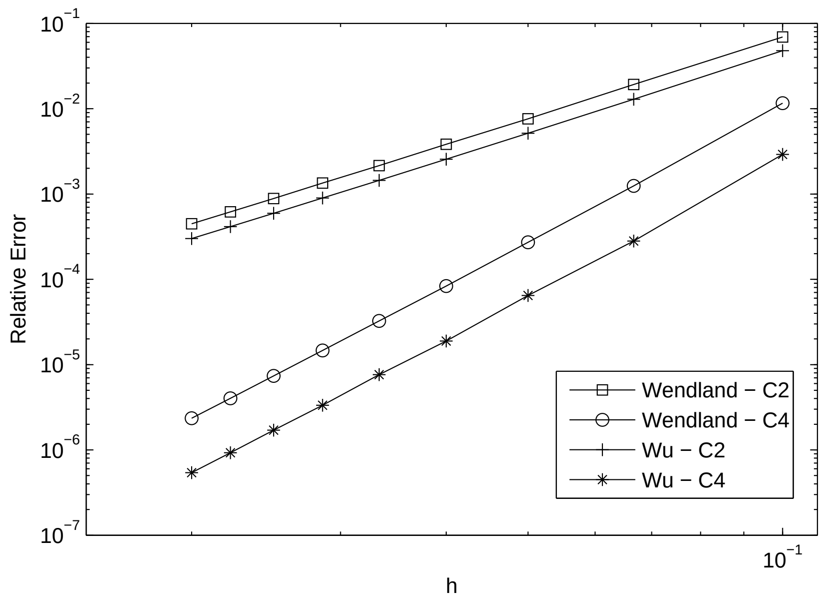


FIGURE 4.6: Relative error between analytical answer $\gamma_s = 2$ and meshless method result for various values of h using 1000 uniform points.

Figure 4.7 shows the results from the experiment to find the best shape parameter for the method. It can be seen from this figure that the result becomes more accurate as the shape factor increases toward 0.5, which is the value of half the domain length, and highly oscillatory for $c > 0.5$.

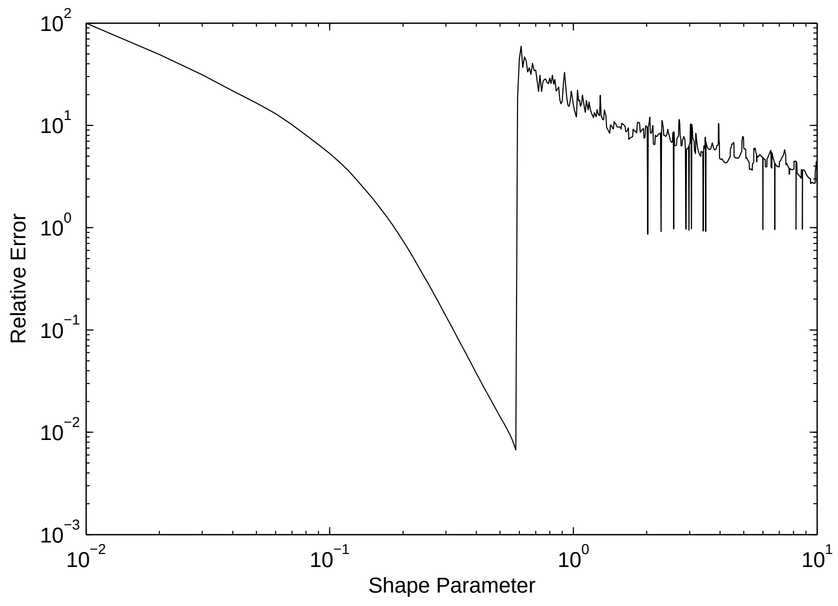


FIGURE 4.7: Relative error between analytical answer $\gamma_s = 2$ and meshless method result for various values of the shape parameter using 500 uniform points.

Figure 4.8 shows the relative errors between the analytical and meshless method results for the first forty non-zero eigenvalues which were obtained using a Sobol sequence of a 1000 points. From Figure 4.8, it is worth noting that there are only ten distinct eigenvalues (as they come in sets of four identical eigenvalues due to symmetry). A single point on the figure represents a set of 4 identical eigenvalues. From this figure it can be seen that the results obtained by Wu-C4 RBF are most accurate and the Wendland-C2 the least accurate. In addition, it can be seen that as the eigenvalue number increases the accuracy of the results decreases slightly.

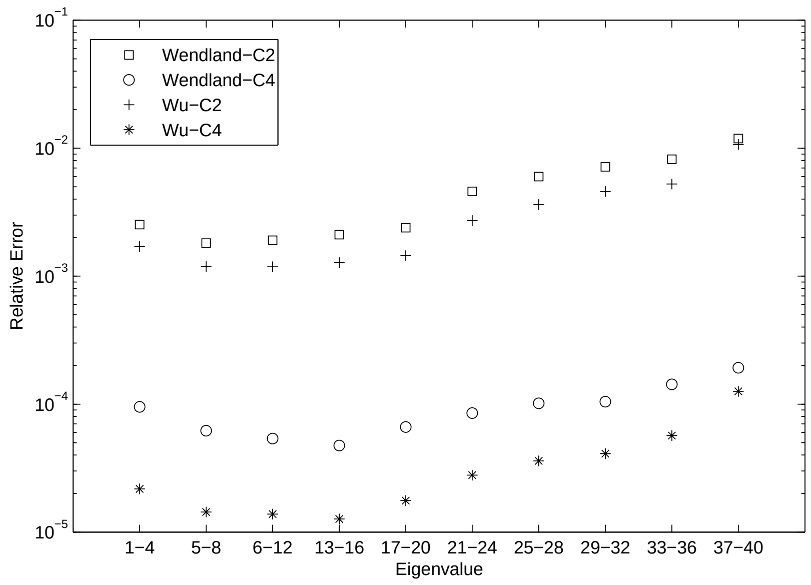


FIGURE 4.8: Relative error between analytical answers for first 40 non-zero λ_s and meshless result using Sobol sequence with 1000 points.

4.7 Conclusion

An eigenvalue problem with periodic boundary conditions was solved in this chapter using a CSRBF meshless method. It has been published in [5]. The method is able to obtain accurate eigenvalues from a set of initial points in the domain of interest, using uniform, random and Sobol points. In comparison, it is found that the results obtained using uniform points are slightly better than those obtained using random or Sobol points. In addition, it is found that the Wu-C4 RBF gives the most accurate results when compared with the analytical solutions to the problem and the Wendland-C2 the least accurate. The experiments show that the best value for the shape parameter of the RBF for this problem is c=0.5. Now that it has been demonstrated that is possible to use meshless methods to model a toroidal problem the next step is to apply a meshless method to Maxwell's equations.

Chapter 5

Photonic Crystal Modelling - Meshless Method

5.1 Introduction

In chapter 2 a FDM was written to solve Maxwell's equations and model two-dimensional photonic crystals. In chapter 4 a meshless method was used to solve a PDE with periodic boundary conditions. This chapter builds on the knowledge from those chapters and a new algorithm for modelling two-dimensional photonic crystals is presented. The TM mode was simple to implement as it required a MLSFM similar to the method used to solve the two-dimensional elliptic Helmholtz equation. The TE mode has discontinuities in ϵ, and requires a MLWFM. Gauss quadrature is used as a way to solve the integration required in this method.

This chapter discuses a new meshless method as an alternative way to model PhCs. Like the FDM the work is compared to the standard results produced by the PWEM. This chapter is structured as follows: section 5.2 presents the formulation of the meshless method for PhCs, the extra derivatives of the RBFs required for this method are presented. Sub-section 5.2.1 details the formulation of the MLSFM method and sub-section 5.2.2 the MLWFM method. The band diagrams produced by the meshless methods are shown in section 5.3 before conclusions are drawn in section 5.4.

5.2 Meshless Method

As discussed in chapter 1, for two-dimensional PhCs only the waves propagating though the plane of periodicity are considered and the problem can be split into two scalar spectral equations; one for the transverse electric polarisation (TE mode) and the other for the transverse magnetic polarisation (TM mode) [70, 71]. Equation (5.1) and (5.2) show the TE mode and TM mode respectively:

$$\begin{aligned} -\nabla \cdot \frac{1}{\epsilon}\nabla\psi &= \lambda\psi, && (5.1)\\ -\frac{1}{\epsilon}\Delta\psi &= \lambda\psi, && (5.2)\end{aligned}$$

where λ is the spectral parameter,

$$\lambda = \left(\frac{\omega}{c}\right)^2. \tag{5.3}$$

The PhC is modelled as infinite, by imposing periodic boundary conditions on the unit cell, then the Bloch-Floquet theory can be applied [72].

Consequently the wave function can be represented as:

$$\psi = e^{\imath \mathbf{k}\mathbf{x}} \cdot u(\mathbf{x}), \tag{5.4}$$

where $\mathbf{k} = \begin{pmatrix} k_1 \\ k_2 \end{pmatrix}$ is the quasimomentum vector and $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$.

5.2.1 Meshless Local-Strong Form Method

For the TM mode, substituting equation (5.4) into (5.2) gives:

$$-(\nabla + \imath\mathbf{k}) \cdot (\nabla + \imath\mathbf{k})u = \epsilon(\mathbf{x})\lambda u. \tag{5.5}$$

Representing $u = \sum_{j=1}^{n} \alpha_j\phi_j$ where $\phi_j(\mathbf{x}_i) \equiv \phi(||\mathbf{x}_i - \mathbf{x}_j||)$ gives rise to a generalised eigenvalue problem of the form:

$$A(\mathbf{k})\boldsymbol{\alpha} = \lambda B\boldsymbol{\alpha}, \tag{5.6}$$

where $\boldsymbol{\alpha}$ are the eigenvectors of each eigensystem corresponding to the nodal field values of allowable modes of propagation through the PhC and λ are the respective eigenvalues that correspond to the frequencies of the mode [71]. The A and B matrices are:

$$\begin{aligned} A_{ij} &= -(\nabla + \imath\mathbf{k}) \cdot (\nabla + \imath\mathbf{k})\phi_j(\mathbf{x}_i), && (5.7)\\ B_{ij} &= \epsilon(\mathbf{x})\phi_j(\mathbf{x}_i). && (5.8)\end{aligned}$$

By expanding the equation that makes up the matrix A_{ij} it is possible to construct the following eigensystem matrices:

$$\begin{aligned} E_{ij} &= \nabla^2 \phi_j(\mathbf{x}_i), & (5.9)\\ \mathbf{F}_{ij} &= \nabla \phi_j(\mathbf{x}_i), & (5.10)\\ H_{ij} &= \phi_j(\mathbf{x}_i). & (5.11) \end{aligned}$$

Hence for the TM mode:

$$A_{ij} = -E_{ij} - 2\mathbf{k}\imath \cdot \mathbf{F}_{ij} + \mathbf{k}^2 H_{ij}. \tag{5.12}$$

In order to calculate the **F** matrix extra derivatives were required. Table 5.1 shows the derivatives of Wendland and Wu's CSRBFs with respect to x and y.

Wendland-C2
$\frac{\partial\phi(r)}{\partial x} = \frac{20}{c^5}(c-r)^3(x-x_i)$ $\frac{\partial\phi(r)}{\partial y} = \frac{20}{c^5}(c-r)^3(y-y_i)$
Wendland-C4
$\frac{\partial\phi(r)}{\partial x} = (1-\frac{r}{c})^5(2x-2x_i)(-\frac{28}{c^2}-\frac{140r}{c^3})$ $\frac{\partial\phi(r)}{\partial y} = (1-\frac{r}{c})^5(2y-2y_i)(-\frac{28}{c^2}-\frac{140r}{c^3})$
Wu-C2
$\frac{\partial\phi(r)}{\partial x} = (1-\frac{r}{c})^4(2x-2x_i)(-\frac{72}{c^2}-\frac{261r}{2c^3}-\frac{96r^2}{c^4}-\frac{45r^3}{2c^5})$ $\frac{\partial\phi(r)}{\partial y} = (1-\frac{r}{c})^4(2y-2y_i)(-\frac{72}{c^2}-\frac{261r}{2c^3}-\frac{96r^2}{c^4}-\frac{45r^3}{2c^5})$
Wu-C4
$\frac{\partial\phi(r)}{\partial x} = (1-\frac{r}{c})^5(2x-2x_i)(-\frac{44}{c^2}-\frac{220r}{c^3}-\frac{264r^2}{c^4}-\frac{275r^3}{2c^5}-\frac{55r^4}{2c^6})$ $\frac{\partial\phi(r)}{\partial y} = (1-\frac{r}{c})^5(2y-2y_i)(-\frac{44}{c^2}-\frac{220r}{c^3}-\frac{264r^2}{c^4}-\frac{275r^3}{2c^5}-\frac{55r^4}{2c^6})$

TABLE 5.1: $\frac{\partial\phi(r)}{\partial x}$ and $\frac{\partial\phi(r)}{\partial y}$ of Wendland and Wu CSRBFs.

For the TE mode, substituting equation (5.4) into (5.1) gives:

$$-(\nabla + \imath\mathbf{k}) \cdot \frac{1}{\epsilon(\mathbf{x})}(\nabla + \imath\mathbf{k})u = \lambda u. \tag{5.13}$$

Expanding the left of the equation leads to:

$$-\nabla \cdot \left(\frac{1}{\epsilon(\mathbf{x})}\nabla u\right) - \imath\nabla \cdot \left(\frac{1}{\epsilon(\mathbf{x})}\mathbf{k}u\right) - \imath\mathbf{k} \cdot \left(\frac{1}{\epsilon(\mathbf{x})}\nabla u\right) + \mathbf{k} \cdot \left(\frac{1}{\epsilon(\mathbf{x})}\mathbf{k}u\right) = \lambda u, \tag{5.14}$$

where

$$\nabla \cdot \left(\frac{1}{\epsilon(\mathbf{x})}\nabla u\right) = \frac{\partial}{\partial x}\left(\frac{1}{\epsilon(\mathbf{x})}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{1}{\epsilon(\mathbf{x})}\frac{\partial u}{\partial y}\right), \tag{5.15}$$

which is a problem as the dielectric constant $\epsilon(\mathbf{x})$ is discontinuous. This means an alternative meshless method is needed to solve the problem for TE.

5.2.2 Meshless Local-Weak Form Method

The solution to the differential equation (5.13), given the periodic boundary conditions, is found using Galerkin's method. The method calculates a discretised approximation to the true solution of the boundary value problem, which gives the following integral for TE:

$$\int \frac{1}{\epsilon(\mathbf{x})}(\nabla + \imath\mathbf{k})u \cdot \overline{(\nabla + \imath\mathbf{k})v}d\mathbf{x} = \lambda \int u\bar{v}d\mathbf{x}. \tag{5.16}$$

Representing $u = \sum_{j=1}^{n} \alpha_j \phi_j$ and $v = \sum_{l=1}^{n} \alpha_l \phi_l$ gives rise to a generalised eigenvalue problem of the form:

$$C(\mathbf{k})\boldsymbol{\alpha} = \lambda D\boldsymbol{\alpha}, \tag{5.17}$$

where the C and D matrices are, for the TE mode:

$$C_{jl} = \int \frac{1}{\epsilon(\mathbf{x})}(\nabla + \imath\mathbf{k})\phi_j \cdot \overline{(\nabla + \imath\mathbf{k})\phi_l}d\mathbf{x}, \tag{5.18}$$

$$D_{jl} = \int \phi_j \phi_l d\mathbf{x}. \tag{5.19}$$

By expanding the integral that makes up the matrix C_{jl} it is possible to construct the following eigensystem matrices:

$$\begin{aligned} S_{jl} &= \int \frac{1}{\epsilon(\mathbf{x})} \nabla\phi_j \cdot \nabla\phi_l d\mathbf{x}, & (5.20)\\ P_{jl} &= \int \frac{1}{\epsilon(\mathbf{x})} \mathbf{k}\phi_j \cdot \nabla\phi_l d\mathbf{x}, & (5.21)\\ Q_{jl} &= \int \frac{1}{\epsilon(\mathbf{x})} \mathbf{k}\phi_l \cdot \nabla\phi_j d\mathbf{x}, & (5.22)\\ T_{jl} &= \int \frac{1}{\epsilon(\mathbf{x})} \phi_j \cdot \phi_l d\mathbf{x}. & (5.23) \end{aligned}$$

Hence for the TE Mode polarisation:

$$C_{jl} = S_{jl} + \imath(P_{jl} + Q_{jl}) + \mathbf{k}^2 T_{jl}. \tag{5.24}$$

This approach can also be applied to the TM mode. The solution to the differential equation (5.5) is again found using Galerkin's method and gives the following integral for TM:

$$\int (\nabla + \imath\mathbf{k})u \cdot \overline{(\nabla + \imath\mathbf{k})v} d\mathbf{x} = \lambda \int \epsilon(\mathbf{x}) u\bar{v} d\mathbf{x}, \tag{5.25}$$

which is another generalised eigenvalue problem with C and D matrices defined as:

$$C_{jl} = \int (\nabla + \imath\mathbf{k})\phi_j \cdot \overline{(\nabla + \imath\mathbf{k})\phi_l} d\mathbf{x}, \tag{5.26}$$

$$D_{jl} = \int \epsilon(\mathbf{x})\phi_j \cdot \phi_l d\mathbf{x}. \tag{5.27}$$

By expanding the integral that makes up the matrix C_{jl} it is possible to construct the following eigensystem matrices:

$$\begin{aligned} S^1_{jl} &= \int \nabla\phi_j \cdot \nabla\phi_l d\mathbf{x}, & (5.28)\\ P^1_{jl} &= \int \mathbf{k}\phi_j \cdot \nabla\phi_l d\mathbf{x}, & (5.29)\\ Q^1_{jl} &= \int \mathbf{k}\phi_l \cdot \nabla\phi_j d\mathbf{x}, & (5.30)\\ T^1_{jl} &= \int \phi_j \cdot \phi_l d\mathbf{x} & (5.31) \end{aligned}$$

where the superscript is used for the TM mode.

Hence for the TM mode polarisation:

$$C_{jl} = S^1_{jl} + \imath(P^1_{jl} + Q^1_{jl}) + \mathbf{k}^2 T^1_{jl}. \tag{5.32}$$

Although it is possible to solve TM using the MLWFM,the MLSFM method is preferred as it requires less computation time.

When a unit cell is discretised by a set of nodes, the nodes can be located exactly on the interface. Unfortunately in this situation, it is not clear which dielectric constant should be assigned to the interface nodes, as the two materials have different dielectric constants. Using the RBF based meshless method will avoid this problem, as it employs a set of integration points (Gaussian points), apart from from the set of nodes, to calculate the integrals in equations (5.19-5.23) and (5.27-5.31). These integration points will be located in either one of the materials, but not on the interface, and thus a given value of dielectric constant can be easily assigned to them. This will make the numerical implementation easy, and not requiring additional methods, such as, for example averaging or smoothing the dielectric function.

5.2.2.1 Gauss Quadrature

The numerical integration in the MLWFM is solved using Gauss Quadrature. In two dimensions the integration over a quadrilateral with $a \leq x \leq b$ and $c \leq y \leq d$ is given by:

$$\int_a^b \int_c^d f(x,y)dxdy = \int_{-1}^{1} \int_{-1}^{1} f(x(\xi), y(\eta))|J|d\xi d\eta, \tag{5.33}$$

where $|J|$ is the Jacobian Matrix, and f is given by $\nabla\phi_j \cdot \nabla\phi_l$ for equation (5.28), $k\phi_j \cdot \nabla\phi_l$ for equation (5.29), $k\phi_l \cdot \nabla\phi_j$ for equation (5.30) and $\phi_j \cdot \phi_l$ for equation (5.31) respectively (for the TM mode). Substituting $x = \left(\frac{b-a}{2}\right)\xi + \left(\frac{b+a}{2}\right)$, and $y = \left(\frac{d-c}{2}\right)\eta + \left(\frac{d+c}{2}\right)$ in to equation (5.33) leads to:

$$\int_a^b \int_c^d f(x,y)dxdy =$$
$$\left(\frac{b-a}{2}\right)\left(\frac{d-c}{2}\right)\int_{-1}^{1}\int_{-1}^{1} f\left(\frac{b-a}{2}\xi + \frac{b+a}{2}, \frac{d-c}{2}\eta + \frac{d+c}{2}\right) d\xi d\eta. \tag{5.34}$$

Figure 5.1 shows a quadrilateral with centre $\xi = 0, \eta = 0$ and 4 Gauss points. Gauss quadrature evaluates an integral as a sum of finite terms, thus:

$$\left(\frac{b-a}{2}\right)\left(\frac{d-c}{2}\right)\int_{-1}^{1}\int_{-1}^{1} f\left(\frac{b-a}{2}\xi + \frac{b+a}{2}, \frac{d-c}{2}\eta + \frac{d+c}{2}\right) d\xi d\eta =$$
$$\sum_{i=1}^{n_x}\sum_{j=1}^{n_x} W_{ix} \cdot W_{jy} \cdot f\left(\frac{b-a}{2}\xi_i + \frac{b+a}{2}, \frac{d-c}{2}\eta_j + \frac{d+c}{2}\right). \tag{5.35}$$

When for 4 Gauss points $n_x = n_y = 2$, $W_{ix} = W_{jy} = 1$ and $\xi_1 = \eta_1 = \frac{\sqrt{3}}{3}$, $\xi_2 = \eta_2 = -\frac{\sqrt{3}}{3}$.

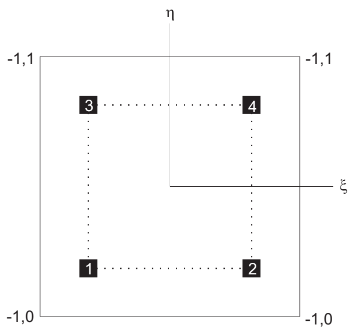


FIGURE 5.1: Gauss point layout for the 4 point rule.

A set of uniform nodes is created that cover the unit cell. For two-dimensions there should be between 3 and 9 times more Gauss points than there are nodes [43]. Then a set of background cells is generated that covers the unit cell and each background cell has 4 gauss points and its own local support domain with radius r. Therefore for the first background cell with Gauss points 1-4, if nodes 6 and 7 were both in the support domain, equation (5.28) can be written as:

$$S^1_{67} = \phi_6(Gp_1) \cdot \phi_7(Gp_1) + \phi_6(Gp_2) \cdot \phi_7(Gp_2) + \phi_6(Gp_3) \cdot \phi_7(Gp_3) + \phi_6(Gp_4) \cdot \phi_7(Gp_4), \tag{5.36}$$

where ϕ is a CSRBF. However nodes 6 and 7 will also be included in other support domains so the values for each S^1_{67} must be summed together to assemble the global S^1 matrix.

5.3 Results

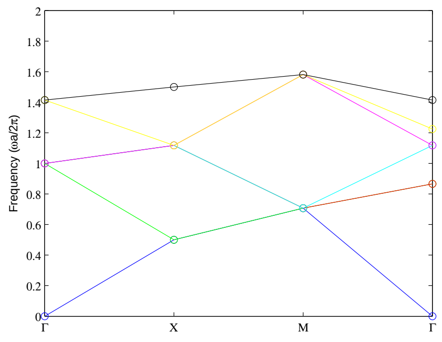
The results from the new algorithms can be validated by comparison with other theoretical results presented in the literature. There are two types of theoretical result, exact (analytical) and approximate (numerical) solutions.

5.3.1 Comparison with Analytical Results

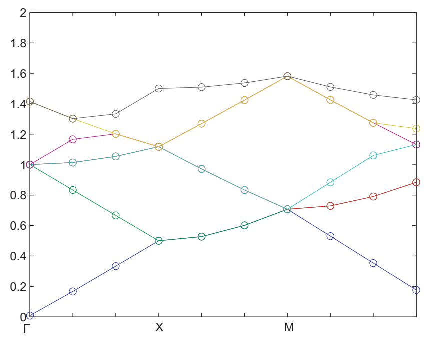
The analytical solution for the propagation mode frequencies for free space can be found using the following equation:

$$\lambda = [\imath(\mathbf{G} + \mathbf{k})]^2 = (\mathbf{G} + \mathbf{k})^2, \tag{5.37}$$

where $\mathbf{G}$ is the reciprocal lattice vector and $\mathbf{k}$ is the wave vector. A full derivation of equation (5.37) can be found in [73]. The unit cell used in the meshless method for comparison is a square with periodic boundary conditions and the dielectric constant set to one ($\epsilon_{air} = 1$). The analytical solutions for the normalised frequencies at the corners of the irreducible Brillouin zone are shown in Figure 5.2(a). Figure 5.2(b) shows the results produced by the MLSFM, using Wu's C2 and 20 uniform grid points. The band diagrams show excellent agreement at the points which represent the corners of the irreducible Brillouin zone (Γ, X, M).



(a) Analytical Solution



(b) MLSFM Wu's C2 function

FIGURE 5.2: Various band diagrams showing the modes for an air unit cell ($\epsilon = 1$)

5.3.2 Comparison with Numerical Results

The computational experiments in this section were carried out using MATLAB. A uniform point layout was used for all of the experiments. In this chapter, the Wu-C2 and Wu-C4 CSRBFs are chosen, as from chapter 4 they are found to have a better convergence rate than the coresponding Wendland C2 and C4 CSRBFs. Figure 5.3 shows the band diagram for a square array of alumina rods in air that was produced by the PWEM. Both the TE (red lines) and the TM (blue lines) band structures are shown. Figure 5.3 is the same as Figure 2.2 but is repeated for ease of comparison with the results from the meshless methods. Note for the band diagrams produced by the meshless methods the x-axis goes from Γ to X to M before returning to just before the Γ point.

Figure 5.4(a) shows a band diagram of the TM mode, produced by the MLSFM using Wu's C2 CSRBF and uniform grid of 40 by 40 nodes, for the same square array of alumina rods in air. Figure 5.4(b) shows the band diagram of the TM mode, produced by the MLSFM using Wu's C4 CSRBF and uniform grid of 40 by 40 nodes, for the same square array of dielectric columns. Both figures are in good agreement with the standard PWEM figure. It can be seen from these two figures that using the C4 RBF produces more accurate results, which is in agreement with the results produced in section 4.6. Both figures show the correct band gap between mode 1 and mode 2.

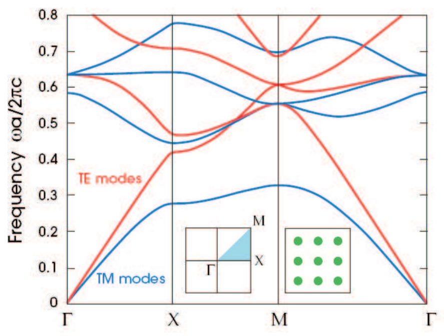
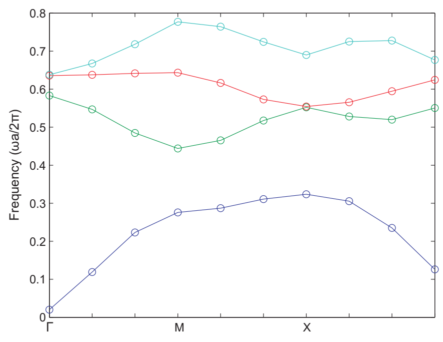
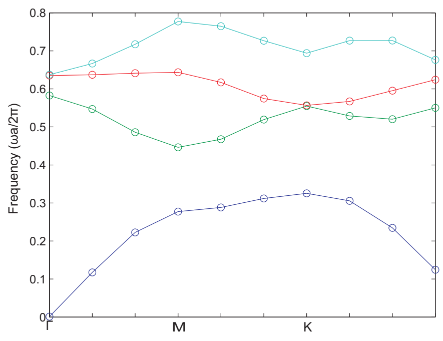


FIGURE 5.3: Reproduced from [2] a band diagram for a square array of dielectric columns ($\epsilon = 8.9$) with $r = 0.2a$ in air ($\epsilon = 1$). The blue bands represent the TM modes and the red bands represent the TE modes. The left inset shows the Brillouin zone, with the irreducible zone shaded light blue. The right inset shows a cross-sectional view of the dielectric.



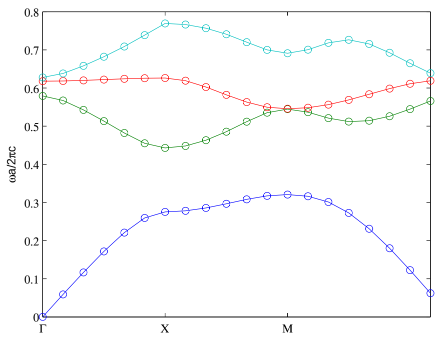
(a) Wu's C2



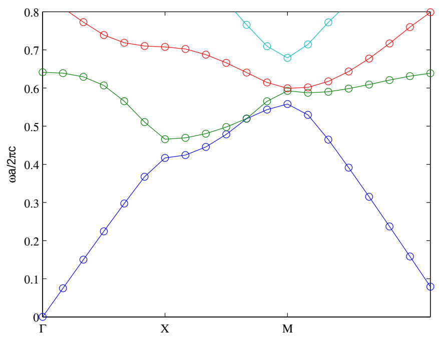
(b) Wu's C4

FIGURE 5.4: Various band diagrams showing the TM modes for a square array of dielectric columns ($\epsilon = 8.9$) with $r = 0.2a$ in air ($\epsilon = 1$) produced by the MLSFM

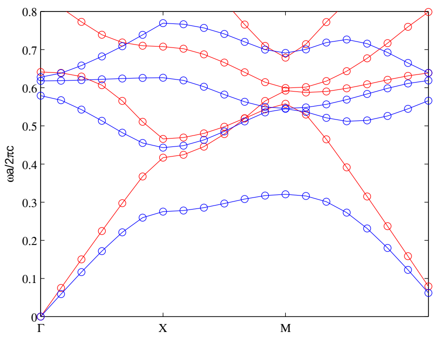
Figure 5.5(a) shows the band diagram of the TM mode, produced by the MLWFM using Wu's C4 function, a uniform grid of 31 by 31 nodes and 37 by 37 background cells each with 4 Gauss points (5476 gauss points). The figure shows the PBG between modes 1 and 2 and is in good agreement with the MLSFM and the PWEM. In this figure using more steps as the k vector goes around the Brillouin zone gives a smoother curve but this required more computation. Figure 5.5(b) shows the band diagram for the TE mode. This figure shows that unlike the MLSFM the MLWFM can also produce the correct figure for TE. Figure 5.5(c) shows the TE and TM modes and correctly has no complete PBG.



(a) TM



(b) TE



(c) TM(blue) and TE(red)

FIGURE 5.5: Various band diagrams showing the modes for a square array of dielectric columns ($\epsilon = 8.9$) with $r = 0.2a$ in air ($\epsilon = 1$) produced by the MLWFM

Figure 5.6 shows the band structure for a square array of dielectric ($\epsilon = 8.9$) veins in air ($\epsilon = 1.0$). The veins have a thickness $= 0.165a$. Both the TE (red lines) and the TM (blue lines) band structures are shown. Figure 2.4 is the same as Figure 5.6 but is repeated for ease of comparison with the results from the meshless methods.

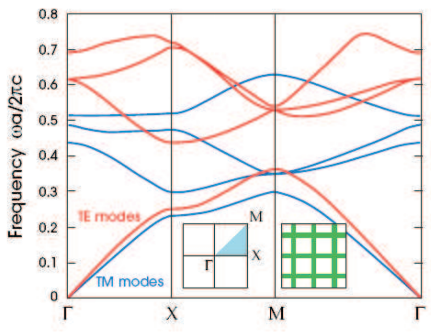
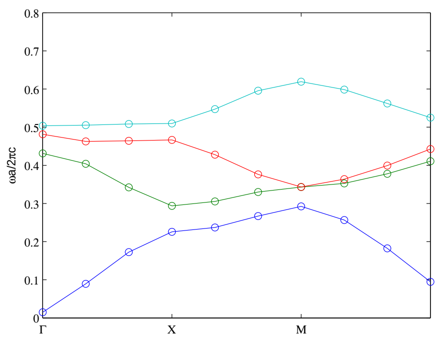
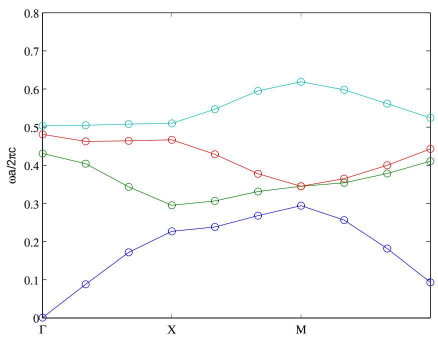


FIGURE 5.6: Reproduced from [2] a band diagram for the lowest frequency modes of a square array of dielectric ($\epsilon = 8.9$) veins (thickness $= 0.165a$) in air ($\epsilon = 1.0$). The blue bands represent the TM modes and the red bands represent the TE modes. The left inset shows the Brillouin zone, with the irreducible zone shaded light blue. The right inset shows a cross-sectional view of the dielectric.



(a) Wu's C2

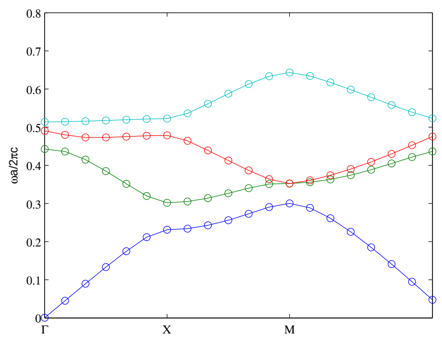


(b) Wu's C4

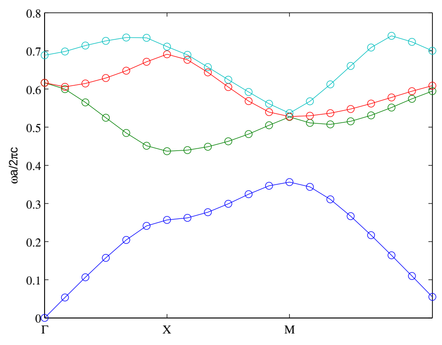
FIGURE 5.7: Various Band Diagram showing the TM modes for a square array of dielectric veins ($\epsilon = 8.9$) with $thickness = 0.165a$ in air ($\epsilon = 1.0$) produced by the MLSFM.

Figure 5.7(a) shows a Band Diagram of the TM mode, produced by the MLSFM using Wu's C2 CSRBF and uniform grid of 40 by 40 nodes, for the same a square array of alumina veins in air. Figure 5.7(b) shows the Band Diagram of the TM mode, produced by the MLSFM using Wu's C4 CSRBF and uniform grid of 40 by 40 nodes, for the same square array of dielectric veins. Both figures are in good agreement with the standard PWEM figure and correctly show no PBG.

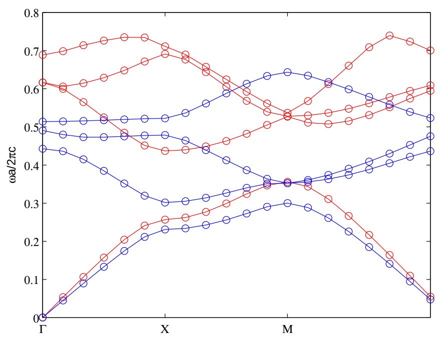
Figure 5.8(a) shows the band diagram of the TM mode, produced by the MLWFM using Wu's C4 function, a uniform grid of 31 by 31 nodes (961 nodes) and 37 by 37 background cells each with 4 Gauss points (5476 gauss points). The figure shows no PBGs and is in good agreement with the MLSFM and the PWEM. Figure 5.8(b) shows the band diagram for the TE mode. The figure correctly shows PBG between modes 1 and 2. Figure 5.8(c) shows the TE and TM modes and correctly has no complete PBG.



(a) TM



(b) TE



(c) TM(blue) and TE(red)

FIGURE 5.8: Various Band Diagram showing the modes for a square array of dielectric veins ($\epsilon = 8.9$) with $thickness = 0.165a$ in air ($\epsilon = 1.0$) produced by the MLWFM.

5.4 Conclusions

In this chapter a new algorithm was presented for use in PhC modelling. The MLSFM is able to obtain TM mode bands that are in good agreement with the standard PWEM. The MLWFM is able to obtain results for both the TM and TE modes that are in agreement with the standard PWEM. Both methods agree with the results found in chapter 4 that the C4 RBFs produce more accurate results that the C2 RBFs. However, due to the large matrices that need to be constructed the MLWFM is very computationally expensive and needs to be optimised further before it can be used as viable modelling algorithm. The next chapter looks at how accelerator technologies could be used can help with filling the large matrices.

Chapter 6

Technologies

6.1 Introduction

This chapter looks to explore the possibility of speeding up the solving of the two problems that the meshless method presents, the matrix fills and the eigenvalue solve. A Graphics Processing Unit (GPU) is presented as an alternative technology which may help with this problem as the matrix fills require the same computation to be done on large amounts of data. Initially a GPU served as a purely dedicated graphics rendering device. However, the introduction of programmable vertex and fragment processors has enabled access to their computing capabilities for general purpose computation. GPUs are attractive for use in High Performance Computing (HPC) applications because they offer multiple cores and very high memory bandwidth, whilst cost stays low due to the high demand from the gaming industry. It can be seen from Figure 6.1 that they offer more floating point operations per second than a CPU.

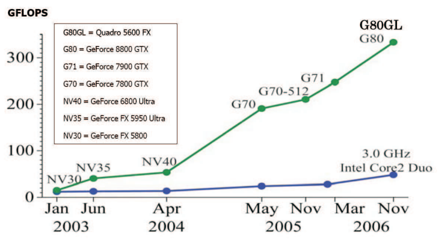


FIGURE 6.1: Floating-point operations per second for the CPU and GPU. Reproduced from [4].

Improvements in semiconductor capabilities and fabrication have increased the performance of both types of hardware but GPUs have increased further due to their architecture. CPUs have large caches and are optimised for high performance of multiple operations on multiple data whereas GPUs are highly optimised for single operations on multiple data. In addition, GPUs offer potential speed ups to specialist problems that can take advantage of their shared memory for multiple threads versus the distributed memory of a compute cluster.

The idea to implement general-purpose algorithms on computer graphics hardware has been introduced more than fifteen years ago, when Lengyel *et al.* used a rasterization device for robot motion planning [74]. Currently graphics hardware is being used for many things including, encryption, computational geometry, audio and signal processing, scientific simulation, computational finance and database management. There are currently two mainstream manufacturers offering commodity off the shelf solutions for general purpose graphics programming: NVIDIA and ATI. These were selected to be worked with because they could present a cost-effective solution in reach of most engineers.

6.2 Parallel Computing

6.2.1 Graphics Processing Unit

The GPU was designed to create on screen images. In August 1999 NVIDIA introduced their GeForce 256 SDR which was the world's first GPU. In 2000 ATI Technologies, now AMD, released the Radeon R100 to be in direct competition with NVIDIA. In 2009 NVIDIA and AMD remain the industry leaders in high performance GPUs. Both have released new cards in 2009. NVIDIA added to the GeForce 200 series with its the GeForce GTX 285. AMD released the Radeon Evergreen (HD 5xxx) series with the most powerful card being Radeon HD 5970. Table 6.1 contains a direct comparison between the GeForce 256 SDR, GeForce GTX 285 and Radeon HD 5970 showing how quickly the industry has moved in less than ten years.

	GF 256 SDR	GF GTX 285	R HD 5970
Memory Amount(MiB)	32	1024/2048	2048
Memory Bandwidth	2.7 GB/s	159.0 GiB/s	256GB/s
Texture Fillrate (MT/s)	480	51840	116000
DirectX	7	10	11
OpenGL	1.2	3.2	3.2
OpenCL	-	1.0	1.0

Table 6.1: Comparisons between the first GPU and the current GPUs.

The current line of GPUs from Intel are known as Intel Graphics Media Accelerator (GMA) and are integrated hardware which are almost exclusively aimed at low to middle end users. The built-in chip offers adequate rendering performance for less cost and power consumption, however functionality and performance of these chips is poor compared to more expensive discrete graphics components [75].

6.2.2 IBM Cell Processor

The Cell Broadband Engine Architecture (CBEA) is a microprocessor that was jointly developed by Sony Computer Entertainment, Toshiba and IBM (STI) to be the heart of the Sony PlayStation3 [76]. The architecture differs from a conventional multiprocessor as instead of using a set of the same cooperating commodity processors, it has a conventional PowerPc (PPC) core that controls eight synergistic processor elements (SPEs). A SPE is a simple SIMD core that contains a synergistic processing unit (SPU), a local memory of 256KB and a memory flow controller. Access to external memory is via the 25.6GB/s XDR memory controller. Four data rings (Element Interconnect Bus) connect the PPC, the eight SPEs, the DRAM controller and the I/O controllers. In April 2008 the fix pack, 3.0-SDKMA-Linux-FP03 was released and contains support for three libraries, BLAS, LAPACK and a Monte Carlo Random Number Generator.

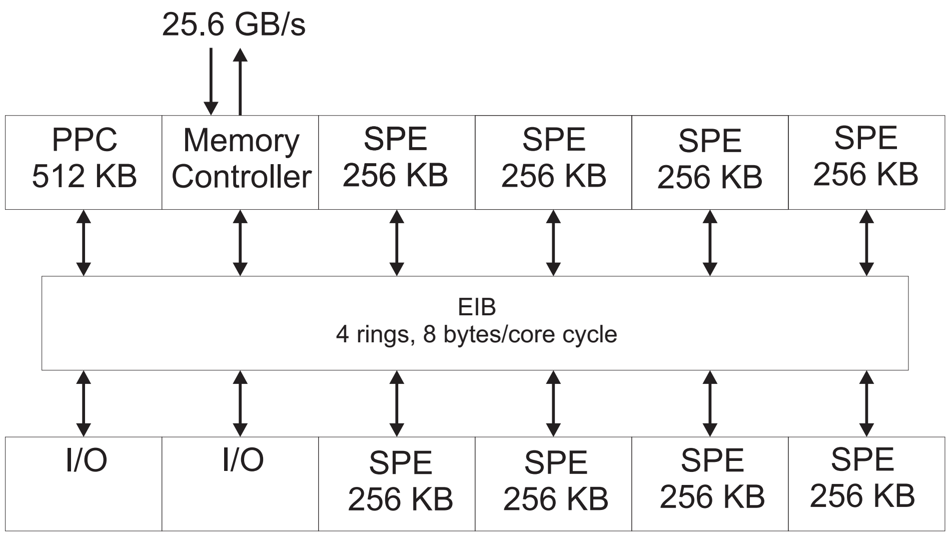


FIGURE 6.2: Overview of the Cell processor.

6.2.3 Multi Core CPUs

Recently Multi core CPUs have become common in new computers. Intel's first dual core processor based platforms include the Pentium Process Extreme Edition 840 which was released in May 2005. Intel's first quad core processors, code named Kentsfield, were released in November 2006. Its top of the range Core 2 Extreme models were numbered

QX6xx0. AMDs first dual core desktop CPU was the Athlon 64 X2 and was released in April 2005 and in November 2007 they released Phenom 9700, their first quad core CPU. The latest Intel processor is the Core 2 Quad Q9550 which has a 2x6144 KiB L2 cache, 2.833 GHz clock speed and a 1333 MHz front side bus compared with AMDs Phenom X4 9850 that has a 4 x 512 KiB L2 cache and a 2 MiB L3 cache, 2.5 GHz clock speed and 2000MHz Hyper Transport. The are many template based libraries available to use these architectures to their full potential, including Intel Threading Building Blocks, which is a runtime based parallel programming model for C++ code that uses threads [77]. Cilk++ is an example of one of the available off the shelf software solutions which help maximise application performance on this these multi core CPUs [78]. Multi Core CPU architectures are designed for general purpose computing and application multi tasking. Whilst they offer high floating point performance per core, a core can only run a single operation at any point in time which restricts their degree of parallelisation.

6.2.4 Clearspeed Accelerators

The Advance e710 and Advance e720 are the newest low power accelerator boards made by Clearspeed Technology, which work in conjunction with the CPU to share the computer intensive parts of an application [79]. The Advance e710 includes one ClearSpeed CSX700 and 2 GBytes of ECC-protected DRAM. When an application makes a call to a standard mathematics library the call is detected by the Clearspeed accelerated math library (CSXL) which then determines if the function call can be accelerated. The CSXL maths library supports both level 3 BLAS and LAPACK functions. There is also a Clearspeed SDK that allows for the development of new applications on a Clearspeed accelerator. The Monte Carlo simulation, an algorithm for finance, was 20.3 times faster when using one Clearspeed card compared to a single dual core 2.4GHz AMD Opteron processor. The advanced accelerator boards support windows and Linux RedHat and SUSE enterprise.

6.2.5 General Purpose Graphics Processing Units

The Tesla GPU is NVIDIA's first dedicated General Purpose GPU (GPGPU) and is intended to be used for HPC [80]. Tesla cards are based on G80 and Quadro but without the display device. Tesla is available as a computing processor that has 128 thread processors and 1.5GB dedicated memory (C870), a deskside system (D87) that consists of 2 tesla GPUs and a server rack (S870) that contains 4 tesla GPUs. Tesla now supports both Windows XP and Red Hat Linux 4 and 5.

In November 2007 AMD announced the FireStream 9170 which is a stream processor or GPGPU that was designed to be used for HPC [81]. Released in May 2008 the FireStream 9170 is as computing processor that has 320 stream cores and 2.0GB of dedicated memory. It is compatible with 32 and 64 bit Windows XP and Linux environments. The FireStream 9250 is AMD's second generation stream processor and has broken the TeraFLOP barrier using under 150 watts. The FireStream 9270 is the newest AMD stream processer.

Larrabee is Intel's discrete GPU being designed explicitly as a GPGPU [82]. It was originally scheduled for release in late 2009 but has now been pushed back to 2010/2011. It will use a derivative of the x86 instruction set for its shader cores instead of a custom graphics oriented instruction set, which in theory should make it more flexible than currently available GPUs.

6.2.6 Distributed Computing

Distributed computing uses a cluster of networked computers in which each processor has its own private memory and information is exchanged by passing messages between the processors. In a conventional off-the-shelf-cluster each node has a CPU as its processor. Recently clusters have begun to include accelerators e.g. GPUs and Cells. In June 2009 the Roadrunner cluster, which has 129600 cores, was the largest supercomputer in the world [83].

6.2.7 Comparison

GPUs were investigated as an alternative parallel technology to use to try and accelerate the meshless method because they are cheap and powerful. Almost all modern desktop computers have a GPU and there is competition to make them better due to multiple manufactures. The Cell offers an alternative to the GPU but they are less accessible due to being in the PlayStation or specifically purchased blades and not everyday computers. ClearSpeed accelerators are promising for specialist operations but were rejected as they are not very mainstream and are difficult to program. Finally multi-core CPUs and conventional clusters were not chosen, as this area has already been widely explored for parallelisation.

6.3 Languages

6.3.1 Shading Languages

Both shape and shading affect the appearance of an image created using computer generated imagery. An object's shape is affected by the geometry of its surfaces and its position with respect to the users view point. The shade of an object depends on the optical properties of its surface and the way in which it is illuminated. Amazing images can be created using very straightforward shapes that have complicated shading. A shader is a set of instructions used by the GPU to define the final surface properties of an onscreen object. The vertex shader and the pixel shader carry out different tasks in the GPU programmable pipeline. Being able to load programs into the vertex and pixel shader memory introduces a higher level of flexibility in the pipeline, compared to the older fixed-function pipeline.

One of the earliest forms of programmable shading was carried out by Max [84] whilst trying to create a realistic water effect. The idea of rewriting the shading code for the renderer was built upon by Whitted and Weimer when they implemented their testbed system [85]. The idea that a single predefined model would never be able to produce all the objects required for a complicated scene inspired Cook's work on shade trees [86]. The shade tree system allows shaders to be read in, parsed and then executed. Perlin took Cook's idea further and developed a Pixel Stream Editing language (PSE), which is a high level programming language available at the pixel level [87]. A pixel stream editor creates an output image by carrying out the same set of instruction on every pixel of the input image. The work of Cook and Perlin was incorporated into the design of a new shading based language called the RenderMan Shading Language (RMSL). In the late 1980s the RMSL was developed by Hanarahan *et al* as a component of the Pixar's offline renderer PhotoRealisitic RenderMan [88, 89]. The RMSL is based loosely on the C language in that it includes loops, conditionals and functions but also has many built in functions specifically for shading and lighting calculations [90]. After being extended in the late 1990 the RMSL is the most widely used language for offline rendering to create photorealism in movies.

Research began into programmable graphics hardware in the 1990s at the University of Carolina. In 1998 Olano *et al.* introduced the new PixelFlow Architecture and PFMan a new hardware amenable-shading language [91]. The PixelFlow system was capable of rendering complex scenes with procedural shading at least 30 frames per second. Due to being too expensive PixelFlow failed to succeed commercially. In 2001 Peercy *et al* at Silicon Graphics worked on interactive programmable shading. They treated the OpenGL architecture as a general SIMD computer and the compiler translated RMSL into commands for OpenGL [92].

The release of second and third-generation GPU's with directly programmable vertex and fragment processors allowed researchers at the Stanford University to begin creating a GPU specific shading language. Proudfoot *et al* developed the real time procedural shading language called the Stanford Real Time Shader Language (RTSL) [93]. In 2001 a collaboration between NVIDIA and Microsoft further developed the RTSL resulting in the creation of the Cg programming language. NVIDIA calls the language 'C for Graphics' (Cg) [94, 95] and Microsoft call their implementation High-Level Shader Language (HLSL). Cg and HLSL are the same language with identical syntax and semantics, the separate names are to allow for the distinct differences in underlying technology. Cg is an independent top layer which integrates fully with the two main graphics APIs, Open Graphics Library (OpenGL) and DirectX, whereas HLSL is a component of the DirectX framework. The third main language that uses real time shaders is the OpenGL shading language (GLSL) [96], which was developed by 3Dlabs and the OpenGL Architecture Review Board to become a part of the OpenGL Standard.

6.3.2 General Purpose

Shading languages are programming languages focused on the improvement of computer graphics, which require extensive knowledge of the latest graphics APIs as well as an understanding of the hardware. Often developers are put off using them for general purpose programming because they require time and money to learn and this hinders their acceptance of GPUs for HPC. The need to harness the power from the highly parallel GPU architecture has lead to the development of languages that focus on general purpose computation.

6.3.2.1 Brook for GPUs

Brook for GPUs was developed at Stanford University Graphics Lab by Buck *et al* [97] and was the first general purpose language. It was developed as a language for Stanford's Merrimac streaming computer and IMAGINE processor and later adapted to the capabilities of the graphics hardware. Brook extends the C language to include simple data-parallel constructs and enables the use of the GPU as a streaming coprocessor. Streams contain a collection of elements which are all changed by the same operations in parallel. Brook kernels are special functions that operate on the streams. Brook has been used as part of the Folding@home project [98], which investigates the serious consequences that can occur when proteins in the human body assemble or fold incorrectly.

6.3.2.2 Close to the Metal

Close to the Metal (CTM) was introduced by ATI in late 2006 [99]. The platform is a Data Parallel Virtual Machine that hides the graphics components and exposes the GPU as a data parallel processor array and memory controller, fed by a simple command processor. The DPVM allows generation of the code which is device dependent and program execution that is ideally device independent. CTM sees the GPU as three major components: a command processor, a memory controller and a data parallel processor array.

6.3.2.3 AMD Stream SDK

AMD's software development kit, the AMD Stream SDK, is available to download for free and works with all Firestream and all AMD GPUs after the Radeon R600 [100]. Figure 6.3 show the Stream computing model. Performance libraries are available and of interest are the AMD core math library which includes full BLAS and some LAPACK along with fast fourier transform and random number generators. The compilers are Brook+, an extension of Brook, and RapidMind a complete development environment. AMD compute abstraction layer (CAL), an extension of CTM, is a lower level driver and programming language which provides a low level access for performance tuning.

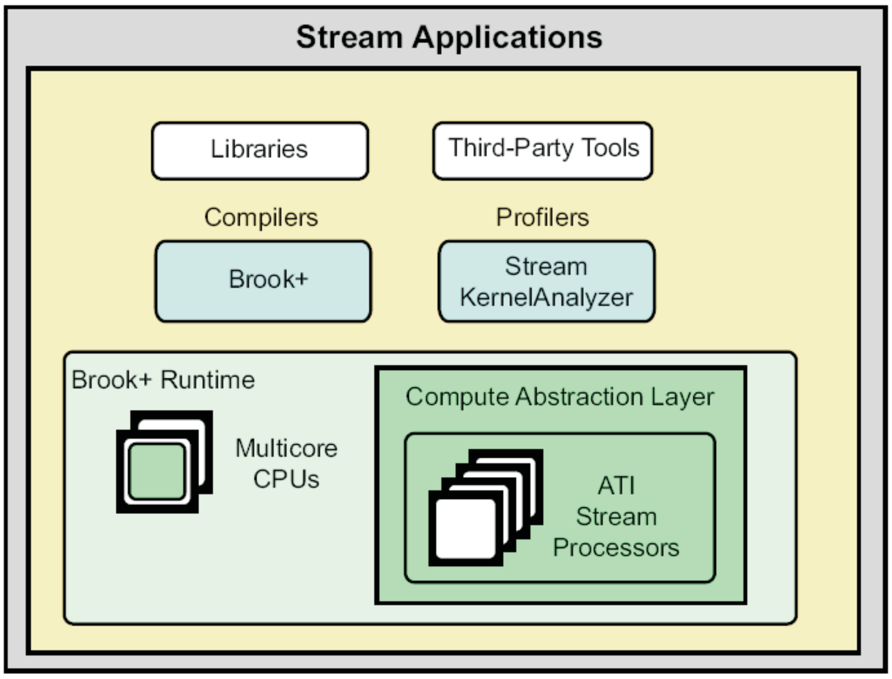


FIGURE 6.3: Stream computing model.

6.3.2.4 Compute Unified Device Architecture SDK

Compute Unified Device Architecture (CUDA) was introduced by NVIDIA also at the very end of 2006 and is a hardware and software architecture for carrying out computation on the GPU as a data parallel computing device without the need of mapping to the graphics API [101]. The CUDA Toolkit is available to download for free. The software stack is shown in figure 6.4. CUDA offers both gather and scatter memory operations and has on chip shared memory which means more programming flexibility. CUDA has two maths libraries: CUBLAS is an implementation of BLAS for the GPU and CUFFT is a fast fourier transform library.

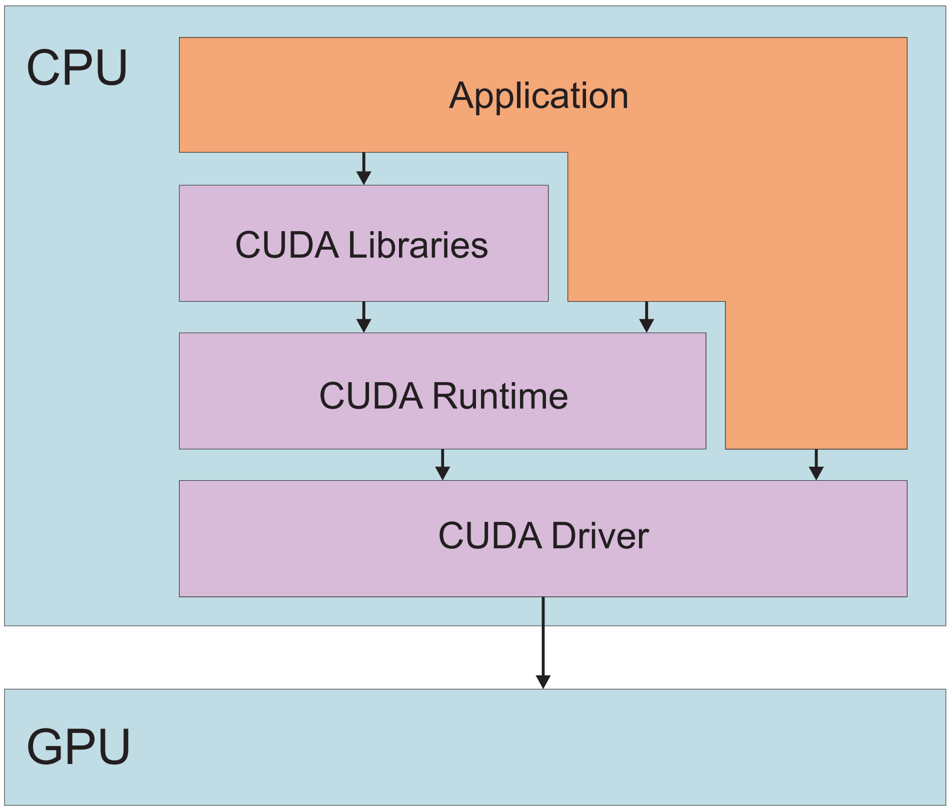


FIGURE 6.4: CUDA software stack.

6.4 Newly Released

The work for this thesis began in October 2006 when the Cg was the newest available programming language being used to experiment with GPGPUs. The GPU industry has moved a long way in just three years e.g. NVIDIA have released three brand new architectures in this time the: G80, GT200 and 'Fermi'. This section reviews some of the exciting new advances in GPUs that have happened in the last months of my research, which due to time constraints, it was not possible to incorporate into the algorithms developed in this thesis.

6.4.1 Open Compute Language

Open Computer Language (OpenCL) is a framework for writing heterogenous programs using multiple processors including CPUs and GPUs [102]. The first public demonstration was in December 2008, but Apple did not make a full release of the OpenCL implementation until the end of August 2009 with NVIDIA following suit in September 2009. AMD also supports the OpenCL standard and has included some OpenCL development tools in the ATI Stream SDK v2.0 Beta Program. The introduction of OpenCL is the first step towards becoming hardware independent when developing on GPUs.

6.4.2 Libraries

Released by AccelerEyes, the ‘Jacket’ Engine for MATLAB enables standard MATLAB code to run on any NVIDIA CUDA-capable GPU [103]. Jacket works by introducing new data types to MATLAB which lets the user move data and computations to the GPU. Tech-X Corporation’s GPU-Lib is a library of mathematical functions that provides bindings for a number of Very High-Level Languages including MATLAB and Python [104]. For both Jacket and GPU-Lib no knowledge of GPU programming or memory management is required and can accelerate new applications or be incorporated into existing applications with minimal effort.

EM Photonics in partnership with NVIDIA in August 2009 released CULA an implementation of the industry-standard LAPACK linear algebra library designed and optimised for CUDA enabled GPUs [105]. All of these libraries are commercial software products which must be purchased, but back up the findings in this thesis that numerical libraries for GPUs need to be developed before engineers can fully harness the power of them in applications.

6.4.3 Architectures

Announced October 2009 Fermi is NVIDIA’s newest GPU architecture [106]. The new Fermi GPUs will be so powerful that they are being termed computational graphics processing units (CGPU). There are many new features of the Fermi architecture, the most relevant to the development of new numerical algorithms will now be reviewed. This architecture has over 4 times more CUDA cores and 8 times the peak double precision floating point performance over GT200. The new Parallel DataCache hierarchy with configurable L1 and unified L2 caches means faster performance as memory sharing will be more efficient. The new Error-Correcting Code (ECC) memory support means that users will no longer need to worry about repeating calculations in case of corrupt memory. The GigaThread Engine enables concurrent kernel execution and out-of-order thread block execution, which means that parts of the device will no longer be redundant, as if one kernel does not fill all the CUDA cores another kernel can be launched to utilise the remaining cores.

6.4.4 Visual Deployment Environment

Due for limited Beta release in October 2009, Nexus brings GPU Computing into Visual Studio 2008 [107]. The key features of Nexus for GPU code development are the Analyzer and Debugger. The debugger supports debugging of CUDA C and HLSL source code transparently on the GPU hardware. This means that the debugger is parallel aware and can be used to debug and look at variables on individual threads. Source and Data break points mean that the code can be stopped at any point in the run time. The Analyzer allows for the viewing of activities and events across your CPU and GPU on a single, correlated timeline. The introduction of thread debugging and the event timeline will make developing heterogenous code a lot easier.

6.5 Analysis of Algorithm for Implementation

The algorithm developed on the GPU uses the MLSFM developed for the elliptic Helmholtz equation. But it is intended to be used as a stepping stone towards the acceleration of the photonic meshless methods. To analyse the algorithm it is split into two problems, the first being create the nodes and use them to fill the matrices and the second solving the generalised eigenvalue problem. Table 6.2 shows the time taken by the photonic MLSFM and MLWFM. From this table the motivation for using a GPU can be seen as large amounts of time is taken in the matrix fills which are a highly parallelisable operation. The result in the table are produced with the Wendland's C4 functions and the minimum number of nodes and background cells required to give the correct results. The machine used had a 6600 Intel Core 2 duo processor (2.4 GHz) and 4 GB of RAM.

Method	Matrix Fills(s)	Total(s)
Photonic MLSFM	94(20%)	468
Photonic MLWFM	266654(99%)	268347

Table 6.2: Time taken to complete the two parts of the meshless methods.

6.6 CUDA

6.6.1 CUDA Programming Model

When used with CUDA the GPU (device) operates as a coprocessor to the CPU (host). The GPU has a large number of threads that execute in parallel and is very good at carrying out the same set of instructions independently on different data. A kernel is a program which runs on the GPU and carries out a set of instructions. Both the host and the device have their own DRAM memory between which data can be copied but it is computationally expensive. A kernel is executed by a batch of threads that are organised into a grid of thread blocks. A thread block is a batch of threads that can easily work together as they can share data though their fast shared memory and can be made to synchronise with each other by setting a synchronisation point. A thread ID is used to identify individual threads within a block. The size of a thread block is specified by the application when calling the kernel, a block can be a two- or three-dimensional array. The maximum number of threads in a thread block is 256, but more threads can be used to execute the same kernel by grouping together blocks of threads to form a grid of blocks. The downside to this is that threads in different blocks in the same grid cannot communicate or synchronise. Like the threads, each block has its own block ID. Figure 6.5 illustrates how the threads are batched.

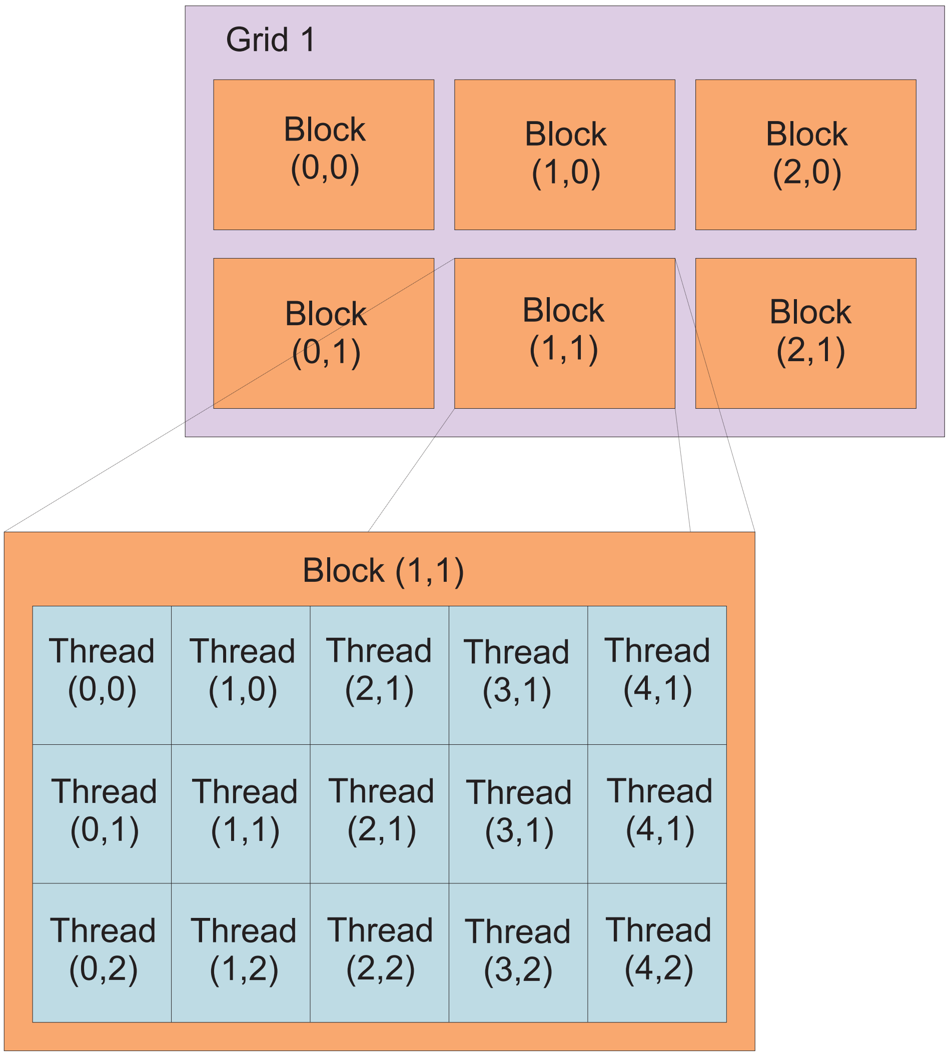


FIGURE 6.5: Thread batching model.

6.6.2 Hardware

The device is a set of Single Instruction Multiple Data (SIMD) multiprocessors. Each of the multiprocessors has four types of on-chip memory; one set of local 32-bit registers, a shared, a read only constant cache and a read-only. The local and global memory spaces are implemented as read-write regions of device memory. Scheduling is used to execute a grid of thread blocks on the multiprocessors. Batches of blocks are processed sequentially by each multiprocessor. In order to achieve quick memory access each block is processed by only one multiprocessor. The number of blocks each multiprocessor can execute in one batch depends on how much shared memory is needed per block and how many registers are required per thread. A kernel will not be able to launch if there are not enough resisters or shared memory for each multiprocessor to be able to process at least one block. Active blocks are blocks that are being processed by one multiprocessor in one batch. A warp is a group of SIMD threads that every active block is split into. All warps contain the same number of threads, which is known as the warp size. A thread scheduler is used to maximise the use of the computational resources of the multiprocessor by switching betweens active warps.

6.6.3 Meshless Method

The meshless method formulated in chapter 4 for the elliptic Helmholtz equation was divided into three steps in order to simplify programming. Step one involves the generation of the uniform grid points, step two involves filling the L and G matrices and step three requires the generalised eigenvalue problem to be solved. Porting steps one and two to the GPU were simple tasks because they are easily implemented, scalable and fast parallel codes that map well the GPU. The only input required for these steps is how big the user requires the resultant matrices to be. The host code then generates the correct number of threads and blocks for the execution configuration. There are two kernels, one for step one and one for step two. The GPU used in this section was the NVIDIA GeForce 8600 GT.

Program 6.1 shows the kernel that generates uniform points. Each thread creates a line of points in the uniform grid. bx and by are the block indices, tx and ty are the thread indices and WG is the order required for the G matrix that was specified by the user. $SIZE$ is set in the code to 16, which will build an optimum block with 256 threads. The outputs X and Y are the co-ordinates of the points. *__global__* is a function type qualifier and declares the function as being a kernel, so it is callable only from the host but executed on the device. This is a good algorithm for the GPU architecture because the threads do not require any communication with each other, hence there is no need for synchronisation points. Each thread performs different calculations so there is no wasted compute time recalculating the same values. In addition, each thread does not require read or write access to the same data, so race conditions will not occur.

Program 6.2 shows the kernel that fills the L and G matrices. After the points are created by the first kernel they are then left on the device to be read by the second kernel, so no time is wasted with copying to the host and then back to the device.

```
__global__ void create_mesh_points(float* X, float* Y)
{
    // Block index
    int bx = blockIdx.x;
    //int by = blockIdx.y;

    // Thread index
    int tx = threadIdx.x;
    int ty = threadIdx.y;

    float wg = WG;
    float point_length_float;
    float dis;
    int point_length_int;
    int number = 0;

    point_length_float = sqrt(wg);
    point_length_int = sqrt((float)wg);
    dis = 1/point_length_float;

    for (int i=0;i<point_length_float; i++)
    {
         number = i + (tx*point_length_int) +
                (ty*point_length_int*SIZE) +
                (SIZE*SIZE*point_length_int*bx);

         if (number < HG)
         {
            X[number] = i*dis;
            Y[number] = ((tx+1)*dis) +
                        (bx*SIZE*SIZE*dis) +
                        (SIZE*dis*ty);
         }
    }
}
```

Listing 6.1: A CUDA kernel for creating uniform grid points.

```
__global__ void fill_G_L(float* X, float* Y, float* G, float* L)
{
    // Block index
    int by = blockIdx.y;
    // Thread index
    int tx = threadIdx.x, ty = threadIdx.y;
    float c = 0.5, dx, dy;
    float r, Xmax = 1.0, Ymax = 1.0;
    float x_diff, y_diff, x ;
    float xi, y, yi;

    for (int i=0; i<HG; i++)
    {
        int stop = i+(ty*HG)+(SIZE*HG*tx)+(by*HG*SIZE*SIZE);
        if (stop < HG*HG)
        {
            x = X[ty+(tx*SIZE) +(SIZE*SIZE*by)];
            xi = X[i];
            y = Y[ty+(tx*SIZE )+(SIZE*SIZE*by)];
            yi = Y[i];
            x_diff = fabsf(x-xi);
            y_diff = fabsf(y-yi);

            if (x_diff < (Xmax - x_diff))
                dx = x_diff;
            else
                dx = Xmax - x_diff;

            if (y_diff < (Ymax - y_diff))
                dy = y_diff;
            else
                dy = Ymax - y_diff;

            r = sqrt((dx*dx)+(dy*dy));

            if (fabsf(r/c) >= 1)
            {
                G[i+(ty*HG)+(SIZE*HG*tx)+(by*HG*SIZE*SIZE)]=0;
                L[i+(ty*HG)+(SIZE*HG*tx)+(by*HG*SIZE*SIZE)]=0;
            }
            else
            {
               G[i+(ty*HG)+(SIZE*HG*tx)+(by*HG*SIZE*SIZE)] =
                    ((1-r/c)*(1-r/c)*(1-r/c)*(1-r/c)) *
                    (1 +4*r/c);

               L[i+(ty*HG)+(SIZE*HG*tx)+(by*HG*SIZE*SIZE)] =
                    (-20/(c*c*c*c*c))*(c-r)*(c-r) *
                    ((2*c) - (5*r));
            }
        }
    }
}
```

Listing 6.2: A CUDA kernel for filling the L and G matrices.

6.6.4 Results

Figure 6.6 compares the time taken to fill the matrices using the GPU and MATLAB, using Wendland's C2 function. The figure shows that CUDA is orders of magnitude faster. The GPU code also scales better populating large matrices. The time taken to fill the matrices of order 225×225 is two orders of magnitude smaller, whereas the time taken to fill the matrices of order 4900×4900 is four orders of magnitude smaller. Step

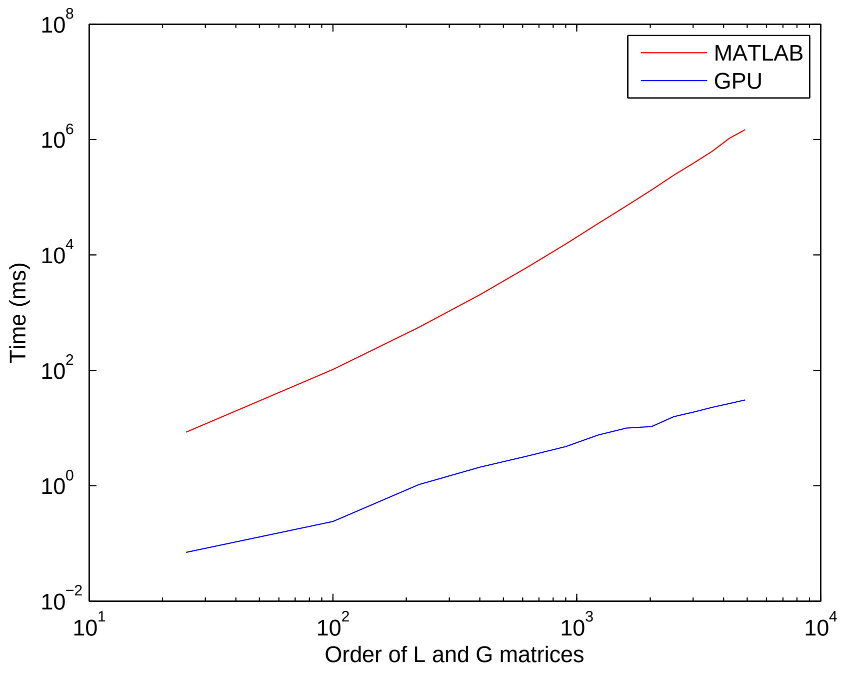


FIGURE 6.6: Comparison between the time taken to fill different sizes of the L and G matrices using MATLAB and CUDA.

three of the meshless method is solving the generalised eigenvalue problem:

$$L\alpha = -\gamma^2 G\alpha, \tag{6.1}$$

This is done in the sequential MATLAB code using the *eigs()* function or a subspace iteration algorithm. Although CUDA has a basic BLAS library it does not have the full function library one would expect to use in C or MATLAB, hence most functions that need to be used had to be written by hand for CUDA. The subspace iteration algorithm has significant communication overheads and does not easily lend itself to the CUDA architecture. Attempts were made to break the algorithm into simpler sequential and parallelisable parts. One way was to write the code in C and then make calls to the CUBLAS library. Also rewriting Cholesky, LU and QR algorithms by taking the LAPACK routines and replacing BLAS with CUBLAS. These codes gave the correct eigenvalues but significant communication overheads occurred leading to overall poor performance.

Building a CPU/GPU heterogenous piece of code meant that the GPU architecture could be utilised for the steps which mapped well to it and then the CPU could be used to solve the eigenvalue problem by calling a CLAPACK routine, *dsygv_*. Figure 6.7 shows the speed up given by the CPU/GPU heterogenous code when compared with the MATLAB code. The heterogenous code is an order of magnitude faster than the MATLAB code.

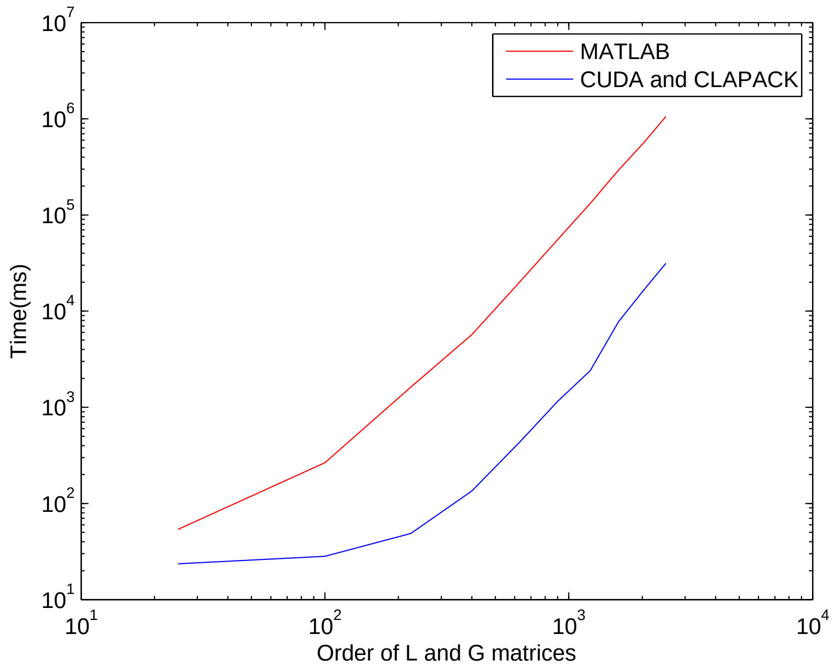


FIGURE 6.7: Comparison between the time taken to complete the meshless method using MATLAB and CUDA with CLAPACK.

6.7 AMD Stream SDK

6.7.1 Brook+ Programming Model

Stream computing uses a virtualised SIMD programming model that operates on arrays of data elements known as streams. Input streams are used to execute stream kernels (user defined programmes) to generate output streams. Each instance of a kernel running on a thread processor inside the SIMD engine is know as a *thread*. The *domain of execution* is a region of the output buffer to which threads are mapped. Threads are scheduled by the stream processor onto the thread processors until they have all been executed.

6.7.2 Hardware

The ATI device is a set of SIMD multiprocessors, which each contain multiple thread processors. Each thread processor contains multiple programmable computational units knows as stream cores and a branch execution unit. The thread processor is arranged as a 5 way very long instruction word (VLIW) processor, such that up to 5 scalar operations can be carried out simultaneously. One of the stream cores is also capable of carrying out transcendental operations. In order to do double precision floating point operations the thread processor must use all of the other 4 stream cores. Every thread processor within a SIMD multiprocessor operates on the same instruction set for a cycle. However, to reduce latency multiple threads are interleaved e.g. in one thread processor 4 threads can issue 4 VLIW instructions over 4 cycles. 'Wavefront' is used to describe a group of threads that are executed together, the size of which varies depending on the stream processor.

6.7.3 Meshless Method

6.7.3.1 Creating Nodes and Filling Matrices

As discussed when implementing the CUDA algorithm creating the nodes and filling the matrices was easy to port to the GPU as they are easily implemented, scalable and fast parallel codes that map well the GPU architecture. Figure 6.3 shows the stream kernel used to fill the L and G matrices. This code is shown to highlight the main differences between the NVIDIA and AMD implementations. The function has five parameters: a value type and four stream references. The *out* prefix on the L and G streams marks them as output streams that can only be written too. The output streams are used by the GPU to determine the domain of the problem. The [] brackets on the X and Y streams mean that the kernel can access any of the elements in the stream. The <> brackets on L and G streams denote that the kernel can only access the array element specified by the *instance* method. The *Float4* data type has four fields x, y, z and w, on which four calculation may be performed concurrently per thread.

6.7.3.2 Eigenvalue Algorithm Breakdown

A subspace iteration (SI) algorithm was chosen as it is capable of finding a set of the smallest eigenvalues, which is what was required in the meshless method. As discussed in the previous section (6.6) solving the generalised eigenvalue problem has caused some implementation challenges. Traditional CPU implementations cannot easily be ported to the GPU architecture because the latter offers only a subset of the features available on a CPU e.g. limited standard libraries and a restrictive memory model. A purely homogenous GPU solution would be difficult to optimise for performance because the

```
kernel void Float4FillMatrices(
    int4 Width, float X[][], float Y[][],
    out float4 L<>, out float4 G<>
    )
{
    int2 vPos = instance().xy;
    int vPosx = vPos.x * 4; //adjust for float4
    int vPosy = vPos.y;
    int4 vPosx4 = int4(vPosx, vPosx + 1, vPosx + 2, vPosx + 3);
    int4 vPosy4 = int4(vPosy, vPosy, vPosy, vPosy);
    float4 g, l, dx, dy, r;
    float4 Xmax = float4(1.0f, 1.0f, 1.0f, 1.0f);
    float4 Ymax = float4(1.0f, 1.0f, 1.0f, 1.0f);
    float4 x_diff, y_diff, x, xi, y,yi;
    int4 xyx, xyy, xyix, xyiy;

    xyy = vPosy4/Width; //int division floors answer
    xyx = vPosy4 - (Width*xyy);
    xyiy = vPosx4/Width; //int division floors answer
    xyix = vPosx4 - (Width*xyiy);
    x = float4(X[xyx.x][xyy.x], X[xyx.y][xyy.y],
               X[xyx.z][xyy.z], X[xyx.w][xyy.w]);
    y = float4(Y[xyx.x][xyy.x], Y[xyx.y][xyy.y],
               Y[xyx.z][xyy.z], Y[xyx.w][xyy.w]);
    xi = float4(X[xyix.x][xyiy.x], X[xyix.y][xyiy.y],
                X[xyix.z][xyiy.z], X[xyix.w][xyiy.w]);
    yi = float4(Y[xyix.x][xyiy.x], Y[xyix.y][xyiy.y],
                Y[xyix.z][xyiy.z], Y[xyix.w][xyiy.w]);

    x_diff = abs(x-xi);
    y_diff = abs(y-yi);
    dx = min(x_diff, Xmax - x_diff);
    dy = min(y_diff, Ymax - y_diff);
    r = sqrt(dx*dx + dy*dy);
    dx = float4(1.0f,1.0f,1.0f,1.0f) - (r + r); // == 1.0f-(r*1/0.5)
    dx = dx * dx; // == (1.0f-r/c)^2
    dx = dx * dx; // == (1.0f-r/c)^4
    g = dx * (float4(1.0f, 1.0f, 1.0f, 1.0f) +
        float4(8.0f, 8.0f, 8.0f, 8.0f)*r); // == dx*(1.0f+4.0f*r/c)

    dx = float4(0.5f, 0.5f, 0.5f, 0.5f) - r;

    l = float4(-640.0f, -640.0f, -640.0f, -640.0f) * dx * dx *
        (float4(1.0f,1.0f,1.0f,1.0f) - float4(5.0f,5.0f,5.0f,5.0f)*r);

    if (r.x >= 0.5f) { g.x = 0.0f; l.x = 0.0f; }
    if (r.y >= 0.5f) { g.y = 0.0f; l.y = 0.0f; }
    if (r.z >= 0.5f) { g.z = 0.0f; l.z = 0.0f; }
    if (r.w >= 0.5f) { g.w = 0.0f; l.w = 0.0f; }

    G = g; L = l;
}
```

Listing 6.3: A Stream kernel for filling the L and G matrices.

algorithm's tasks have significant communication and synchronisation requirements. The strategy implemented in this section is to split the algorithm into tasks better suited to CPU or GPU architecture, to: improve performance, ease implementation and allow modularisation of the code. Table 6.3 shows a break down of the SI algorithm, it shows matrix sizes that would give accurate answers. The LU decomposition was carried out on the host as it only needed to be done once. Eventually this can be swapped to a device job when a kernel is written as this will avoid a device-host copy. Matrix multiplication is done on the device as it is simple to implement and well suited. The backwards and forwards substitutions are carried out on the device to avoid copying. Again the matrix multiplications are carried out on the device. The next two steps are carried out on the host as the data sizes are very small. Finally the last step is carried out on the GPU so that w is updated for the next loop.

SI Step	Input Matrices	Output Matrices	CPU or GPU

LU Decomposition	G = 1600x1600	l = 1600x1600,
		u = 1600x1600
			CPU

Matrix Multiplication	L = 1600x1600
	v0 = 1600x16
		w = 1600x16	GPU

While Loop Starts			

Forward Substitution	l = 1600x1600
	w = 1600x16
		y = 1600x16	GPU

Back Substitution	u = 1600x1600
	y = 1600x16
		v = 1600x16	GPU

3 Matrix
Multiplications
	v' = 16x1600
	w = 1600x16
	L = 1600x1600
	v = 1600x16
	v' = 16x1600
	w = 1600x16
		e = 16x16
		
		w = 1600x16
		
		a = 16x16
			CPU

c = a\e	a = 16x16 (copy to host)
	e = 16x16 (copy to host)
		c = 16x16	CPU

Find Eigenvalues
and Eigenvectors
	c = 16 x 16	d = 16x1
		v = 16x16
			CPU

Matrix Multiplication
and
w = w./norm(w)
	w = 1600x16
	v = 16x16 (copy to device)
		w= 1600x16	GPU

Table 6.3: Subspace iteration algorithm breakdown.

6.7.3.3 Stream Implementation

The program was developed in Visual Studio using C++, Brook+ and the MATLAB Engine application programming interface (API). There were a number of programming challenges that needed to be addressed. Firstly, the algorithm is complicated and spilt into many parts which meant that errors could be difficult to find and debug. Secondly, Brook+ and C++ uses zero-based array indexing whilst MATLAB uses one-based array indexing. In addition, MATLAB uses column-major ordering for multi-dimensional arrays whilst Brook+ and C++ use row-major ordering. Thirdly, Brook+ offers no built-in mathematical matrix methods. Fourthly, there was no GPU support for complex numbers. The solution to these challenges was to break down the program into a number of helper classes that provided functions to simplify the implementation of the meshless method.

An interface was created to define the methods that were needed to be performed on the matrices e.g. reading/writing between the GPU and CPU, setting values on a GPU matrix, returning the size of a GPU matrix. An implementation of the interface was written as a base class that encapsulated stream specific code for creating, accessing and destroying streams. Four sub-classes were then created from this base class for the data types that were needed for the program: *float*, *float2*, *float4* and *double*. Each sub-class implemented the following functions: sequence equality, constant fill, random fill, matrix multiplied by a scalar, matrix multiplied by a matrix, forward substitution, forward substitution with pivoting, and backwards substitution. The forwards and backwards substitution required synchronisation e.g to be able to read the outputs created by earlier loops of the code. Brook+ only allows scatter/gather write to the output stream and not read so a buffer had to be written to duplicate the values stored in the output stream. Also this function used the two Kernel Interface functions *domainOffset* and *domainSize* to constrain the domain of execution. The *float2* sub-class contains support for complex matrices. The *real* part of the number is stored in the *.x* field of the underlying data type and the *imaginary* part in the *.y* field. Two classes were written that provided support for MATLAB. The MATLAB matrix class had functions for copying to MATLAB from the buffer and to the GPU from the buffer. Starting in MATLAB the matrix must be copied to the CPU and then copied to the GPU where a kernel manipulates it into the correct order. The MATLAB engine class insures that the matrices are copied to and from the correct instance of MATLAB. Unit tests were created so that each component of the implementation could be verified in isolation.

6.7.3.4 Results

The timing in the section was done using performance counters and the GPU used was a Radeon HD 4870 X2. Figure 6.8 shows the comparison between the time taken to create the nodes and fill different sizes of the L and G matrices using the CPU with C++ and the GPU with Brook+. The size of the matrices vary from 144x144 to 2304x2304 as this is the range generally used to the method to get rough to suitably accurate answers. The red line in the figure shows that the time taken by the CPU increases as the number of elements in the matrices increases. The blue line in the figure shows that the time taken to fill the matrices on the GPU stays the same as the matrices increase in size. This is because the kernel on the GPU is embarrassingly parallel. The kernel would eventually hit a limit and the time start to increase when it requires more thread processors than the GPU has available.

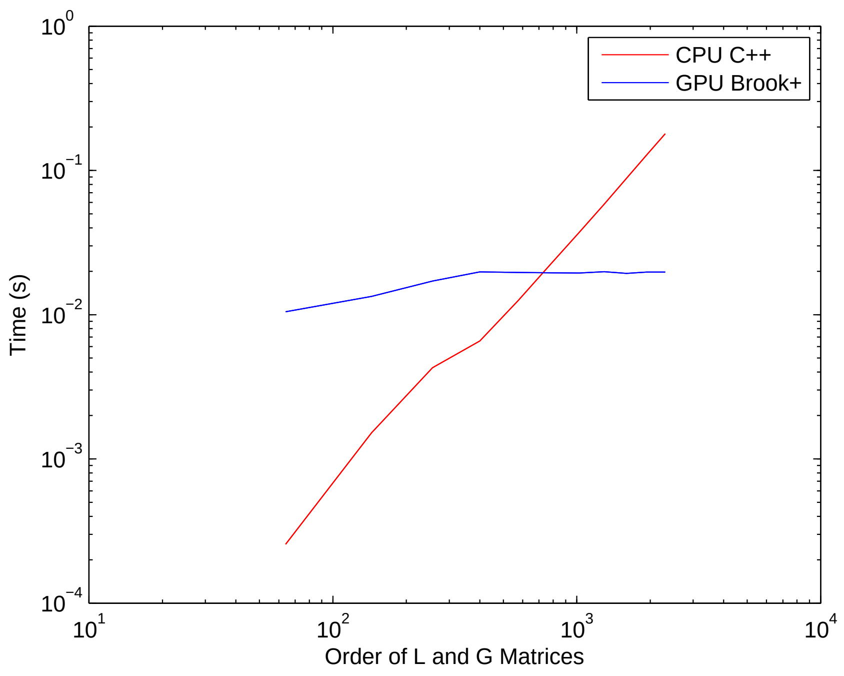


FIGURE 6.8: Comparison between the time taken to create the nodes and fill different sizes of the L and G matrices using C++ on the CPU and Brook+ on the GPU.

A program was written that created the nodes and filled the matrices using C++ and solved the generalised eigenvalue problem using CLAPACK. This program could then be used to gain a more accurate comparison between times taken to complete the meshless method. Another program was written that had Brook+ create the nodes and fill the matrices. The matrices were then copied to the CPU and CLAPACK used to find the eigenvalues. Figure 6.9 shows the comparison between the time taken to complete the meshless method using C++ and CLAPACK, Brook+ and CLAPACK and the heterogeneous implementation with SI. The the red line and the blue line shows the implementations that use the CLAPACK eigenvalue solve. These implementations highlight that the largest amount of time in this meshless method is spent in the eigenvalue solve. The green line on the figure shows the time taken by the GPU creating the nodes and filling the matrices and then using the heterogenous SI algorithm to solve the eigenvalues. The SI implementation takes the most time due to significant overheads. As previously discussed the copying between MATLAB and the GPU involved reordering the arrays and providing the complex support, which were expensive operations.

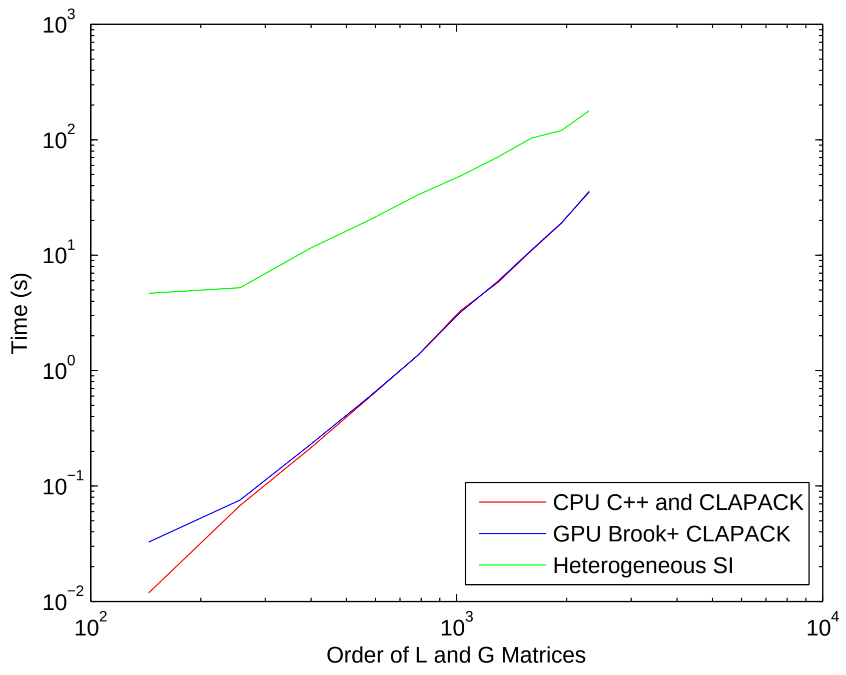


FIGURE 6.9: Comparison between the time taken to complete the meshless method using C++ and CLAPACK, Brook+ and CLAPACK and the heterogeneous implementation with SI.

6.8 Conclusion

In this chapter a meshless method that was accelerated in parts by a GPU was presented. The CUDA implementation showed a significant speed up when compared to the MATLAB version of the meshless method. The Stream implementation was tested against a C++ version and for the range of values tested increasing the size of the matrix did not increase the time taken for the GPU to fill the matrix. The heterogenous SI algorithm had significant copy overheads which resulted in it being slower than the CLAPACK solver. Use of the GPU as an alternative processor to the CPU has a way to go before it will be easily adopted for use in engineering problems. The low-level programming required and knowledge of the complicated underlying hardware is a steep learning curve. The Stream and CUDA programming models are different and each require specialist knowledge, this generally means programmers will be locked to one hardware. Also, there are not very many standard libraries as such time is often spent implementing functions that the user would expect as standard on the CPU. Doing general purpose computation on GPUs is in the early stages of development. Promising additions for 2010/2011 are more high-level libraries, better architectures (Fermi), debugging environments and OpenCl, this will make GPU programming more inviting for engineers.

Chapter 7

Conclusion

7.1 Summary

This summary looks at how the objectives of the thesis have been met.

7.1.1 Modelling two-dimensional PhCs

Chapter 1 introduces PhCs and the motivation to model them. As discussed in the chapter PhCs, prevent the propagation of certain wave lengths of electromagnetic radiation. Two-dimensional PhCs are already being used commercially for increasing light extraction from LEDs, and there are many other things they could be used for if they can be tailored to have the required properties e.g. larger gaps, high order gaps, sets of gaps or flatter gaps. Fabrication is currently limiting the complexity of PhCs but it can be seen from nature that very complex structures can have interesting effects on light. With the many potential applications of PhCs, including lasers, optical circuits (computers) and optical communications, there is large motivation to have powerful modelling algorithms that can deal with complex novel crystal structures.

Modelling PhCs involves solving Maxwell's equations when they are in the form of a generalised eigenvalue problem. As we concentrated on two-dimensional PhCs there are two distinct modes, the TE and the TM, each with its own generalised eigenvalue problem. There are various methods used in the literature to solve the problem. The standard method being the PWEM but this is slow to converge due to large dense matrices. The FEM is also used but this requires the creation of mesh which is a slow and difficult process in order to gain good accuracy. The aim of writing our own PhC algorithms was to produce code specifically for the analysis and optimisation of PBGs.

7.1.2 Finite Difference Method

After reviewing PhC theory and terminology in the first chapter, a FDM is introduced in chapter 2 as a way to solve the two generalised eigenvalue problems. The FDM was chosen as it creates large sparse matrices and is simple to implement with small amounts of code which is the opposite of the FEM that has large dense matrices and requires large amounts of coding. The formulation of the method based on a FD grid is shown and the matrices of the generalised eigenvalue problem are assembled. Once the eigenvalues are found they are presented as a band diagram so that any PBGs can be clearly seen. The method is used to model PhCs to show the crystal structures preferred by first the TM mode and then the TE mode. The TM mode favours regions of isolated high dielectric material whereas the TE mode favours a connected lattice of high dielectric material. The results produced by the FDM for both structures are in good agreement with the standard results in the literature produced by the PWEM. To illustrate the ability of the FDM to model novel structures in the final part of chapter the Superformula was introduced as a function that can produce many different shapes. The Superformula was chosen as it can create concave polygons with sharp angled corners. The results from the FDM for the novel structure show good agrement with the results produced by the commercial software BandSolve. The highlight of chapter 2 is the use of the FDM with curved surfaces.

7.1.3 Novel Shaped Structures

Chapter 2 validated the FDM as an accurate algorithm for use when modelling two-dimensional PhCs, which meant it could then be used in an optimisation process in chapter 3. At present most PhCs have simple cylindrical rods in air or solid substrates with air cylinders drilled into them. An experiment was done using the FDM to review the effect, on the largest PBG, of changing the radius and dielectric of a solid rod in air. The experiment was then repeated but this time the dielectric of the substrate and the radius of the air cylinder was varied. Solid rods with a radius of $0.2a$ and a dielectric of 13 surrounded by air produced the largest gap-midgap ratio of 41%. These experiments were carried out so that the results from them could be compared with the outputs from the optimisation.

The main aim of the parameterisation was to concentrate on getting whole novel shapes for the solid or ‘air’ rods. The first parameterisation used a NURBS curve with 8 control points and a scale factor. The second parameterisation placed 4 Hicks-Henne bump functions on a circle within which the bump functions were alternating. An important feature of both parameterisations was that they could make a circle. This was important because the experiments were trying to prove that there were better crystals structures than cylinders.

A GA was chosen as a suitable optimisation tool after reviewing how PhCs has been designed and optimised to date. The GA generated an initial population from which a pool of parents was chosen. The parents were then used to create the next generation either by being copied, mated or mutated. When a GA has run for enough generations it will converge on the best solution. There were 12 optimisation runs it total, 6 for the Hicks-Henne parameterisation and 6 for the NURBS parameterisation. For each set of 6 runs, 3 were run for solid high dielectric rods and the other 3 for solid high dielectric substrates with air holes. The 3 runs were broken down into finding the largest TM PBG, the largest TE PBG and the largest complete PBG. The Hicks-Henne parameterisation found the largest PBGs for every case and in all cases the novel shape parameterisations found better results that the cylindrical rods. The optimisation results consolidated the results produced by the FDM in the previous chapter. For the TM mode in both cases the optimisation produced 'shapes' that gave crystal structures with isolated regions of high dielectric material. For the TE mode in both case the optimisation found 'shapes' that produced structure that were near to connected lattices. To be able to find more interesting structures the parameterisation needed to be more complicated. But for the FDM to be able to discretise complex novel structures accurately, it would require a very fine FD grid. Unfortunately the fine grid would be computationally expensive and the GA would take a very long time to converge, which is not practical. The optimisation process worked well with the PhC modelling algorithms but the simpleness of the parameterisations prevented any new complex novel structures from being found. The highlight of chapter 3 is the use of the new parametrisation technique which uses distinct shapes and sharp edges.

7.1.4 Periodic Systems

Chapters 2 and 3 showed the motivation for developing a new algorithm for use in PhC modelling that can simulate more complex structures. In order to solve PhC problems with a meshless method there are two main steps. First, solve a PDE with periodic boundary condition and second, apply this to Maxwell's equations. Chapter 4 presents a solution to the first part of the problem. The PDE chosen to be modelled with periodic boundary conditions was the elliptic Helmholtz equation as it has analytical solutions. The chapter introduced RBFs as a technique for approximating functions. The formulation of a MLSFM is shown and the system matrices assembled for the generalised eigenvalue problem. The CSRBFs used in the method and their second derivatives are shown. CSRBFs were chosen for use in the method as they had a shape parameter which meant their domain could be limited to the unit cell. The domain of the system was made periodic by enforcing the relevant boundary conditions.

The analytical results of the elliptic Helmholtz equation are presented as they are used in the validation of the method. At the end of the chapter several computational experiments are carried out on the method. The experiments test the accuracy of the method when the different RBFs are used, as well as what effect varying the layout and number of nodes has and what effect varying the shape parameter has. The results found that the uniform grid point layout was the best for accuracy, with the Sobol second and random last. Using the C4 CSRBFs gives more accurate answers than the C2 CSRBFs. The best shape parameter to use for a 1 by 1 unit cell is $c = 0.5$. The experiments proved that the MLSFM works accurately and which are the best parameters to use with it. The key point from chapter 4 is that meshless methods can be used to solve PDEs with periodic boundary conditions.

7.1.5 Photonic Meshless Methods

Chapter 4 solves the first part of the problem to model PhCs with a meshless method, in chapter 5 the second part of the problem is solved when the meshless method is applied to Maxwell's equations. A MLSFM similar to the method in the previous chapter is formulated as a way to solve the simpler TM mode. The TE mode is more complicated due to discontinuities in the dielectric at the boundaries and cannot be solved with a MLSFM. The extra first partial derivatives required for the computation are presented. A MLWFM is formulated to solve the TE mode. This method can also be used to solve the TM mode but is not advised as it takes more computation time than the MLSFM. The integration in the MLWFM is approximated using a Gauss Quadrature method, with a 4 point rule.

The new meshless method algorithms are then validated against analytical solutions and results presented in the literature from the PWEM. The MLSFM showed good agreement with the analytical solutions found at the edges of the irreducible Brillouin zone. Both methods are then compared with the PWEM results for a square lattice of cylindrical rods in air. The rods have a dielectric of $\epsilon = 13$ and a radius $r = 0.2a$. Then they are compared to square array of dielectric ($\epsilon = 8.9$) veins in air ($\epsilon = 1.0$). The veins have a thickness $= 0.165a$. All of the methods use Wu's CSRBFs and a uniform grids of nodes. The MLSFM produces results for the TM mode that are in good agreement with the PWEM. The MLWFM produces results for the TM and TE modes that are in good agreement with the PWEM. The MLWFM was difficult to construct and therefore took time to complete. As such the method requires optimising before it can be used as viable modelling algorithm. The main problem with the method is the amount of time it takes to compute an accurate answer. However there are several improvements that can be made so that the method could be used to quickly and accurately model more complex structures and these are discussed in the future work section 7.2. The highlight of chapter 5 is the use of meshless methods to model two-dimensional PhCs.

7.1.6 Accelerator Technologies

In chapter 5 a new modelling algorithm was developed, but the method had bottle necks that slowed it down. Interest in using accelerator technologies to speed up the algorithm meant that in chapter 6 a GPU was used to accelerate the simpler MLSFM developed in chapter 4. This is the first step in porting the photonic meshless methods to the GPU. The chapter begins with an overview of parallel computing and a discussion on why GPUs were chosen over other devices. The chapter then reviews the history of languages for programming GPUs. The two main languages currently used are CUDA and Brook+/CAL. Interesting new releases for developing programs on GPUs are then discussed.

The CUDA SDK is then reviewed in more detail. The meshless method was split into two parts to implement on the GPU. The first part had to create the nodes and fill the matrices and the second part involved solving the generalised eigenvalue problem. Porting the first part to the GPU was a simple task because it maps well the GPU architecture. The kernels used for these tasks are discussed to highlight the CUDA language. Comparisons between the MATLAB code and the CUDA code are then presented. For just the matrix fills the GPU is two orders of magnitude faster for small matrices (225x225) and four orders of magnitude faster four large matrices (4900x4900). Difficulty in porting the eigenvalue solve resulted in a heterogeneous program being written. CLAPACK was used to solve the eigenvalue problem on the CPU. The time taken to complete the meshless method is an order of magnitude faster on the GPU.

The Stream SDK is then discussed in more detail. The aim on the stream was to try and write an eigenvalue solver. Again the meshless method was split into two parts. The kernel used to fill the matrices is then presented to highlight the differences between CUDA and Brook+. The SI algorithm is then analysed and broken down into smaller parts to allow for a heterogenous implementation. Mainly parts with large amounts of data were processed on the GPU and parts with small amounts of data on the CPU. The stream implementation required the creation of helper classes that had complex number support, higher order matrix support and support for copying between the host and device. The main meshless method class used C++, Brook+ and the API to call MATLAB. For comparison a C++ version of the code was written that used CLAPACK for the eigenvalue solve. Also a GPU version was written that used Brook+ for the first part and CLAPACK for the second part. The time taken to create the nodes and fill the matrices is then compared. The time taken on the CPU increases as the matrices get larger. On the GPU for the range of values tested the time remains constant due to the very parallel nature of the algorithm. Comparison between the time taken to complete the meshless method using C++ and CLAPACK, Brook+ and CLAPACK and the heterogeneous implementation with SI showed that the heterogenous implementation is slower. This is due to significant overheads when moving between the CPU and GPU:

reordering the arrays and providing the complex support. Moving parts of the code from MATLAB to C++ would help reduce the overheads. In chapter 6 the unique contribution is the use a of GPU to try and accelerate a meshless method.

7.2 Further Work

The difficulty in implementing the MLWFM has left scope for further work to be done so that the method can be considered a useful tool in modelling PhCs. The method needs to be tested like the MLSFM was in chapter 4 so that it is known how changing the number of nodes and the number of background cells will affect the accuracy. There are 3 main areas in which the method needs to be improved: accuracy, performance and capability.

7.2.1 Accuracy

In section 4.6 the figures showed that the C4 CSRBFs produced more accurate results. Equations 7.1 and 7.2 show Wendland's and Wu's C6 CSRBFs, using these in the meshless method would produce more accurate results.

$$\phi(r) = \left(1-\frac{r}{c}\right)^8 \left(1+8\frac{r}{c}+25\frac{r^2}{c^2}+32\frac{r^3}{c^3}\right) \tag{7.1}$$

$$\phi(r) = \left(1-\frac{r}{c}\right)^7 \left(5+35\frac{r}{c}+101\frac{r^2}{c^2}+147\frac{r^3}{c^3}+101\frac{r^4}{c^4}+35\frac{r^5}{c^5}+5\frac{r^6}{c^6}\right) \tag{7.2}$$

The integration in the MLWFM was solved using Gauss Quadrature with 4 Gauss points per back ground cell that had equal weightings. The nine point rule has different weightings for each of the points, with the middle point carrying the most weight. The extra points would give a better approximation over each background cell. Each of the background cells used in the method were all of uniform size. The method could be made more accurate by having many smaller background cells around the interesting features in the unit cell and then having larger background cells in the region of solid dielectric.

7.2.2 Performance

7.2.2.1 Algorithms

Varying the size of the support domains in the MLWFM will affect the speed but some work will need to be done to compare the loss in performance vs. the gain in speed.

A meshless weak strong form method is suggested as another way to speed up the PhC modelling process. In this type of method the more difficult to solve weak form equations are only used to solve the problem at the boundaries and the strong form equations are used everywhere else. This will reduce the amount of computation required.

7.2.2.2 GPU

In Chapter 6 the GPU was used to accelerate the simple meshless method. By taking modular code as a starting point it should be possible to gain a large speed up in the MLWFM by porting sections of it to the GPU. Care will need to be given to getting the complex matrices, produced by the photonic meshless methods, to work on the GPU as complex numbers are not currently supported. The new libraries and languages that have been released offer exciting possibilities in this area, especially the release of the open CL standard. NVIDIA have made a big effort in making GPU programming simpler. The new 'Fermi' architecture offers greater programming flexibility with the new memory hierarchic. The GigaThread scheduler threading will make it easier to implement a bigger range of algorithms. The 'Nexus' visual studio environment will make development and debugging much easier.

7.2.3 Capability

The MLWFM can currently only solve PhCs that have a square or rectangular primitive unit cell. The gauss quadrature used is for a quadrilateral so can be edited to work with a rhombus unit cell so that hexagonal lattices can be modelled.

7.3 Concluding Remarks

This section highlights the unique contributions of the thesis and shows how the overall aim has been met. The main aim of the thesis was to develop novel numerical algorithms for the efficient design of new PhC structures based on band diagram analysis. Two new modelling algorithms were written: a finite difference method and a meshless method. The new FDM method was written to model shapes with curved surfaces. Once validated the FDM was used to do photonic band gap optimisation with a unique and novel parametrisation technique that evolved using whole shapes. The meshless method was validated for solving periodic systems. Then a brand new meshless method was written for modelling two-dimensional PhCs. A GPU was used as an alternative processor to the CPU to speed up the filling of the large matrices required for the meshless method. This work hopefully provides practical contributions in the field of photonic crystal modelling to enable novel engineering applications.

Bibliography

[1] K. Sakoda. *Optical Properties of Photonic Crystals.* Springer, 2001.

[2] J. N. Winn, J. D. Joannopoulos, S. G. Johnson, and R. D. Meade. *Photonic crystals moulding the flow of light.* Princeton University Press, 2008.

[3] P. Vukusic and R. Sambles. Photonic structures in biology. *Nature*, 424:852–855, 2003.

[4] NVIDIA CUDA programming guide (1.1). http://www.nvidia.com/cuda/.

[5] E. E. Hart, S. J. Cox, K. Djidjeli, and V. O. Kubytskyi. Solving an eigenvalue problem with a periodic domain using radial basis functions. *Engineering Analysis with Boundary Elements*, 33(2):258 – 262, 2009.

[6] E. Yablonovitch. Inhibited spontaneous emission in solid-state physics and electronics. *Physical Review Letters*, 58(20):2059–2062, 1987.

[7] S. John. Strong localization of photons in certain disordered dielectric superlattices. *Physical Review Letters*, 58(23):2486–2489, 1987.

[8] F. Bloch. Über die quantenmechanik der elektronen in kristallgittern. *Zeitschrift für Physik A Hadrons and Nuclei*, 52:555–600, 1929.

[9] G. Floquet. Sur les équations différentieller linéaries à coefficients périodiques. *Annales Scientifiques de l'École Normale Supérieure*, 12:47–88, 1883.

[10] J. W. Strutt (Lord Rayleigh). On the maintenance of vibrations by forces of double frequency, and on the propagation of waves through a medium endowed with a periodic structure. *Philosophical Magazine*, 24:145–159, 1887.

[11] Y. Zeng, X. Chen, and W. Lu. Propagation loss in two-dimensional polaritonic photonic crystal waveguides. *Physics Letters A*, 351(4–5):319–322, 2006.

[12] T. F. Krauss and R. M. De La Rue. Photonic crystals in the optical regime - past, present and future. *Progress in Quantum Electronics*, 23(2):51–96, 1999.

[13] P. R. Villeneuve and M. Piché. Photonic bandgaps: What is the best numerical representation of periodic structures? *Journal of Modern Optics*, 41(2):241–256, 1994.

[14] P. R. Villeneuve and M. Piché. Photonic band gaps of transverse-electric modes in two-dimensionally periodic media. *Journal of the Optical Society of American A*, 8(8):1296–1305, 1991.

[15] P. R. Villeneuve and M. Piché. Photonic band gaps in periodic dielectric structures. *Progress in Quantum Electronics*, 18:153–200, 1994.

[16] B. P. Hiett, J. M. Generowicz, S. J. Cox, M. Molinari, D. H. Beckett, and K. S. Thomas. Application of finite element methods to photonic crystal modelling. In *IEE. The fourth International Conference of Computation in Electromagnetics*, volume 149(5), pages 293–296, 2002.

[17] H. Y. D. Yang. Finite difference analysis of 2-D photonic crystals. *IEEE Transactions on Microwave Theory and Techiques*, 44(12(2)):2688–2695, 1996.

[18] K. Bierwirth, N. Schulz, and F. Arndt. Finite-difference analysis of rectangular dielectric waveguide structures. *IEEE Transactions on Microwave Theory and Techiques*, 34(11):1104–1114, 1986.

[19] R. S. Varga. *Matrix Iterative Analysis.* Prentice Hall, 1962.

[20] J. Gielis. A generic geometric transformation that unifies a wide range of natural and abstract shapes. *American Journal of Botany*, 90:333–338, 2003.

[21] BandSOLVE. Rsoft design group, inc. http://www.rsoftdesign.com/.

[22] R. D. Meade, A. M. Rappe, K. D. Brommer, and J. D. Joannopoulos. Nature of the photonic band gap: some insights from a field analysis. *Jounal of the Optical Society of America*, 10(2):328–332, 1993.

[23] R. Padjen, J. M. Gerard, and J. Y. Marzin. Analysis of the filling pattern dependence of the photonic band gap for two-dimensional systems. *Journal of Modern Optics*, 41(2):295–310, 1994.

[24] P. R. Villeneuve and M. Piché. Photonic band gaps in two-dimensional square and hexagonal lattices. *Physical Review B*, 46(8):4969–4972, 1992.

[25] M. Plihal and A. A. Maradudin. Photonic band structure of two-dimensional systems: The triangular lattice. *Physical Review B*, 44(16):8565–8571, 1991.

[26] J. M. Geremia, J. Williams, and H. Mabuchi. Inverse-problem approach to designing photonic crystals for cavity QED experiments. *Physical Review E*, 66:066606, 2002.

[27] D. Englund, I. Fushman, and Vačković. General recipe for designing photonic crystal cavities. *Jounal of the Optical Society of America*, 13(16):5961–5975, 2005.

[28] S. J. Cox and D. C. Dobson. Band structure optimization of two-dimensional photonic crystals in H-polarization. *Journal of Computational Physics*, 158:214–224, 2000.

[29] Y. Chen, R. Yu, O. Nohadani, S. Haas, and A. F. J. Levi. Adaptive design of nanoscale dielectric structures for photonics. *Journal of Applied Physics*, 94(9):6065–6068, 2003.

[30] C. Y. Kao, S. Osher, and E. Yablonovitch. Maximising band gaps in two-dimensional photonic crystals by using level set methods. *Journal of Applied Physics B: Lasers and Optics*, 81:235–244, 2005.

[31] Y. Jiao, S. Fan, and D. A. B. Miller. Demonstration of systematic photonic crystal device design and optimization by low-rank adjustments: an extremely compact mode separator. *Optics Letters*, 30(2):141–143, 2005.

[32] J. S. Jensem and O. Sigmund. Topology optimization of photonic crystal structures: a high-bandwidth low-loss t-junction waveguide. *Jounal of the Optical Society of America B*, 22(6):1191–1198, 2005.

[33] L. Shen, Z. Ye, and S. He. Design of two-dimensional photonic crystals with large absolute band gaps using a genetic algorithm. *Physical Review B*, 68:035109, 2003.

[34] R. P. Drupp, J. A. Bossard, D. H. Werner, and T. S. Mayer. Single-layer multiband infrared metallodielectric photonic crystals designed by genetic algorithm optimization. *Applied Physics Letters*, 86:081102, 2005.

[35] S. Preble, M. Lipson, and H. Lipson. Two-dimensional photonic crystals designed by evolutionary algorithms. *Applied physics letters*, 86:061111, 2005.

[36] J. Goh, I. Fushman, D. Englund, and J. Vučković. Genetic optimization of photonic bandgap structures. *The International Journal of Optics*, 15(13):8218–8230, 2007.

[37] L. A. Piegl and W. Tiller. *The NURBS Book (Monographs in Visual Communication)*. Springer, 2nd edition, 1996.

[38] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15:407–412, 1978.

[39] A. Sóbester and A. J. Keane. Empirical comparison of gradient-based methods on an engine inlet shape optimization problem. *AIAA Paper*, 2002-5507, 2002.

[40] John H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992.

[41] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139:1–77, 1996.

[42] J. T. Batina. A gridless Euler/Navier-Stokes solution algorithm for complex-aircraft applications. *AIAA Paper*, 93-0333, 1993.

[43] G. R. Liu and Y. T. Gu. *An Introduction to Meshfree Methods and Their Programming*. Springer: Dordrecht, 2005.

[44] P. P. Chinchapatnam, K. Djidjeli, P. B. Nair, and M. Tan. A compact RBF-FD based meshless method for the incompressible Navier-Stokes equations. *Journal of Engineering for the Maritime Environment*, 223(3):275–290, 2009.

[45] R. Franke. Smooth interpolation of scattered data by local thin plate splines. *Computers and Mathematics with Applications*, 8:273–281, 1982.

[46] M. A. Golberg and C. S. Chen. A bibliography on radial basis function approximations. *Boundary Element Communications*, 7:155–263, 1996.

[47] H. Wendland. On the smoothness of positive definite and radial functions. *Journal of Computational and Applied Mathematics*, 101:177–188, 1999.

[48] E. J. Kansa. Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics-I. Surface approximations and partial derivatives estimates. *Journal of Computers and Mathematics with Applications*, 19(8-9):127–145, 1990.

[49] E. J. Kansa. Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics-II. Solution to parabolic, hyperbolic and elliptical partial differential equations. *Journal of Computers and Mathematics with Applications*, 19(8-9):147–161, 1990.

[50] Y. C. Hon and X. Z. Mao. An efficient numerical scheme for Burger's equations. *Journal of Applied Mathematics and Computation*, 95:37–50, 1998.

[51] A. I. Fedoseyev, M. J. Friedman, and E. J. Kansa. Continuation for nonlinear elliptic partial differential equations discretized by the multiquadric method. *Journal of Bifurcation and Chaos*, 10:481–492, 2000.

[52] P. P. Chinchapatnam, K. Djidjeli, and P. B. Nair. Meshless domain decomposition schemes for nonlinear elliptic pdes. In *Proceedings of the Short Papers of the 3rd M.I.T. Conference on Computational Fluid and Solid Mechanics*, pages 1082–1086, 2005.

[53] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Journal of Advances in Computational Mathematics*, 11(2-3):193–210, 1999.

[54] R. B. Platte and T. A. Driscoll. Computing eigenmodes of elliptic operators using radial basis functions. *Journal of Computers and Mathematics with Applications*, 48:561–576, 2004.

[55] E. J. Kansa and Y. C. Hon. Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations. *Journal of Computers and Mathematics with Applications*, 39(7-8):123–137, 2000.

[56] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Journal of Advances in Computational Mathematics*, 4(4):389–396, 1995.

[57] Z. Wu. Compactly supported positive definite radial functions. *Journal of Advances in Computational Mathematics*, 4(3):283–292, 1995.

[58] M. D. Buhmann. A new class of radial basis functions with compact support. *Journal of Mathematics of Computation*, 70:307–318, 2000.

[59] Z. Wu. Solving PDEs with radial basis functions. In *Advances in Computational Mathematics, Lecture Notes in Pure and Applied Mathematics*, volume 202, 1998.

[60] B. Nayroles, G. Touzot, and P. Villion. Generalizing the finite element method: diffuse approximation and diffuse elements. *Journal of Computational Mechanics*, 10:307–318, 1992.

[61] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37:141–158, 1981.

[62] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[63] G. R. Liu and Y. T. Gu. A local radial point interpolation method (LR-PIM) for free vibration analyses of 2-D solids. *Journal of Sound and Vibration*, 246(1):29–46, 2001.

[64] W. K. Liu, S. Jun, and Y. F. Zhang. Reproducing kernel particle methods. *International Journal for Numerical Methods in Fulids*, 20:1081–1106, 1995.

[65] S. N. Atluri and T. Zhu. A new meshless local petrov-galerkin (MLPG) approach in computational mechanics. *Computational Mechanics*, 22:117–127, 1998.

[66] V. Girault. Theory of a finite difference method on irregular networks,. *Journal of Numerical Analysis*, 11:260–282, 1974.

[67] G. R. Liu and Y. T. Gu. A meshfree method: meshfree weak-strong (MWS) form method, for 2-D solids. *Journal of Computational Mechanics*, 33(1):2–14, 2002.

[68] M. J. D. Powell. The theory of radial basis function approximation. In *Wavelet, Subdivision Algorithm and Radial Basis Functions*, volume 2(3), pages 105–210, 1990.

[69] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press: Cambridge, 1992.

[70] W. Axmann and P. Kuchment. An efficient finite element method for computing spectra of photonic and acoustic band-gap materials - I. scalar case. *Journal of Computational Physics*, 150(2):468–481, 1999.

[71] D. C. Dobson. An efficient method for band structure calculations in 2D photonic crystals. *Journal of Computational Physics*, 149:363–376, 1999.

[72] P. Kuchment. *Floquet theory for partial differential equations.* Birkhäuser Basel, P.O. Box 133, CH-4010 Basel, Switzerland: Birkhäuser Verlag, 1993.

[73] B. P. Hiett. *Photonic Crystal Modelling using Finite Element Analysis.* PhD thesis, School of Engineering Sciences, University of Southampton, 2002.

[74] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, volume 24, pages 327–335, 1990.

[75] Intel graphics support. http://www.intel.com/support/graphics/.

[76] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick. The potential of the cell processor for scientific computing. In *Proceedings of the 3rd conference on Computing frontiers*, pages 9–20, New York, NY, USA, 2006. ACM.

[77] Intel threading building blocks. http://www.threadingbuildingblocks.org/.

[78] Cilk arts. http://www.cilk.com/.

[79] Clearspeed. http://www.clearspeed.com/.

[80] NVIDIA Tesla. http://www.nvidia.com/object/tesla.

[81] ATI Firestream. http://ati.amd.com/products/streamprocessor/specs.html.

[82] Intel Larrabee. http://software.intel.com/en-us/articles/larrabee/.

[83] Top 500 Supercomputers. http://www.top500.org/.

[84] N. L. Max. Vectorized procedural models for natural terrain: Waves and islands in the sunset. In *Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1981. ACM.

[85] T. Whitted and D. M. Weimer. A software test-bed for the development of 3-D raster graphics systems. *ACM SIGGRAPH Computer Graphics*, 15(3):271–277, 1981.

[86] R. L. Cook. Shade trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, volume 18, pages 223–231, New York, NY, USA, 1984. ACM.

[87] K. Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985.

[88] P. Hanrahan and J. Lawson. A language for shading and lighting calculations. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1990. ACM.

[89] A. A. Apodaca and L. Gritz. *Advanced RenderMan: Creating CGI for Motion Picture*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[90] B. W. Kernighan and D. M. Ritchie. *The C programming language*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1988.

[91] M. Olano and A. Lastra. A shading language on graphics hardware: the pixelflow shading system. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 159–168, New York, NY, USA, 1998. ACM.

[92] M. S. Peercy, M. Olano, J. Airey, and P. J. Ungar. Interactive multi-pass programmable shading. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 425–432, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[93] K. Proudfoot, W. R. Mark, S. Tzvetkov, and P. Hanrahan. A real-time procedural shading system for programmable graphics hardware. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 159–170, New York, NY, USA, 2001. ACM.

[94] W. Mark, S. Glanville, and K. Akeley. Cg: A system for programming graphics hardware in a C-like language, 2003.

[95] R. Fernando and M. J. Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[96] R. J. Rost. *OpenGL(R) Shading Language (2nd Edition)*. Addison-Wesley Professional, 2005.

[97] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Mike, and H. Pat. Brook for GPUs: Stream computing on graphics hardware, 2004.

[98] Folding@home. http://folding.stanford.edu/.

[99] AMD Developer Central. http://developer.amd.com/programs/researcher/.

[100] AMD Stream SDK. http://developer.amd.com/gpu/ATIStreamSDK/.

[101] NVIDIA CUDA. http://www.nvidia.com/cuda.

[102] OpenCL. http://www.khronos.org/opencl/.

[103] Accelereyes Jacket. http://www.accelereyes.com/.

[104] GPULib. http://www.txcorp.com/products/GPULib/.

[105] EM Photonics CULA. http://www.culatools.com/.

[106] NVIDIA Fermi. http://www.nvidia.com/fermi.

[107] NVIDIA Nexus. http://www.nvidia.com/nexus.