University of Southampton

Faculty of Engineering and the Environment

Algorithms for Scientific Computing

Neil Stephen O’Brien

Doctor of Philosophy

October 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND THE ENVIRONMENT

Doctor of Philosophy

ALGORITHMS FOR SCIENTIFIC COMPUTING

Neil Stephen O'Brien

There has long been interest in algorithms for simulating physical systems. We are concerned with two areas within this field: fast multipole methods and meshless methods.

Since Greengard and Rokhlin's seminal paper in 1987, considerable interest has arisen in fast multipole methods for finding the energy of particle systems in two and three dimensions, and more recently in many other applications where fast matrix-vector multiplication is called for. We develop a new fast multipole method that allows the calculation of the energy of a system of N particles in $\mathcal{O}(N)$ time, where the particles' interactions are governed by the 2D Yukawa potential which takes the form of a modified Bessel function K_v.

We then turn our attention to meshless methods. We formulate and test a new radial basis function finite difference method for solving an eigenvalue problem on a periodic domain. We then apply meshless methods to modelling photonic crystals. After an initial background study of the field, we detail the Maxwell equations, which govern the interaction of the light with the photonic crystal, and show how photonic band gaps may be given rise to. We present a novel meshless weak-strong form method with reduced computational cost compared to the existing meshless weak form method. Furthermore, we develop a new radial basis function finite difference method for photonic band gap calculations.

Throughout the work we demonstrate the application of cutting-edge technologies such as cloud computing to the development and verification of algorithms for physical simulations.

Contents

1 Introduction to algorithms **1**

2 Computational technologies **5**

2.1 Special-purpose hardware 6

2.1.1 Historical special purpose hardware 6

2.1.2 GRAPE 7

2.2 Cell Broadband Engine 11

2.3 GPU acceleration 12

2.3.1 Introduction 13

2.3.2 GPU programming environments 15

2.3.3 Higher-level GPU interfaces 18

2.4 Cloud computing 22

2.4.1 Introducing cloud computing 22

2.4.2 Areas of strength for cloud-based architectures 26

2.4.3 Scientific computing in the cloud 27

2.5 Summary 30

3 Fast multipole methods **31**

3.1 Introduction 31

3.2 Particle-in-cell methods 32

3.2.1 Particle-particle/particle-mesh (P^{3}M) methods 33

3.3 Tree codes 34

3.4 The 2D Greengard-Rokhlin FMM 35
3.5 Enhanced FMM algorithms 40
3.6 Summary 42

4 A fast multipole method for the 2D Yukawa potential 43

4.1 Introduction 43
4.2 Multipole formulæ 45
4.2.1 Interaction energy 46
4.2.2 Multipole expansion 48
4.2.3 Translation of expansions 48
4.3 Algorithm 50
4.4 Periodic boundary conditions 53
4.5 Results 54
4.6 Conclusions 61

5 Meshless methods 63

5.1 Introduction 63
5.2 Mesh based methods 64
5.2.1 Finite element method 64
5.2.2 Finite difference method 65
5.2.3 Finite volume method 66
5.2.4 Other mesh-based methods 66
5.3 Meshless methods 67
5.3.1 Introduction 67
5.3.2 Radial basis functions 69
5.3.3 RBF interpolation 75
5.3.4 Classes of meshless method 76
5.3.5 Disadvantages of meshless methods 78
5.4 An RBF-FD method for an eigenvalue problem on a periodic domain . . 79
5.4.1 Introduction 79

5.4.2 Formulation 81
5.4.3 Formulation of a higher-order RBF-FD scheme 86
5.4.4 Results 91
5.5 Summary and conclusions 97

6 Meshless methods for photonic crystal modelling 101

6.1 Introduction to photonic crystals 102
6.2 Mathematical and physical background 102
6.2.1 Prerequisite crystallography 103
6.2.2 The Maxwell equations 106
6.2.3 Harmonic solutions and the eigenvalue equation 107
6.2.4 Photonic band gaps in one-dimensional photonic crystals 108
6.2.5 Scale invariance in the Maxwell equations 111
6.2.6 Two-dimensional photonic crystals 111
6.3 Meshless methods for photonic crystal modelling: background 112
6.3.1 The plane-wave expansion method as a reference for comparison 114
6.3.2 Meshless local strong form method 115
6.3.3 Meshless local weak form method 115
6.3.4 Other meshless methods for photonic crystal modelling 116
6.4 A meshless local weak-strong form method for 2D bandgap calculations 117
6.4.1 Meshless method formulation 117
6.4.2 An illustrative example 125
6.4.3 MLWSFM in the cloud 131
6.4.4 Results 135
6.5 A meshless RBF-FD method for 2D bandgap calculations 139
6.5.1 TM mode formulation 140
6.5.2 Results 143
6.6 Conclusions and outlook 144

7 Summary and Outlook **147**

7.1 Summary 147

7.2 Outlook and discussion 149

Bibliography **155**

List of Tables

2.1 Comparison of GRAPE systems 8
2.2 Comparison of specifications of GRAPE-DR and GPUs 10
2.3 Relative price and performance of Cell BE and Tesla accelerators 19

4.1 FMM accuracy 56

6.1 Parameters chosen for illustrative example, MLWSFM 127
6.2 Parameters chosen for illustrative example, PWEM 128
6.3 Memory consumption for MLWSFM data structures 130
6.4 Comparison of resource use between PWEM and MLWSFM 131

List of Figures

2.1 The GRAPE-DR architecture 9
2.2 The relative speed of consumer GPUs and CPUs from 2003 to 2012 . . . 14
2.3 OpenCL and native Cell/CUDA SDK speedup comparisons 18

3.1 Spatial division and Barnes-Hut tree structure 35
3.2 Well-separated sets of particles 37

4.1 Interaction set of a single box 45
4.2 Vectorised Bessel functions, definition of angles and vectors 46
4.3 Well-separated boxes b and $\widetilde{b}$ 46
4.4 Translation of multipole expansions from a child box to its parent box . 49
4.5 Interaction set of box 0 at mesh level 3, with periodic boundaries 54
4.6 Interaction set of box 0 at mesh level 2, with periodic boundaries 54
4.7 Inhomogeneous particle distribution 55
4.8 Series cut-off error as a function of number of terms retained 58
4.9 Variation of series cut-off error with magnetic penetration depth 58
4.10 Time to evaluate non-periodic system energy directly and by the FMM . 59
4.11 Time to evaluate energy of infinitely tiled system 60

5.1 Globally supported RBFs for three values of the shape parameter 71
5.2 Compactly supported RBFs 74
5.3 System domain with periodic boundary conditions. 80

5.4 RBF-FD stencil 82
5.5 Nodes and the RBF-FD influence domain 83
5.6 Higher order RBF-FD stencil 87
5.7 Mean RBF-FD relative error against number of nodes 92
5.8 Relative error for RBF-FD and higher order RBF-FD 93
5.9 Determination of orders of convergence for RBF-FD methods 94
5.10 Runtime comparison for RBF-FD methods 95
5.11 Relative RBF-FD error against shape parameter 96
5.12 RBF-FD error against number of nodes, various RBFs, uniform nodes . . 97
5.13 RBF-FD error against number of nodes, various RBFs, Sobol nodes . . . 98

6.1 Brillouin zone for a square lattice 105
6.2 One-dimensional photonic crystal 109
6.3 Dispersion relation of 1D photonic crystal 110
6.4 Part of the two-dimensional photonic crystal 111
6.5 Band gap of two-dimensional photonic crystal 113
6.6 Physical system and subdomains in MLWSFM 118
6.7 Node positions in MLWSFM example 128
6.8 Gaussian quadrature points and background cells in MLWSFM example 129
6.9 Cloud worker architecture pattern 132
6.10 MLWSFM error as a function of number of nodes and background cells . 136
6.11 MLWSFM error as a function of shape parameter and domain size 138
6.12 TE and TM mode band diagrams from PWEM and MLWSFM 139
6.13 Domain of the periodic system 142
6.14 TM mode band diagrams from PWEM and RBF-FD 143

7.1 TE mode band diagrams from PWEM and RBF-FD 152

Author's declaration

I, Neil O'Brien, declare that the thesis entitled *Algorithms for Scientific Computing* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. 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;
- work which has been published, presented, or is under review for publication is listed under "Contributions" on the following page.

Signed:

Date:

Contributions

The following work has been published and/or presented:

- Johnston, S.J., O'Brien, N.S., Lewis, H.G., Hart, E.E., White, A. and Cox, S.J. Clouds in Space: Scientific Computing using Windows Azure. *Journal of Cloud Computing: Advances, Systems and Applications*, Vol. 2, No. 1, 2013.
- O'Brien, N.S., Djidjeli, K and Cox, S.J. A meshless RBF-FD method for 2D bandgap calculations, in *Photon12* conference. Institute of Physics: Optics and Photonics Division, Durham, UK, September 2012
- Sóbester, A., Johnston, S.J., Scanlan, J.P., Hart, E.E. and O'Brien, N.S. Rapid development of Bespoke Unmanned Platforms for Atmospheric Science. *Geophysical Research Abstracts*, Vol. 14, EGU2012-11103, 2012
- Johnston, S.J., Lewis, H.G., Hart, E.E., White, A., O'Brien, N.S., Takeda, K. and Cox, S.J. Space situational awareness using a cloud based architecture. In *2011 Beijing Space Sustainability Conference*. Secure World Foundation, Beijing, China, October 2011.
- O'Brien, N.S., Johnston, S.J., Hart, E.E., Djidjeli, K and Cox, S.J. Exploiting cloud computing for algorithm development. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, pages 336-342, October 2011. doi:10.1109/CyberC.2011.60.
- Sóbester, A, Johnston, S.J., Scanlan, J.P., O'Brien, N.S., Hart, E.E., Crispin, C.I. and Cox, S.J. High Altitude Unmanned Air System for Atmospheric Science Missions. In *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, September 2011

The following work has been accepted for publication:

- Cox, S.J., Cox, J.T., Boardman, R.P., Johnston, S.J., Scott, M., O'Brien, N.S. Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing*. To appear.

The following work is under review for publication:

- O'Brien, N.S., Djidjeli, K., Cox, S.J. Solving an eigenvalue problem on a periodic domain using a radial basis function finite difference scheme. Submitted.

Acknowledgements

I would like to thank my supervisors Simon Cox and Kamal Djidjeli for providing me the opportunity to carry out this work, and for their guidance and support. I'd also like to thank Marc Molinari for his expertise and guidance.

I'm very grateful to my family for their unending help and support. I wouldn't have made it without you.

Thanks are also due to all my friends who have helped along the way. Particular thanks go to Steven Johnston for the invaluable motivation and inspiration he gave me, and for patiently helping me to make the best of many opportunities; to Rich Boardman for some very helpful technical discussions and LaTeX advice, and some fun woodworking projects at the weekends; and to Liz Hart, whose work on photonic crystals paved the way for some of the developments in this work, and who often helped me out. András Sóbester's dry humour brightened up many of my days and his ASTRA balloon launches lead to some enjoyable adventures. Marc Bunyard and Andy Chipperfield gave me brilliant support through some tough times; Greg Jasion and Mark Scott are great friends who were always happy to help out.

I'm also thankful to Microsoft and EPSRC for their financial support.

Lámh láidir an uachtar

the strong hand uppermost

Trademarks and copyright information

Nomenclature

i	$\sqrt{-1}$
δ	Radius of Ω_{w} is $r + \delta$ in testing the MLWSFM
δ_{ij}	Dirac δ-function: $\delta_{ij} = 1$ if $i = j$, 0 otherwise
ϵ	Precision demanded from a multipole method with respect to the total charge
Γ	A special point at the centre of the first Brillouin zone
γ_j	Coefficients in RBF interpolants
κ	Imaginary component of $\boldsymbol{k}$ when it is complex
λ_{m}	Magnetic penetration depth in a high-temperature superconductor
λ_{nk}	Solutions of the elliptic Helmholtz equation on a square domain
$\mathcal{L}$	An arbitrary, linear differential operator
$\mu(\boldsymbol{x})$	Relative magnetic permeability
μ_0	Vacuum magnetic permeability
$\boldsymbol{\alpha}$	Eigenvectors describing the nodal field values of allowable modes of propagation in 2D photonic crystals
$\boldsymbol{\eta}$	Vector containing m node numbers
$\boldsymbol{\Psi}_{\boldsymbol{k}}(\boldsymbol{r})$	Wave function of Schrödinger's equation

$\boldsymbol{\varrho}_{1,\dots,4}$	Positions relative to the origin of the four Gaussian quadrature evaluation points in a background cell
$\boldsymbol{a}, \boldsymbol{b}$	Basis vectors for a 2D lattice
$\boldsymbol{B}$	Magnetic induction field
$\boldsymbol{D}$	Displacement field
$\boldsymbol{E}$	Macroscopic electric field
$\boldsymbol{e}$	Vector where all elements $e_i = 1$
$\boldsymbol{H}$	Macroscopic magnetic field
$\boldsymbol{k}$	Wave vector (with components k_x, k_y, k_z)
$\boldsymbol{q}_j$	Vector from the centre of a box b to particle j in the box
$\boldsymbol{R}$	Lattice vector
∇^2	The operator $(\partial^2/\partial x^2, \partial^2/\partial y^2)$
ω	Frequency of the electromagnetic radiation
Ω_b	Interaction set of box b
Ω_i	RBF-FD stencil for node i
Ω_s	Subdomain in MLWSFM where strong form is applied
Ω_w	Subdomain in MLWSFM where weak form is applied
$\phi(z)$	Potential at the point $z \in \mathbb{C}$ in multipole methods
$\psi(\boldsymbol{x})$	Shape function in RBF-FD method
$\Theta(\boldsymbol{a})$	$\arg(a_1 + \mathrm{i}a_2)$
$\varepsilon(\boldsymbol{x})$	Relative permittivity

ε_0	Vacuum permittivity
$\varepsilon_{\text{series}}$	Some measure of the error due to truncation of multipole-like series
φ	Wave functions for in-plane light in 2D photonic crystals
$\widetilde{\gamma}$	Expansion coefficients in a higher-order RBF-FD scheme
$\widetilde{w}_{(k,j)}$	Weights in a higher order RBF-FD scheme (superscript may denote relevant operator)
ζ, η	Coordinates in which Gaussian integration is performed
a	Lattice parameter
a_k	Multipole expansion coefficients
c	Shape parameter of a radial basis function
c	Speed of light, $c = 1/\sqrt{\varepsilon_0 \mu_0}$
C^k	Smoothness class of a function with k continuous derivatives
$E^{(b,\widetilde{b},l)}$	Energy of interaction between boxes b and $\widetilde{b}$ at level l where $\widetilde{b}$ is in the interaction set of b
h	Spacing between adjacent uniformly-distributed nodes
$I_\nu(\cdot)$	modified Bessel function of the first kind, of order ν
k_{trunc}	Number of terms retained in a truncated multipole-like series
$K_\nu(\cdot)$	modified Bessel function of the second kind, of order ν
l	Level of subdivision in a multipole method, where the unit cell is at $l = 1$ and larger l-values correspond to finer division
M	Number of grid points used in a PIC scheme Number of nodes in an RBF-FD influence domain A special point at the corner of the first Brillouin zone

m	Number of nodes where we use derivative information in a higher-order RBF-FD scheme
N	Number of bodies in a many-body problem Number of nodes used in meshless methods
Q	Total charge in multipole method, $Q = \sum_i q_i$
$q_{1,\dots,m}$	Charges of particles located at $\boldsymbol{x}_{1,\dots,m}$ in a multipole method
R	Groups of particles of radius R are well separated when their centres are at least $3R$ apart in a multipole method
r	Radius of rod in unit cell of 2D photonic crystal
$s(\boldsymbol{x})$	RBF interpolant for $u(\boldsymbol{x})$
$U(\boldsymbol{s})$	Potential at location $\boldsymbol{s}$ in the multipole method for the Yukawa potential
$u(\boldsymbol{x})$	Unknown function in meshless methods
V	Volume of a cell at some level of subdivision in a tree code
$w_{(k,j)}$	Weights in an RBF-FD scheme (superscript may denote relevant operator)
X	A special point on the face of the first Brillouin zone

Chapter 1

A brief introduction to algorithms and context for this thesis

Since the early development of digital computers around the era of World War II, a vast body of work has been undertaken on the development of increasingly efficient and involved algorithms to enable computers to tackle a broad range of problems. This work has been accompanied by a continual increase in the capabilities and speeds of available hardware, captured by Moore's Law [1, 2], which has together lead to a massive increase in the usefulness of digital computers in science and engineering problems.

In a special issue of *Computing in Science and Engineering* the editors, Jack Dongarra and Francis Sullivan, reviewed the algorithms developed over the 20th century and picked the top ten of those algorithms [3]. The methods that they picked each changed the way that a certain class of problem is approached, or enabled new classes of problems to be tackled.

The algorithms that were selected include the following:

- Metropolis algorithm for Monte Carlo
- Simplex method for linear programming

- Krylov subspace iteration
- Decompositional approach to matrix computations
- The FORTRAN optimising compiler
- QR algorithm for eigenvalue computations
- Quicksort algorithm
- Fast Fourier transform
- Integer relation detection
- Fast multipole method

In the present work, we develop several new algorithms, each of which is applicable to a specific problem in scientific computing. These algorithms are:

- A new fast multipole method for the 2D Yukawa potential,
- A meshless local weak-strong form method for photonic crystal modelling,
- A meshless radial basis function finite difference method for solving an eigenvalue problem on a periodic domain,
- A meshless radial basis function finite difference method for photonic crystal modelling.

These contributions are aligned rather closely with the top 10 algorithms picked out by Dongarra and Sullivan.

The fast multipole method is formulated in a similar style to that of Greengard and Rokhlin's seminal work [4], with the important difference that it is suitable for the 2D Yukawa potential and naturally handles infinitely-tiled repeats of the unit cell. It has applications in modelling the vortex state in superconductor physics simulations, which often make use of the Metropolis-Hastings algorithm. In implementing this method, it was necessary to calculate many convolutions, and to this end a fast Fourier transform could have been employed, though it was concluded that in this case, the overheads made the FFT unfavourable. The meshless methods for photonic crystal modelling all reduce the problem eventually to a generalised eigenvalue problem, which is solved by library routines formulated either in terms of the QR algorithm, or subspace iterative methods – which rely on Krylov techniques and may employ LU decomposition. Moreover, all the algorithms in the present work owe some of their performance to modern optimising compilers. Thus, the present work has direct relation to several of the editors' picks.

In addition to the algorithmic contributions made herein, we review the recent developments in novel technologies for scientific computations, including such machines as the GRAPE series of computers, and the increasingly popular trend towards applying the new technologies of GPGPU and cloud computing in science and engineering. We present some results from the development and verification of a meshless method using a cloud-based architecture.

This thesis is structured as follows:

Introduction. The present chapter introduces the historical context for the work and illustrates how it fits in with some of the key developments in the field of numerical algorithms. Introductions to specific areas and algorithms are included in each of the subsequent chapters.

Computational technologies. In this chapter, hardware developments targeted at accelerating numerical computing to reduce wall-clock time are reviewed. At first we focus on special purpose, dedicated hardware such as the GRAPE series of machines, and then move on to the current leading-edge trends of general purpose and scientific computing on GPUs (GPGPU) and cloud based architectures. The content in this section foreshadows the application of a cloud-based architecture during the development and testing of a meshless method for photonic crystal modelling in a later chapter.

Fast multipole methods. This chapter provides background to the many-body problem in physics. Since there is no general analytical solution for $N > 2$ bodies, numerical solution methods are reviewed, beginning with the naïve method that takes a time of $\mathcal{O}(N^2)$ through various accelerated methods to the fast multipole method which reduces the problem to linear (*i.e.* $\mathcal{O}(N)$) time.

A fast multipole method for the 2D Yukawa potential. This chapter gives the formulation of a fast multipole method for the energy of systems of particles interacting via the 2D Yukawa potential. The method was implemented and performance results are given in which it is compared to the naïve method for run time and accuracy with various parameters.

Meshless methods. In this chapter, the background to meshless methods is surveyed, and by way of comparison to mesh-based methods such as the finite element and finite difference methods, advantages of the meshless methods are elucidated. The chapter also introduces radial basis functions, and gives the formulation of a new radial basis function finite difference (RBF-FD) method for solving an eigenvalue problem on a periodic domain, including the formulation of a higher-order RBF-FD scheme.

Meshless methods for photonic crystal modelling. In this chapter, meshless methods are applied to the modelling of photonic crystals. Existing work in the field has proved that these methods can successfully model such crystals. This chapter presents the formulation of, and some results from, two new meshless methods suited to modelling photonic crystals.

Conclusions and outlook. This chapter reviews the main findings of the work, and goes on to show some areas in which further useful development would be likely as a result of continued research efforts.

Chapter 2

Computational technologies

While the algorithms of high-performance computing have been evolving rapidly in terms of the size and complexity of the problems that can be solved, from the Monte Carlo method in 1949 [5] to the fast algorithms of today, the computer hardware on which they run has also undergone dramatic changes. In this chapter, some of the more recent developments related to high-performance computing are reviewed. For historical context, we briefly consider some special purpose computers in §2.1.1. In §2.1.2 we review the GRAPE series of machines which, although initially designed just for astrophysical *N*-body simulations, are being developed into increasingly general purpose parallel scientific computers.

Moving further towards the commodity hardware that underlies progress in some fields of scientific computing, we devote §2.2 to the Cell Broadband Engine. A significant and exciting recent development is the advent of general-purpose programming on graphics processing units (GPUs). We discuss applications of this technology to scientific problems in §2.3. This is currently a field enjoying huge growth and rapid innovation, driven both by the increasing demands of consumers for immersive gaming and multimedia experiences, and of the scientific community for affordable, massively parallel processing. We finish the section by introducing the paradigm of cloud computing and its applicability to scientific computation in §2.4, and present a summary in §2.5. The material in this chapter underlies the development and tuning of one of the meshless algorithms presented in §6.4, a large proportion of which was accelerated by the application of a cloud based architecture.

2.1 Special-purpose hardware

In this section, we introduce some special-purpose hardware that was specially designed for scientific computations. Many of the machines mentioned here are of historical interest, and are mentioned to give some context for the more contemporary technologies discussed later. These machines typically did not enjoy the benefits and economies of scale associated with mass-market products, and therefore were often very costly. This is in stark contrast to GPUs, which we shall examine in §2.3.

2.1.1 Historical special purpose hardware

The literature contains many examples of historical special-purpose computers. To provide a little historical context, we summarise two such machines in this section, before moving on to look at more contemporary solutions.

Auerbach, *et al.*, designed and implemented a special purpose computer for molecular dynamics (MD) [6]. This work was done in 1987, a time when the efficiency of the serial von Neumann architecture was being called into question in scientific computations. At that time, the peak performance, since described as '*the performance vendors guarantee not to exceed*'[1], of a single processor supercomputer was 4 Gflops. The sustained performance was considerably lower. Therefore, Auerbach *et al.* introduced a special-purpose parallel computer which was built at IBM's Almaden Research Centre. They noted that MD problems are eminently suitable for parallelisation since they may be evenly divided up for load-balancing across processors, and they have low communications overheads. The prototype special-purpose computer, known as SPARK, achieved a sustained compute rate 1.73 times greater than the IBM

[1]`http://www.hoise.com/primeur/01/articles/weekly/UH-PR-09-01-1.html` accessed 11 November 2011

370/3081 (a mainframe introduced in 1980). It was anticipated that an array of ten SPARK nodes would have a performance of around 30 Mflops, or 17× faster than the IBM mainframe.

Boehncke, *et al.* proposed a special-purpose machine for MD calculations in 1990 [7]. The machine consisted of a network of Transputer nodes, running at 20 MHz and each having 4 MB RAM. They used 60 such nodes in a systolic ring network. The authors note that their system "does very well when large numbers of atoms are involved." It is interesting to note that in a ring topology, the processes carried out by the nodes must explicitly take account of the topology: coordinates, in this example, were sent clockwise around the ring, whilst forces were sent counter-clockwise. When forces are calculated, they are sent round the ring, with each Transputer running a routing process which decides if the force is relevant to the atoms that processor is dealing with; if it is not, the only action taken is to pass it on to the next neighbour.

2.1.2 GRAPE

The gravity pipe (GRAPE) was a special-purpose computer, developed for astrophysical N-body simulations. The GRAPE project was commenced at Tokyo University in 1989 and underwent several revisions.

The special GRAPE hardware acts as a force-calculation accelerator to a host workstation. The workstation is responsible for executing the vast majority of the instructions in any simulation. The GRAPE is called only to perform gravitational force calculations, in the innermost loop of the simulation [8]. The parts of the calculation that are run by the host workstation are generally the $\mathcal{O}(N)$ parts, whereas the force calculation, in some cases evaluated by the naïve $\mathcal{O}(N^2)$ approach, and in others by an $\mathcal{O}(N \log N)$ algorithm, is carried out on the GRAPE. Even the faster tree codes can benefit from being run on the GRAPE [9].

There exist GRAPE systems that allow arbitrary force implementations, lending themselves to applications such as molecular dynamics, for which the MDGRAPE system was specifically designed, but most systems are specific to astrophysical prob-

Machine	Year	Peak Speed	Notes
		Low-precision machines	
GRAPE-1	1989	240 Mflops	Concept system
GRAPE-3	1991	15 Gflops	Custom integration
GRAPE-5	1999	1 Tflops	Supports forces with arbitrary cutoff functions
		High-precision machines	
GRAPE-2	1990	40 Mflops	Supports IEEE-754 single and double precision
HARP-1	1993	180 Mflops	Calculates force and its time-derivative
GRAPE-4	1995	1 Tflops	Single-chip pipeline
GRAPE-6	2002	64 Tflops	6 pipelines per chip
GRAPE-DR	2008	2 Pflops	New architecture

Table 2.1: Comparison of GRAPE hardware, with speeds in floating-point operations per second, after [8].

lems [8]. In table 2.1 a summary of various GRAPE machines is presented.

From a programmer's perspective, the effort of harnessing the power of a GRAPE system is minimal: the use of the special-purpose hardware just requires calling a few library functions [10]. Thus, writing new software or porting existing codes is fairly straightforward. Since there is essentially just one calculation to be done on the GRAPE, writing the library function itself is also relatively straightforward.

The next GRAPE iteration was the greatly reduced array of processor elements with data reduction (GRAPE-DR), and it represented quite a departure from the previous GRAPE machine architectures. It was planned that the system would consist of 4096 processor chips, each having 512 cores clocked at 500 MHz. A processor chip's peak speed would be 512 Gflops. A prototype was built, featuring a single GRAPE-DR chip on the PCI-X bus. The final system was envisaged to be a cluster of 512 PCs each featuring two four-chip GRAPE-DR boards, for a total compute power of 2 Pflops [11].

The GRAPE-DR architecture, illustrated in fig. 2.1, is a modified form of the ba-

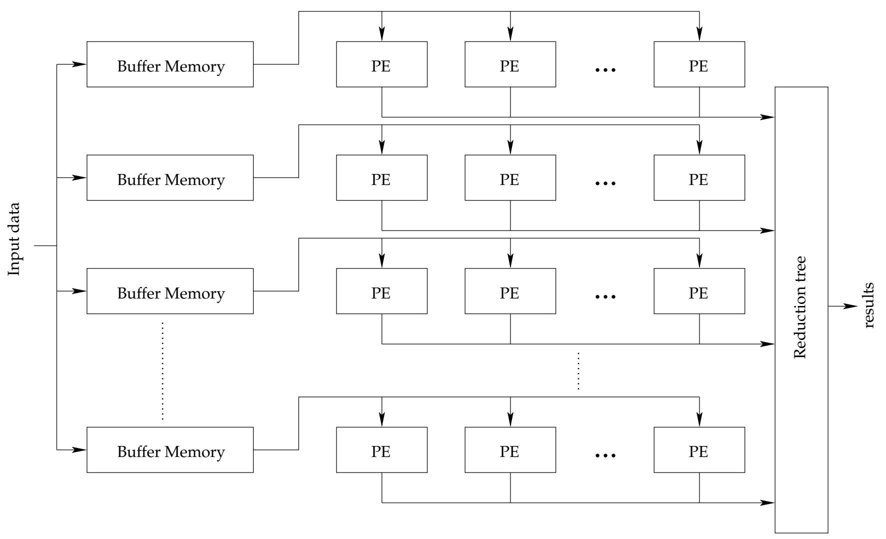


Figure 2.1: The GRAPE-DR processor architecture. In the figure, PE is an abbreviation for Processing Element. After fig. 4 in [11].

sic single instruction, multiple data (SIMD) architecture, where the processor elements (PEs) are grouped into blocks in hardware, and each block has some buffer memory and a connection to a reduction network. These customisations made the hardware ideal for scientific computing, where the answer to the problem being solved usually requires some reduction operation over the results of all the individual parallel computations.

Although at the time of writing there appear to be no recent published works on the GRAPE-DR project, a presentation appearing online [12] announced that a single chip achieves 0.5 Tflops, with a projected chip initial development cost of approximately \$10 M. In the presentation, the GRAPE-DR was compared to field programmable gate array (FPGA) solutions, and was claimed to offer significant advantages over the latter, because GRAPE-DR is programmable via assembly language and compilers whilst FPGAs require the use of VHDL (very-high-speed integrated circuit hardware description language). GRAPE-DR also offers the possibility of better use of silicon area, and potentially faster clock speeds than FPGAs.

The same talk compared the GRAPE-DR to a GPU (see §2.3), claiming its advant-

	GRAPE-DR	NVIDIA G92	AMD FS9170
Design rule (nm)	90	65	55
Clock speed (GHz)	0.5	1.5	0.8
No. of FPUs	512	112	320
Single precision peak Gflops	512	336	512
Double precision peak Gflops	256	U/S	102.4
Power consumed (W)	65	$\sim$70	$\sim$150

Table 2.2: Comparison of specifications of GRAPE-DR and GPUs, some data from [12]. U/S: double precision calculations are unsupported on the NVIDIA G92.

ages are the better usage of silicon, and the fact that as a processor designed for scientific computing, the GRAPE-DR offers efficient reduction, and low communications overheads. Conversely, the GRAPE-DR is a small-quantity device, which makes it expensive, and its development takes place over a much longer cycle of around 5 years, *vs.* closer to one year between generations of GPUs.

The current status of the GRAPE-DR project is that it is available commercially from K&F Computing Research Co[2]. Prices start from[3] ¥340 K (approx. £2650) for a PCI-Express card with a single GRAPE-DR chip to ¥1.6 M (approx. £12500) for an 8-processor version in an external chassis with 16-lane PCI Express connections to the host system. The performance figures for one chip, reproduced from [12] in table 2.2, were similar to those of GPUs, so it remains to be seen whether GRAPE-DR can continue to offer an advantage over the considerably cheaper GPUs whose performance is increasing rapidly, due both to the consumer demand for rich multimedia and gaming experiences, and an increasing interest from the scientific computing sector.

[2]`http://www.kfcr.jp/grapedr-e.html`

[3]prices and exchange rates current as of September 2012

2.2 Cell Broadband Engine

The Cell Broadband Engine was the first chip multiprocessor implementation that provided significant numbers of programmable, general-purpose cores [13]. The Cell Broadband Engine Architecture (CBEA) is a joint venture between Sony, Toshiba, and IBM. It was targeted at multimedia and compute-intensive workloads, and the first target for the architecture was a games console, the Sony PlayStation 3 (PS3) [14]. The architecture was designed to support large amounts of parallelism, and to offer high memory bandwidth. The CBEA allows programmers to explicitly perform data transfers in parallel with computation, potentially leading to a far better use of available bandwidth.

The Cell has been recognised as early as 2006 as having huge potential for carrying out scientific computations with both high performance and power efficiency [15]. IBM's Cell offerings are currently twofold: the PowerXCell 8i processor is at the core of Roadrunner, the world's second-fastest supercomputer at the time of writing[4]; and the Cell Broadband Engine, the chip used in the PS3.

The CBEA provides two processor components: the main processor, called the Power processor element (PPE), and the parallel processing accelerators, called synergistic processor elements (SPEs). The other major components of CBEA are an on-chip interconnect, the element interconnect bus (EIB), and the I/O interfaces which consist of a memory interface controller (MIC) and a Cell Broadband Engine Interface (BEI).

The PPE is intended to run the operating system, manage system resources, and control the SPEs, managing threads running thereupon. The SPEs are designed to handle the computational workload. The PPE supports 2-way hardware simultaneous multi-threading, and completes two double-precision operations in a clock cycle, so that a 3.2 GHz part has a peak performance of 6.4 Gflops.

The SPEs are where the computation is performed. The SPEs are based on SIMD RISC (reduced instruction set computer) processor and an MFC (memory flow control-

[4]`http://top500.org/system/ranking/10377`, accessed March 2010

ler), facilitating memory transfers to be carried out in parallel with computation. Each SPE also has a 256 kB local store for instructions and data, which must be explicitly managed by the programmer. Because the MFC is able to make up to 16 simultaneous transfers of up to 16 kB each between system and local memory, large memory operands can be pre-fetched into local memory before they are needed, saving on the high cost of frequent cache misses [16, §9]. The SPEs feature a double-precision arithmetic unit, with a throughput of 12.8 Gflops per SPE, so a PowerXCell 8i processor, featuring eight SPEs, has an overall peak performance of 108 Gflops. At the time of writing, however, the price-performance ratio does not compare favourably with that of GPUs (see §2.3): an IBM BladeCenter QS21 system, which contains two Cell Broadband Engine processors at 3.2 GHz, gives 216 Gflops peak aggregate performance and is priced around £3,800[5]. This compares to an NVIDIA GeForce GTX 295 that offers 1789 Gflops single precision or 149 Gflops double precision, at a cost of less than £500[6].

The Cell Broadband Engine has been found to deliver good speedups in some computational science applications. For example, in a quantum chromodynamics application, the US National Centre for Supercomputing Applications (NCSA) ported an existing program to the Cell architecture, and achieved a 3.4× speedup for a small simulation and a 5.7× speedup for a larger one, compared to a parallel implementation running on a quad-core Intel Xeon processor [14]. Other applications already using the CBEA include cosmology [17] and bioinformatics [18].

2.3 GPU acceleration

In this section, we address the increasingly popular trend of applying graphics processing units to scientific computations. We begin by giving some background and recent history, before considering programming environments such as CUDA, ATI Stream Computing and OpenCL. We also consider some higher-level programming interfaces designed to make the functionality of the GPU available to engineers and

[5]`http://www.hp-sales.co.uk/products.asp?partno=079232G` accessed on 29th March 2010

[6]Results from a shopping search at `http://www.google.com`, 29th March 2010.

scientists who do not have a low-level familiarity with the hardware.

2.3.1 Introduction

For many decades, Moore's law, which states that the number of transistors on a chip will double approximately every 18 months, has held true. For a long time these extra transistors and other advances in processor design offered considerable performance enhancement for conventional, single-threaded codes. However, the rate of increase of clock rates on CPUs has fallen considerably and CPU vendors are moving to new product generations which introduce more cores, rather than enhance the speed of a single core. These architectures permit running code with a degree of parallelism, but for intensely parallel problems, it is not clear that they represent an ideal solution.

GPUs offer a considerably different, and hugely parallel architecture. In recent years, the compute power of commodity, consumer GPUs has tremendously increased, such that the graphics processor is now viewed as a high performance co-processor for scientific as well as consumer game-driven calculations. The rapid increase of compute performance available with GPUs is shown in fig. 2.2, in which GPU performance is compared with CPU performance over a time period of almost 10 years.

The growth of general-purpose computing on GPUs (GPGPU) up to the year 2006 was reviewed in [21]. In that review, Owens chronicled the evolution of graphics hardware from a fixed-function pipeline where each stage was hardwired for specific tasks through several generations, each of which have been increasingly flexible and programmable, from NVIDIA's 1999 introduction of the first programmable stage, through to the third revision of the vertex shader and pixel shader standards in 2006, by which time the hardware was explicitly designed to process multiple data-parallel primitives at the same time. The history of GPU development, from early times to 2002, was also charted by Fernando and Kilgard [22].

More recent developments include the NVIDIA CUDA toolkit [19], and the ATI Stream Computing SDK [23], both of which afford considerably more flexibility to the programmer. They also dispense with the paradigms of the older GPGPU program-

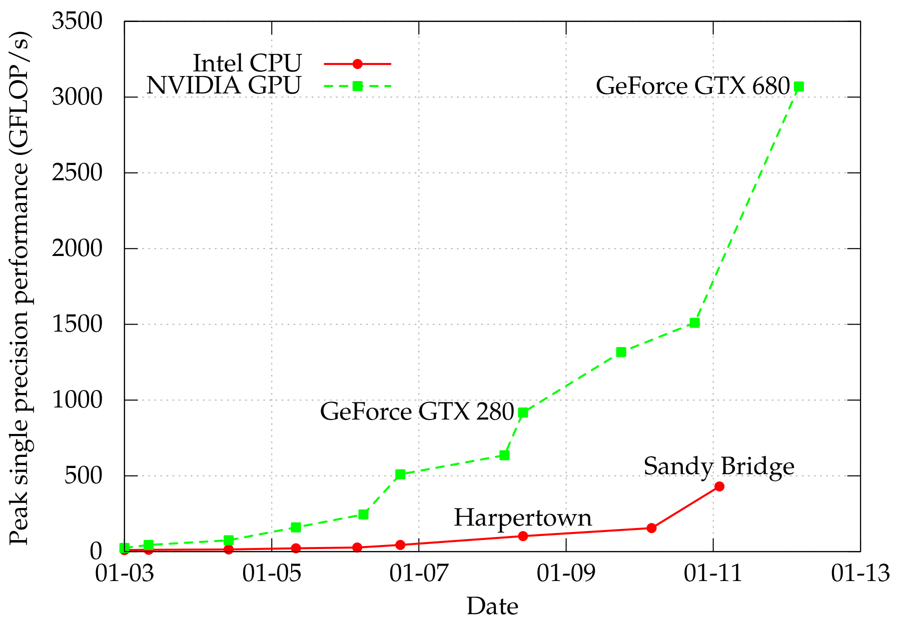


Figure 2.2: The (approximate) relative speed of consumer GPUs and CPUs from 2003 to 2012. The graph is based on data extracted from figures in [19] and [20].

ming environments, in which scientific computations of interest had to be recast into graphics terms. In 2007, solving involved scientific problems on the GPU required significant efforts by both computer graphics experts, and by specialists within the domain of the problem to be solved. The trend is towards increasing the accessibility of graphics hardware to all programmers, and lowering the entry threshold in terms of specialist graphics knowledge required.

Although GPU acceleration is becoming increasingly popular, it only achieves peak efficiency with data-parallel computations. These are problems in which the solution is found by executing the same program code on many data elements in parallel, such as when filling an array with function evaluations in which each value is independent of other calculated values. The ratio of arithmetic operations to memory operations should also be large. This is because GPUs lack large cache memories, and have a high latency for accesses to local or global memory (up to 600 cycles for both local and global memory; local memory is accessed only for some automatic variables, at the compiler's discretion [19]). When the workload is compute-intensive, threads awaiting data from memory are swapped out, and those with data present continue running, effectively masking the latency. The current generations of NVIDIA hardware are able to swap out inactive threads with zero overhead [19, pg. 72].

2.3.2 GPU programming environments

NVIDIA's CUDA

CUDA was introduced in November 2006 by NVIDIA [19]. It is a general purpose parallel computing architecture, designed to leverage NVIDIA GPUs' parallel compute engines. Initially the CUDA toolkit and development environment supported an extended C language only, but by release v2.3, it had support for NVIDIA's extended C, as well as OpenCL, DirectX Compute, and FORTRAN.

CUDA is designed to allow programs to scale dynamically to make good use of the capabilities of whatever GPU is available in the machine on which the program runs. These capabilities vary widely: as of CUDA v2.3, supported GPUs have between one and 60 multiprocessors, where each multiprocessor consists of eight processors. In order to scale the same program code efficiently to hardware with this large a range of compute ability, CUDA introduces abstractions. These consist of a hierarchy of thread groups, barrier synchronisation, and shared memories. By decomposing the problem into sub-problems, and these into threads, it is possible to schedule each sub-problem on any available processor cores. This enables run-time scaling of compiled CUDA code.

ATI Stream Computing

ATI Stream Computing is ATI's toolkit for using its GPUs for high-performance, data parallel computing. It is similar to NVIDIA's CUDA in its overall aim and structure, in that it includes software tools and drivers which expose the compute capabilities of compatible graphics hardware.

ATI Stream processors consist of programmable stream cores, which run user code expressed as kernels that operate on streams of data [23]. ATI defines a stream as a collection of data elements, all of the same type, that can be operated on in parallel.

The stream cores run kernels using a virtual SIMD model. Input data is stored in arrays in memory, and mapped onto a number of SIMD engines. The results that these generate by running the supplied kernel on the inputs are then written to output arrays. In the Stream model, threads exist in an array. This is known as the domain of execution and corresponds to a rectangular region of the output buffer to which threads are mapped. As in the CUDA model, the array of threads is scheduled onto a group of thread processors, until all the threads have been processed.

As in the CUDA model, ATI Stream Computing supports a number of higher level languages which abstract the details of the hardware from the programmer. This means that the job of the developer is restricted to providing inputs, outputs, and kernels that are to be executed over defined domains.

The ATI Stream Computing environment explicitly supports open-systems and open-platform standards [23], such that third parties can provide development tools. Particularly, it is accompanied by the Brook+ open source data-parallel C compiler. This has its roots in the Brook C-language extension from Stanford University. AMD implemented the Brook GPU specification on top of its compute abstraction layer, and added some enhancements.

OpenCL

OpenCL is a recently standardised framework designed to enable programming across heterogeneous platforms, including GPUs, CPUs, Cell-type architectures, and other parallel processors such as digital signal processors (DSPs). The specification was formalised in 2009 [24]. It is an open and royalty-free standard, and has the benefit of wide industry support, from companies such as AMD, Apple, Fujitsu, IBM, Intel, NVIDIA, and many others in the computer, mobile telecommunications, and semiconductor markets.

By using OpenCL as opposed to the proprietary CUDA environment or the open

– but vendor specific – ATI Stream platform, the developer gains the advantage of being able to target multiple vendors' hardware from a single code base, and leverage the available facilities of the heterogeneous systems on which his code may ultimately run. OpenCL combines acceleration and portability across devices and architectures. It is important to note that OpenCL defines requirements on numerical precision, which guarantees mathematical consistency across hardware of different sorts and from different vendors [25].

OpenCL is a younger technology than ATI's stream computing or NVIDIA's CUDA. There exists limited literature treating OpenCL. However, Khanna and McKennon [25] have implemented a modelling application for gravitational wave sources, which is based on a finite-difference, linear, hyperbolic, inhomogeneous partial differential equation (PDE) solver using OpenCL. They compared the results to their previous implementations which were based on native CUDA and Cell (see §2.2) SDKs, and found that OpenCL delivers comparable performance along with the advantage that from identical source code, it is possible to build programs for both architectures. They also noted that the OpenCL libraries and compilers they utilised were in beta, and likely to be significantly improved in the future.

The results of Khanna and McKennon on their Cell-based system put the performance factor of the native Cell SDK code at approximately 37, normalised to the performance of the PPE code, whilst the OpenCL implementation had a performance factor of approximately 34. That is, the Cell SDK code running on the Cell-based system performed at 37× the speed of the same code running on the Cell's PPE. They noted that the OpenCL must use a pre-compiled kernel in order to demonstrate this speedup, since when they elected to dynamically compile the OpenCL kernel, the performance was approximately equal to the unaccelerated PPE, the runtime being dominated by the compilation stage.

Their results in comparing OpenCL with CUDA on a Tesla S1060 GPU were even more favourable for OpenCL; the CUDA SDK and OpenCL with pre-compiled kernel achieved an identical speedup of approximately 26 times compared to their baseline,

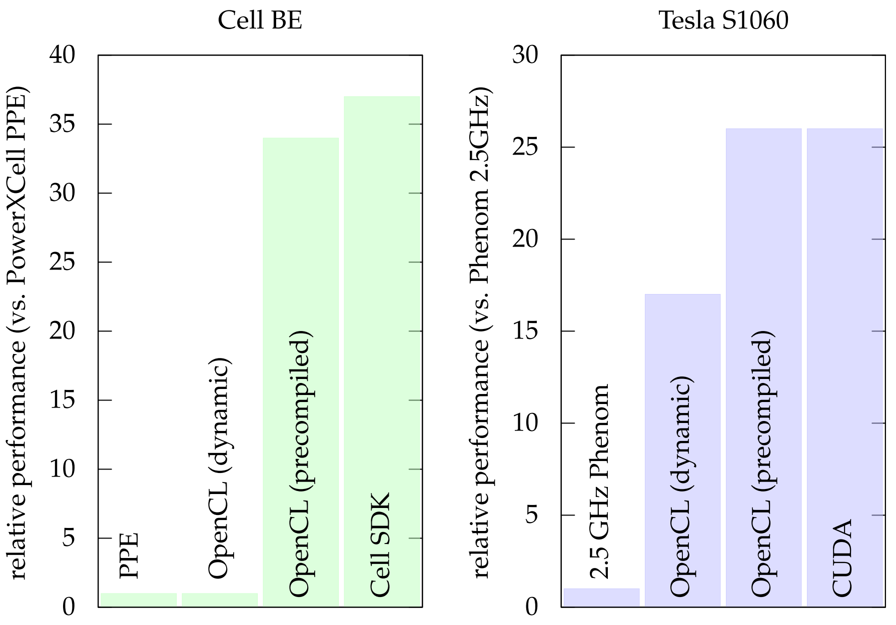


Figure 2.3: Illustration of the approximate speedup obtained for a gravitational wave modelling PDE solver on an NVIDIA Tesla S1060 accelerator and a Cell Broadband Engine processor, using OpenCL and the native Cell/CUDA SDKs. Data from [25].

chosen to be the system's 2.5 GHz AMD Phenom processor. In this case, the OpenCL implementation with kernel left as source code for runtime compilation also provided a significant speedup compared to the baseline, by a factor of approx. 17 times. Moreover, the performance gains that their code achieves overall on the Cell and the GPU are comparable, but the GPU is considerably cheaper; thus, in this application the GPU offers a considerably better price/performance ratio. The results of Khanna and McKennon are summarised in fig. 2.3 and in table 2.3 (in which the prices used are the list prices quoted in the reference).

2.3.3 Higher-level GPU interfaces

There also exist a number of higher-level programming interfaces to GPUs, which aim to expose the power of GPUs to programmers in engineering and scientific fields who do not wish to be concerned with the lower-level details of programming the GPU.

	relative price	relative performance	price-to-performance
Phenom 2.5 GHz	1	1	1
Cell BE	5	25	0.2
Tesla S1060	1.5	26	0.06

Table 2.3: Relative price, performance, and price-performance characteristics of the Cell Broadband Engine and Tesla S1060 accelerators, all compared against a 2.5 GHz AMD Phenom CPU. Data from [25].

Here we summarise two of the better known such interfaces, with very brief code examples to illustrate the potential simplicity of employing these accelerator technologies.

Jacket

AccelerEyes produces a software product called Jacket. This facilitates access to the GPU as a compute accelerator within MATLAB. Jacket is specific to CUDA, and therefore to NVIDIA's GPU lineup. Although it is possible to use MATLAB's MEX facility to compile and link external code, such as C for CUDA programs, and access them within MATLAB, Jacket eliminates the need for the user to know any specifics of GPU programming. The code example that follows, taken from [26], illustrates this with a simple example:

Jacket for MATLAB Example

```
A = eye(5);         % Creates 5x5 identity matrix
A = gsingle(A);     % Casts A from CPU to GPU as single precision
A = A * 5;          % Uses GPU to multiply matrix by scalar
A = double(A);      % Casts A from CPU to GPU
```

As shown, the use of the GPU is transparent to the user; the `gsingle()` command is the only one which deviates from the standard MATLAB language. The Jacket toolbox gives real-time, transparent access to the GPU which maintains the interpretive nature of the MATLAB language [27]. The same straightforward syntax allows the acceleration of user-defined functions, so the above example is not constrained to simple operators: if the arguments of a user function have been cast onto the GPU, Jacket will attempt to accelerate that function when the function is called.

PGI Compilers

Another commercial offering in the GPU programming sector is from the Portland Group (PGI); their FORTRAN compiler incorporates support for accessing CUDA functionality via extensions to the FORTRAN language. Because of the huge number of existing HPC codes written in FORTRAN, this is an attractive option to developers who wish to harness some of the power of the GPU as a computational accelerator without incurring the effort and time penalties associated with porting large code bases to C. PGI also offer support for GPU acceleration in their C compiler.

The CUDA FORTRAN compiler exposes much of the CUDA language as a small set of language extensions [28]. This allows as much fine-grained control of the parallelisation as may be necessary, and is comparable to C for CUDA. However, PGI also offer the PGI Accelerator “Kernels” programming model [29]. The PGI Accelerator Programming Model shares with OpenCL the goal of avoiding the need for programmers to maintain different versions of a project that must be able to target both x64 CPU-only systems and those with GPUs suitable for accelerating the code. However, they seek also to avoid the labour of converting the existing code into host and device portions, and possibly also of having to change the language used.

The Accelerator Programming Model is implemented in PGI FORTRAN and C compilers as a set of directives, similar in appearance as OpenMP directives. The resulting program, PGI claims [29], will run unmodified on the CPU alone, or on the CPU and GPU in concert.

One of the most trivial examples that illustrates the syntax is shown below, taken from an example in [29]. This FORTRAN code assigns values to the `n`-element vector `a(i)`. It doubles the vector both on the GPU, storing the result in `r(i)` (lines 4-8), and on the CPU (lines 9-11), storing it in `e(i)`, and raises an error if the values calculated do not agree. The simplicity of specifying the region for GPU acceleration is evident; it only required the use of the `!$acc region` and `!$acc end region` directives enclosing the loop. The compiler handles all the required steps to set up and use the GPU, such as copying data to and from the GPU, and working out where parallelisation is possible. It should be noted that other compilers would treat the `!$acc...` directives as comments and therefore the code remains portable.

PGI Accelerator Programming Example

```
1  do i = 1,n
2      a(i) = i*2.0
3  enddo
4  !$acc region
5      do i = 1,n
6          r(i) = a(i) * 2.0
7      enddo
8  !$acc end region
9      do i = 1,n
10         e(i) = a(i) * 2.0
11     enddo
12 ! check the results
13 do i = 1,n
14     if( r(i) .ne. e(i) )then
15         print *, i, r(i), e(i)
16         stop 'error found'
17     endif
18 enddo
19 print *, n, 'iterations completed'
```

Of course, this is not a 'silver bullet,' and there is skill required in utilising it effectively. The algorithms within accelerated regions must be appropriately expressed in a suitably data-parallel form. For example, an assignment statement in which the values being assigned appear on the right hand side is not parallelisable, whereas when they do not, it is. Other conditions for successful parallelisation include the independence of the iterations of loops. These conditions are general to all parallel programming on GPUs, but they are highlighted here to clarify that even applying the relatively high-level Accelerator Programming Model requires specific knowledge and careful consideration of the principles of parallel programming.

2.4 Cloud computing

In this section we introduce the paradigm of cloud computing, and discuss its applicability to the challenges of scientific computing. In contrast to the previous sections, which all dealt with tangible pieces of hardware that may be precisely and completely specified, and which would usually be purchased and commissioned by an individual or an organisation, cloud computing is a broad concept that is expressed in several different forms and many implementations.

2.4.1 Introducing cloud computing

One commonly quoted definition of cloud computing is from NIST [30]; it begins with the following summary of the essential elements that are present in all cloud computing offerings:

> cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (*e.g.*, networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

It goes on to detail the essential characteristics, service models, and deployment models that compose the NIST cloud model.

Cloud computing has been described as “the next stage in the evolution of computational and data handling infrastructure,” [31]. A key feature of cloud computing is that it provides a dynamically scalable compute resource, available on-demand with a so-called utility pricing model [32]. Under this model users are charged on the basis of the amount of resources they actually consume. Available resources are typically so large as to be limited by the user’s budget rather than the amount of hardware that a provider possesses. Therefore, as computational problems outstrip the capabilities of the machines or clusters available locally, and increasing amounts of resource are demanded, cloud computing can offer a solution.

Cloud computing is not the only suitable technology in such situations; grid computing [33], and other distributed computing and data-processing efforts seek to fulfil similar requirements. Indeed, cloud computing could be seen as an evolution of the grid.

Cloud computing is provided commercially by several vendors, such as Amazon[7], Google[8] and Microsoft[9]. They offer customers the ability to provision resources rapidly (on the order of minutes), and typically have short minimum rental periods (at the time of writing, this is one hour for Microsoft Windows Azure). The providers buy hardware in bulk and benefit from economies of scale that are almost impossible for smaller companies or institutions to achieve. They also optimise administration practices, which minimises their costs. At the time of writing, a small range of different hardware specifications are available from each of the major providers, so that users may request appropriately-sized machines. Thus, cloud-based applications can usually be scaled “up” (by moving to larger hardware) or “out” (by moving to more hardware instances) when higher throughput is required.

Cost to the user is reduced because idle resources can be rapidly returned to the provider, for subsequent provisioning to other customers, and cease incurring costs. This is in stark contrast to local ownership of hardware (either of the types of accel-

[7]`http://aws.amazon.com/ec2/`

[8]`http://cloud.google.com/`

[9]`http://www.windowsazure.com/en-us/`

erator hardware described in the preceding sections, or of generic servers or workstations) where an asset, once purchased, is typically difficult or impossible to return to the supplier, and continues to consume space and power even when it is underutilised.

Classes of cloud service

Three main classes of cloud service are available commercially [34]. These are software-as-a-service (SaaS), platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS). These correspond to the NIST service models [30].

SaaS is a model in which the user procures from the provider a license and facility to use some piece of software. Code for that software is not kept on the user's machine, nor is data stored there. However, in examples such as Microsoft Office Live, the code may run in a web browser on the user's machine, and it may be possible for the user to download copies of the data. SaaS offerings can be built on top of the PaaS or IaaS models.

PaaS is similar to the SaaS model, but lower-level, in that the cloud provider establishes a programming interface against which the end-user or other organisations can develop their own programs to run on the cloud provider's hardware. Access will, again, be over the network, and the experience for an end-user on the desktop may resemble that for SaaS. A typical example of PaaS software is Microsoft Windows Azure, which provides an OS and libraries (the .NET framework) upon which customers' workers are run. PaaS offerings can be seen as being built upon the IaaS model.

IaaS is the lowest-level of the common cloud models. In this case, the cloud provider offers infrastructure such as networking, servers or virtual machines, and it is up to the developer or end-user to provide the operating system, libraries, and user applications. Amazon's EC2 service allows users to rent virtual machines in this manner.

At one end of the scale, SaaS puts the vast majority of the administrative control (and burden) on the cloud supplier. They are responsible for setting up the software that is made available, applying updates when required, and maintaining the service security. In stark contrast, IaaS puts these responsibilities into the hands of the developer rather than the cloud provider. This offers more flexibility and control, but at the cost of having more complexity to manage. PaaS offerings strike a balance where application updates may be pushed out by the customer, but the platform itself may be maintained by the cloud provider (using virtual machine "live migration" techniques, it will be possible for them to update the host OS, and by upgrading various guest instances asynchronously, it is possible to patch any part of the software stack without loss of service).

Cloud deployment models

Cloud services may be deployed on several different models, which differ in their intended user community and provider. The US NIST breaks the models down as follows [30]:

Public cloud is an infrastructure which is made available for use by the general public. Such infrastructure may be provided by business, government, or academia, and is physically located upon the premises of the provider.

Private cloud is an infrastructure for the exclusive use of one organisation but typically shared between business units of the organisation. Such infrastructure may be located on or off the organisation's premises and owned and managed by the organisation, an external party, or some combination thereof.

Community cloud is where a cloud infrastructure is used by a community of consumers form various organisations who share certain requirements (such as security or policy). These clouds are either on- or off-premises and may be managed by one of the organisations from the community served, a third party, or a combination thereof.

Hybrid cloud schemes arise where at least two distinct cloud infrastructures are somehow bound together. The constituent clouds remain separately identifiable, but data and applications are made portable across the clouds. This facilitates migration of jobs between clouds, e.g. for intra-cloud load balancing.

2.4.2 Areas of strength for cloud-based architectures

Just as GPUs and other accelerator technologies discussed previously have specific areas of strength and areas of weakness, so does cloud computing. The strengths of cloud computing may be divided into four key areas [31]:

Algorithm development is accelerated by the ability to procure appropriate types of hardware on-demand and almost instantly when required. It is possible for an algorithm developer to rent powerful machines, or large numbers of machines to be run in parallel, for the time it takes to assess the performance or validity of a particular implementation of his algorithm. In cases where the input or output data sets for the test or validation process are very large, cloud providers can supply storage that is near the compute, in order to minimise time spent in I/O routines. It should also be noted that using cloud-based architectures in scientific computing in general encourages modular designs, which simplifies the substitution and comparison of alternative algorithms [35]. However, at the time of writing, the dedicated high-performance interconnects found in typical supercomputers are not commonly available in commercial cloud computing offerings, as will be discussed in §2.4.3.

Data dissemination is often a slow and possibly expensive process, especially when large volumes of data are to be copied to multiple locations. When data is stored in a cloud service, however, it is possible to grant data access permissions to collaborators without needing to transfer the data to them. They will be enabled to perform further analysis or computation on the data using compute resources located near the data. There may in future be an expectation that published work would be required to make its underlying data available [36] so that readers can confirm findings, and cloud based data storage naturally facilitates this.

Burst capability is considerably better for cloud-based solutions than for traditional, data centre-based architectures. Data centres may be scaled to cope with estimated peak demands, but this may leave hardware idle for the majority of the time; otherwise, peak demands may be under-resourced, reducing quality of service. Cloud-based architectures, though, allow the provisioning of a massive amount of resources very quickly to cope with demand peaks. In algorithm development work, a burst of demand may be given rise to each time a new version of the algorithm is to be tested.

Scalability is inherently available in cloud-based architectures, because of the huge amount of hardware owned and operated by cloud suppliers. The utility pricing model ensures that for solutions which begin small and later need to scale, undue costs are not incurred.

2.4.3 Scientific computing in the cloud

A variety of investigations have been performed into the suitability of cloud computing for scientific and engineering applications. By 2008 it was accepted that cloud-based computing could offer “a feasible, cost-effective model in many application areas” [37] where traditional high-performance computing systems might otherwise be used. However, the type of problem to be solved, and its attendant level of parallelisability and communication requirement, is a massive determining factor in whether a cloud-based solution will be an attractive proposition.

Applications such as parameter sweeps, where the algorithm used can take effective advantage of all the cores in one node, and where inter-node communication is low, are ideal candidates for processing on the cloud. However, those tasks that make heavy use of communications between nodes are less well suited, because, typically, cloud providers currently utilise 1 Gbit s^{-1} Ethernet between nodes. This interconnect features both a higher latency (60–125 μs) and a lower bandwidth (45–50 MB s^{-1}) than a specialist interconnect that may be used on a traditional HPC system, such as InfiniBand Architecture, whose latency and bandwidth were measured in one study as 13–17 μs and 560–610 MB s^{-1} respectively [38]. There have been some technological and business developments since the cited study, and indeed some cloud providers will now provision nodes with 10 Gbit s^{-1} Ethernet interconnects, but at the same time, faster specialised interconnects are now available for top-end HPC clusters. It is still the case that cloud computing thrives best on commodity hardware in homogeneous data centres which power its economies of scale, although it is expected that the most common of specialised hardware requirements will eventually be satisfied by cloud providers [39].

One example of a scientific workload poorly suited to cloud computing was given by Napper and Bientinesi, who benchmarked cloud performance for dense linear algebra loads [40]. This class of application relies heavily upon the high-speed interconnects and available memory on a traditional HPC cluster, so it is unsurprising that they concluded that a cluster of any number of Amazon EC2 nodes would not be able to claim a place in the TOP500 list. Moreover, they analysed how the cost of computation scales for their test load, and found that their throughput, measured in Gflops returned per dollar spent, dropped exponentially with increasing core counts – implying an exponential increase in the cost of solving linear systems with the problem size. The authors of that study explain that this exponential decrease in performance is easily attributed to the communications taking place over the slow interconnect.

Cost-effectiveness is difficult to evaluate without knowing the specifics of any given application, and it can also be difficult to assign accurate costs to the alternatives, such as an in-house compute cluster. Hazelhurst noted [37] that for scientists as end users, large shared clusters often have no direct cost, but access can depend on successful research proposals. He compared the US$2700 cost of a local server (neglecting electricity, cooling, maintenance and space overheads) to the US$0.20 per hour cost of a comparable Amazon EC2 instance. The purchase price of the local machine would fund approximately 1.5 years of continuous processing on EC2, so, assuming the local server to have a 3-year lifetime, he suggested that a local machine with utilisation below 50% would be more economically replaced by a cloud-based system; higher utilisations, for this scenario, would be more economically served by local machines. However, this analysis neglects the potential costs of data storage and transfers in the cloud.

A more comprehensive analysis of the costs of another scientific application hosted locally and in the cloud was carried out by Berriman, *et al.* in 2010 [41]. The sample workload here was running montage, an image-processing application that assembles images in the astronomical data format FITS (flexible image transport system) into custom mosaics. The cost effectiveness was calculated on the basis of having a local and a cloud-based service, each of which would be called upon to service requests for 36000 mosaics of data from the Two Micron All Sky Survey. Including the costs of power, administration, cooling, and support contracts, the local service would return a cost of US$0.64 per mosaic whilst a similar service in Amazon's EC2 cloud would cost US$1.46 per mosaic.

Although the work by Berriman *et al.* referred to above found that a local hardware resource may offer lower costs, it is not possible to reach a conclusion on the question of whether cloud-based architectures are more or less suited for scientific and engineering computations in general, because there are many classes of problem and many possible implementations of solutions. Moreover, it should be noted that many sample applications in the literature don't align particularly well with the strengths of cloud-based architectures given in §2.4.2. In cases where these strengths are better utilised, however, cloud computing can offer results that are significantly difficult or highly costly to achieve by other means [42, 35].

2.5 Summary

This chapter was devoted to discussing the technological advances made in computational technologies and how they may be applied to scientific and high-performance computing. Initially special-purpose hardware for scientific computation was discussed, before considering a recent and growing trend towards scientific computing on graphics processors (GPUs). Considerable momentum is gathering in respect of utilising consumer GPUs for their unparallelled price-performance ratio. Higher-level interfaces to the accelerator hardware were also discussed. The chapter closed by introducing another technology that has recently been attracting considerable attention, namely cloud computing. We introduced the ideas of cloud computing, described various types of cloud computing service that are available, and considered some studies that have attempted to assess the merit of cloud computing for scientific applications. The introduction to cloud computing given here underlies the application of cloud computing to the process of developing and verifying the meshless local weak-strong form method as described in §6.4 and [43].

Chapter 3

Fast multipole methods

3.1 Introduction

In this chapter, we introduce the N-body problem and discuss methods by which it may be solved. We point out the limitations of the naïve approach, and discuss schemes whereby the computation time is reduced. These methods include particle-in-cell (PIC) methods, tree codes, and the fast multipole method (FMM), which we describe in detail. Additionally, in certain work, special-purpose hardware has been exploited in order to accelerate the N-body problem; §2 provides an overview of this, including machines such as some of the GRAPE series which were developed specifically for the N-body problem, and technologies which allow modern graphics processing units (GPUs) to run highly parallel general-purpose algorithms.

In many physics and engineering problems, it is required that we consider the dynamics of large numbers of particles interacting via long-range forces. This is known as the N-body problem. For $N > 2$ there is no general analytic solution to this class of problem [44]. The naïve numerical solution is to interact each particle directly with the $N - 1$ other particles in the system, and sum the contributions of every interaction. The compute time scales as $\mathcal{O}(N^2)$ in these naïve codes, which severely limits the size of system that may be simulated.

It should be noted that there have been many simulations in which the long range nature of the Coulomb force has been neglected by truncation beyond some set distance. These sorts of simulations are popular in molecular dynamics, where there are many complicated short-range interactions to account for, as well as the infinite-range Coulomb interactions [45]. Truncation ranges are typically of the order of 8 Å to 20 Å [46], and in certain simulations where the many other forces involved are accurate only to a few significant figures, this approach is successful. However, such an approach is inappropriate for modelling the results of the long-range interaction itself, such as in astrophysics. It can also give rise to problems when applied naïvely to certain systems, such as layered high temperature superconductors [47]. Therefore, a strong motivation has long existed to pursue efficient methods of computing such simulations without introducing an artificial cutoff.

3.2 Particle-in-cell methods

Many particle-in-cell (PIC) methods are applied in cases where the potential satisfies Poisson's equation. This equation takes the form (3.1), where the Laplace operator is Δ, and f and ξ are real- or complex-valued functions on a manifold:

$$\Delta\xi = f. \tag{3.1}$$

The Newtonian gravitational potential may be shown to satisfy the Poisson's equation for gravity,

$$\nabla^2\phi_{\mathrm{g}} = 4\pi G\rho,$$

where on 2D Euclidean space we write Δ as ∇^2 and we have introduced the scalar gravitational potential ϕ_{g}, the density ρ of the massive object that gives rise to the gravitational potential, and G, the universal gravitational constant. It can also be shown that a similar relationship holds in electrostatics:

$$\nabla^2\phi_{\mathrm{e}} = -\frac{\rho_{\mathrm{f}}}{\varepsilon},$$

where we have labelled the electrostatic potential as ϕ_{e}, and introduced the free charge density ρ_{f} and the permittivity of the material ε. It can also be seen that the potential arising between stacks of pancake vortices in layered high-temperature superconductors, (4.1), which will be discussed in §4.1, is of the same form as (3.1).

In these PIC methods, a grid is superimposed upon the simulation domain, and the particles of the system contribute their charges, masses or other properties of interest to a density field. At the grid points, this density is evaluated and the potential is then calculated at each grid location. A fast Poisson solver, of which many examples can be found, *e.g.* [48, 49, 50], can be used to reduce computational overheads of this stage [4]. The solution is then interpolated back to the positions of the individual particles [44]. When the force is required as well as the energy, it is calculated before the interpolation.

For N particles in a system which has M grid points, the computational costs of the PIC method scale as $\mathcal{O}(N + M \log M)$. The $\mathcal{O}(N)$ part of the scaling is given rise to by the processes of setting up the field, and of interpolating the solution; in both cases these require calculations involving each particle once. The fast Poisson solver typically utilises FFT techniques to achieve a solution at the M grid points in $\mathcal{O}(M \log M)$ time.

The PIC method can therefore be efficient, because in many scenarios it is usual to have $M \ll N$, so that the time required scales approximately proportionally to N. However, the PIC method has some considerable weaknesses. Especially in systems with highly nonuniform particle distributions, the mesh spacing limits the resolution and thus accuracy attainable. Moreover, if there are strong local particle-particle interactions within cells of the mesh, these are smoothed away when the potential is evaluated at the mesh points. These qualities make it unsuitable for modelling many gravitational systems, where there are considerable effects from local interactions in a highly nonuniform overall structure.

3.2.1 Particle-particle/particle-mesh (P^{3}M) methods

It is possible to improve the accuracy of PIC methods by directly calculating the short-range interactions, and using the mesh technique described above for the long-range interactions. Such methods are known as particle-particle/particle-mesh (P^{3}M) methods [51]. These algorithms can provide theoretically arbitrary precision. However, in cases where the requested accuracy is high or the particle distribution is not close to uniform spacing in a rectangular region, the time required by these algorithms can become excessive [4].

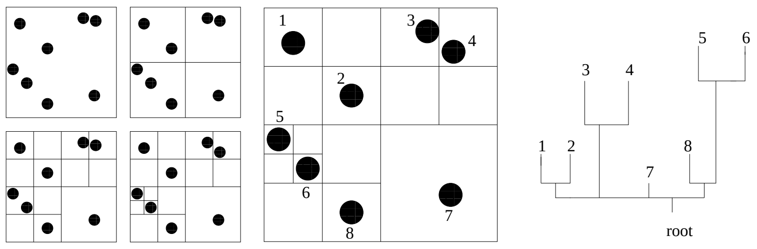
3.3 Tree codes

An improvement on PIC codes is to be found in tree codes. These algorithms make use of a hierarchical tree data structure from which information about the separation of particles may be inferred without explicit distance calculations. The interactions with nearby particles are calculated directly as particle-particle interactions. More distant particles are grouped into clusters and the influence of the group is represented by a single, particle-group, interaction.

The tree codes achieve their results in $\mathcal{O}(N \log N)$ time. The FMM codes discussed in §3.4 are based upon similar fundamentals as the tree codes, but can achieve $\mathcal{O}(N)$ run times.

Although various independent researchers introduced different types of tree codes in the early 1980s [44], the most influential and widely implemented scheme is that proposed by Barnes and Hut [52], which uses an octree in 3D space. In their algorithm, particles are added one by one to the simulation cell. As soon as more than one particle is inside any cell, that cell is divided into eight daughter cells, each having half the width, depth, and breadth of its parent. If more than one particle remains inside any cell, it is recursively subdivided until at most one particle is located within each cell. This process is repeated for each particle that is added. A two-dimensional example is shown in fig. 3.1(a). In this 2D case, each subdivision gives rise to four daughter cells rather than eight. Shown in fig. 3.1(b) is the tree structure resulting from such a subdivision process.

The work in creating a tree of N particles is of $\mathcal{O}(N \log N)$, which can be verified by considering the average volume of a finely-divided cell, which is given by V/N,



(a) Subdivision of space (b) Tree structure and corresponding spatial divisions

Figure 3.1: Division of space and creation of tree structure to track nodes, as user in Barnes and Hut's algorithm. Shown in 2-D for clarity. Based on figures in [44].

where V is the volume of the entire simulation cell. Such cells will have an average edge length which is some power x of $V^{1/3}/2$, where x is the height of the tree, so we have [44, §2.1]:

$$\left(\frac{1}{N}\right)^{1/3} = \left(\frac{1}{2}\right)^{x},$$
$$\implies x = \frac{1}{3\log 2}\log N \approx \log N. \tag{3.2}$$

On average, from the root of the tree, there are $\log N$ divisions to be made to reach a cell containing only one particle. Since there are N such particles, total effort is $\mathcal{O}(N \log N)$.

It has been shown [44] that the average work required to calculate the interactions scales as $\mathcal{O}(N \log N)$ also, so that the method overall scales as $\mathcal{O}(N \log N)$.

3.4 The 2D Greengard-Rokhlin FMM

In 1987, Greengard and Rokhlin introduced a new scheme to solve the N-body problem, with an asymptotic runtime estimate of $\mathcal{O}(N)$ [4]. It has since been named one of the top 10 algorithms of the 20th century [46, 3].

The ideas behind the FMM were perhaps first conceived in literature on protein simulation [53], in which Pincus and Scheraga observed that when simulating proteins, it is usual practise to neglect long-range interactions beyond some arbitrary cutoff distance. However, although every such interaction is of small magnitude, there are many of these interactions, so that their net effect need not necessarily be insignificant. They go on to propose a method in which, beyond an arbitrary cutoff distance, interactions are no longer computed directly. They introduce computation of interactions between two spheres, where the charge distribution of the molecules being simulated is represented by two interacting dipoles rather than individual particles.

Such approximations give rise to energies comparable to those computed from all pairwise interactions. The authors note that their method results in substantial reductions in computational times when compared to full direct simulations.

In the FMM, the first key idea is the process of replacing distant groups of particles by pseudo-particles, whose properties – such as net charge, dipole moment, and quadrupole moment – are identical to those of the group of particles represented. These properties are represented as a multipole series, which converges rapidly so that retaining between three and eight terms typically give good accuracy [46].

The FMM also employs the hierarchical division of space as in the Barnes-Hut tree code, discussed in §3.3. The creation of the tree and the interaction of particles with other particles and pseudo-particles where appropriate has a computational complexity of $\mathcal{O}(N \log N)$ but the FMM introduced a new concept, which reduces the algorithm's complexity to $\mathcal{O}(N)$. The new concept is the local expansion, whereby the interaction of distant groups of particles is calculated with groups of target particles at once. Multipole expansions are used in representing both the target particles and the distant group. The calculation of an interaction between the groups then consists largely of a single convolution of arrays containing the coefficients of the appropriate multipole expansions [46].

Greengard and Rokhlin introduced several lemmas which provide the mathematical foundations of the FMM, developing a fast method for use in two-dimensional systems governed by the electrostatic or gravitational potentials [4]. To demonstrate the advantage of using multipole expansions in calculations with potential fields, they

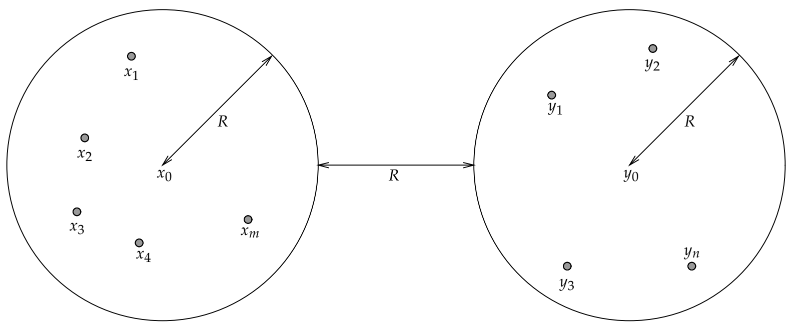


Figure 3.2: Well-separated sets of particles, after [4].

gave the following example, which is illustrated in fig. 3.2.

They placed the charges q_1, q_2, ..., q_m at the points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_m \in \mathbb{C}$, with $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, ..., $\boldsymbol{y}_n$ being other points in $\mathbb{C}$ (we adopt the notation that a vector $\boldsymbol{v} = (v_0, v_1)$ may be represented as the complex number $v_0 + \mathrm{i}v_1$, where $\mathrm{i} = \sqrt{-1}$). If the conditions of (3.3) are met, where R is the radius of circles centred at $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$ containing the sets $\{\boldsymbol{x}_i\}_{i=1}^{m}$ and $\{\boldsymbol{y}_j\}_{j=1}^{n}$ respectively, as illustrated in fig. 3.2, then the sets $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_j\}$ are described as well-separated.

$$\begin{aligned} |\boldsymbol{x}_i - \boldsymbol{x}_0| &< R \quad \forall \quad i = 1, ..., m \\ |\boldsymbol{y}_j - \boldsymbol{y}_0| &< R \quad \forall \quad j = 1, ..., n \\ |\boldsymbol{x}_0 - \boldsymbol{y}_0| &> 3R \end{aligned} \tag{3.3}$$

Naïvely one could compute the potential at the points y_j by evaluating

$$\sum_{i=1}^{m} \phi_{x_i}\left(\boldsymbol{y}_j\right) \quad \forall \quad j = 1, ..., n, \tag{3.4}$$

where $\phi_{x_i}\left(\boldsymbol{y}_j\right)$ is the contribution at the point $\boldsymbol{y}_j$ due to the source at $\boldsymbol{x}_i$. This evaluates m fields at n positions, so it requires $\mathcal{O}(mn)$ work.

A speedup comes from evaluating a p-term multipole expansion about the point $\boldsymbol{x}_0$ for the potential due to the charges $q_1, q_2, ..., q_m$, expending work of order $\mathcal{O}(mp)$. The expansion can be evaluated at the n points $\boldsymbol{y}_j$ with $\mathcal{O}(np)$ work so that the total work is reduced to $\mathcal{O}((m+n)p)$.

The expansion of the two-dimensional electrostatic potential $\phi(z)$ at some location z due to these charges $\{q_i\}_{i=1}^{m}$ is given in exact terms in (3.5),

$$\phi(z) = \sum_{i=1}^{m} q_i \log(|z|) + \sum_{k=1}^{\infty} \frac{a_k}{z^k},$$
$$\text{where} \quad a_k = \sum_{j=1}^{m} \frac{-q_j z_j^k}{k}, \tag{3.5}$$

but Greengard and Rokhlin also derived an uncertainty bound on the truncation of this expansion to p terms, as shown in (3.6).

$$\left| \phi(z) - Q \log(|z|) - \sum_{k=1}^{p} \frac{a_k}{z^k} \right| \leq \sum_{i=1}^{m} |q_i| \left(\frac{1}{2}\right)^p, \tag{3.6}$$

with

$$Q \quad = \quad \sum_{j=1}^{m} q_j.$$

If one demands a precision ϵ relative to the overall total charge, it can be shown [4] that p should be of order $-\log_2(\epsilon)$, and specifying the precision leads to a computation time of $\mathcal{O}(m) + \mathcal{O}(n)$.

Greengard and Rokhlin then went on to furnish lemmas which allow manipulation of multipole expansions. Specifically, they provided for the shifting of the centre of a multipole expansion, the conversion of a multipole expansion to a local expansion, and for shifting the centre of a Taylor expansion. They derived error bounds for the translation operators.

They presented their FMM algorithm using 2D examples. They first introduced a hierarchy of meshes that divide the computational domain into small boxes. The levels are labelled such that level 0 is the entire domain, and level $l + 1$ is derived from level l by subdividing each box into 4 equal sub-boxes. Thus, at level l, there are 4^l boxes. They used a tree structure in which the children of a box i are the the boxes obtained when i is subdivided. Interaction lists were created for each box i, which contain those boxes that are children of the nearest neighbours of i's parents, but which are well-separated from i.

The steps of the algorithm are then as follows:

1. Form multipole expansions for each box, about the box centre, representing the potential due to the particles in the box.

2. Translate these multipole expansions up the tree, forming multipole expansions about the centres of all boxes at coarser levels, using the translation operator and adding so that each expansion represents all the particles within the box.

3. Starting from the coarsest level, form a local expansion about the centre of each box, at each refinement level. These expansions represent the field due to all particles not in the box or its nearest neighbours. Expand the local expansions about the centres of each child box to form the expansions at the next finer level.

4. At the finest mesh level, for each box i, form a local expansion about the centre of box i from the multipole expansion of each box j in the interaction list of box i. The final local expansion is the sum of all appropriate local expansions, and describes the potential field due to all the particles in boxes other than i's nearest neighbours.

5. For each box i, and for each particle in that box, evaluate the potential due to the well-separated particles via the local expansion. Evaluate interactions with other particles inside the same box and its nearest neighbours directly. Sum the direct and far-field terms.

None of the stages above require work proportional to any power of N greater than unity. Greengard and Rokhlin estimated the total running time to be

$$N\left(-2a\log_2\epsilon + 56b\left(\log_2\epsilon\right)^2 + 4.5dk_n + e\right),$$

where k_n is a bound on the number of particles per box at the most-refined mesh level; a, b, d, and e are constants dependent on the implementation and computer system, *etc.*

Results of the Greengard-Rokhlin implementation of the FMM demonstrate clearly that the error bounds required are satisfied, and that the speedup is significant. Indeed, for a simulation with 6,400 particles in the unit cell, the FMM took 24.7 seconds, with the largest relative error being 7.2×10^{-5}, compared to the direct summation in single precision that took 4480 seconds, with a maximum relative error of 6.8×10^{-5} (where the errors stated for the FMM and the single precision direct calculation are relative to the results of the direct calculation in double precision). The FMM in this case gave rise to a 180× speedup compared to the single precision direct method, and achieved a similar accuracy [4].

3.5 Enhanced FMM algorithms

Since the original FMM implementation, a 2D-only code for gravitational and electrostatic potentials [4], many researchers have contributed enhanced versions of the algorithm, widening its applicability considerably. This section draws attention to a small selection of these enhanced FMM algorithms.

Carrier, Greengard and Rokhlin introduced an adaptive version of the FMM soon after the publication of Greengard and Rokhlin's original work [54]. The adaptive version is an $\mathcal{O}(N)$ algorithm, but unlike that described in their original paper, it does not depend upon the statistics of the particle distribution for its efficient performance. For a nonuniform distribution of points, this algorithm imposes a hierarchy of meshes rather than using a uniformly spaced mesh. This was achieved by altering the subdivision process so as to only subdivide those boxes containing more than some fixed number s of particles. Any empty boxes generated are immediately forgotten, to save memory and because they are of no use. It is concluded that the scheme is more efficient for nonuniform than uniform distributions. Both the storage and time requirements are found to scale as $\mathcal{O}(N)$.

Elliott and Board introduced a faster version of the Greengard-Rokhlin FMM algorithm in 1996 [55]. They applied FFT acceleration techniques to the convolution-like calculation in the FMM and achieved runtime savings dependent upon the value of p, the number of coefficients retained. For $p = 8$ on a single processor their speedup was 2; on a vector processor they achieved a speedup of 6 for $p = 16$. Their enhancements reduced the p-dependence of the time scaling of the FMM from $\mathcal{O}(p^4 N)$ to $\mathcal{O}(p^2 \log_2(p) N)$.

In their 1997 paper, Greengard and Rokhlin introduced a new version of the FMM, which utilises new diagonal forms of the translation operators to achieve high accuracy in 3D with an acceptable cost [56]. They pointed out in their introduction that after about ten years in research, many FMM schemes had appeared in the literature, but for 3D applications there were few schemes offering acceptable accuracy and computational costs. The new algorithm was later described as "highly efficient over a wide range of accuracies" [57]. The mathematical apparatus involved in the new method, however, was considerably more involved than that usually used in the design of fast multipole algorithms. The work nevertheless brought high accuracy 3D calculations within practical reach.

FMMs have also been applied to the Maxwell equations, where the FMM may be used to calculate fast matrix vector products and to optimally compress the matrix [58] when analysing the scattering of harmonic plane waves from perfectly-conducting obstacles. The application of the FMM leads to a large speedup in runtime and a reduced memory usage, which means that problems of unprecedented size may be tackled. In this application, the formulation consists of computing the currents on the surface of the scattering object. A Galerkin method is applied and the resulting linear system can be solved iteratively. During the iterative solution, the FMM calculates fast matrix-vector products. Another researcher applied an FMM to the problem of evaluating the parasitic capacitance of the microstrip signal lines that are used to carry microwave signals above stratified dielectric media [59]. The algorithm developed for this application retains the FMM's characteristic $\mathcal{O}(N)$ scaling in time and memory.

Still other FMMs that have been developed are known as the "black box," or kernel-independent methods, because unlike the methods developed for specific kernels (such as the Newtonian kernel $\frac{1}{2\pi} \log \|\boldsymbol{x}\|$ which applies to gravitational and electrostatic potentials), they use schemes such as interpolating between kernel evaluations via Chebyshev polynomials to achieve independence of the kernel. Therefore, such methods have application to arbitrary non-oscillatory potentials, which need only be numerically obtainable for all the required coordinates, so it is not always necessary to have even an analytical representation of the potential [60, 61].

3.6 Summary

In this chapter the N-body problem was introduced. The naïve $\mathcal{O}(N^2)$ solution was considered, and a summary was provided of some of the key methods developed since the 1980s to solve the problem more efficiently. Specifically, particle-in-cell (PIC) methods were introduced, followed by the particle-particle/particle-mesh (P^3M) methods that they inspired. The PIC methods scale as $\mathcal{O}(N + M \log M)$ where N is the particle count and M is the number of grid points; with $M \ll N$, this can amount to a significant saving compared to the naïve approach. Subsequently tree codes, which typically run in $\mathcal{O}(N \log N)$ time, were considered, and these methods lead naturally to a detailed review of the original Greengard-Rokhlin $\mathcal{O}(N)$ FMM and a summary of some more recent FMM developments.

In §4 we present a new fast multipole method which was developed utilising the concepts introduced in this section. Our new method is applicable to systems governed by the two-dimensional Yukawa potential, and with suitably chosen parameters its runtime scales as $\mathcal{O}(N)$. We begin that chapter with a definition of the 2D Yukawa potential and a derivation of the required multipole-analogue series, which are in our case obtained from vectorised forms of the Gegenbauer addition formulæ. We then describe our implementation, which naturally handles periodic boundary conditions, and give performance results obtained in numerical experiments.

Chapter 4

An $\mathcal{O}(N)$ multipole method for energies of systems governed by the 2D Yukawa potential

This chapter presents a new fast multipole method for the energy of systems of particles interacting via the 2D Yukawa potential. Simon Cox and Geoff Daniell worked on the original derivation in this chapter. I re-derived the mathematics; the presentation, implementation, illustrations, results and analysis in this chapter are my own work.

4.1 Introduction

In this chapter, we describe a new multipole method which was developed to find the energy of systems of particles in which the inter-particle potential is governed by a modified Bessel function, *i.e.* the 2D Yukawa potential.

As discussed in §3, a volume of work has been done on FMMs for potentials of the form $U(\boldsymbol{x}) = -\log(\|\boldsymbol{x} - \boldsymbol{x}_0\|)$, which typifies the gravitational and Coulomb potentials in two dimensions. Much effort has also been expended on multipole methods applied to scattering problems in fields such as antenna design. However, our interest is in simulating the vortex state in high-temperature superconductors, where the potential U is governed by the solution of the 2D Helmholtz equation [62],

$$\nabla^2 U - \frac{U}{\lambda_{\mathrm{m}}^2} = \delta_2(\boldsymbol{s}), \tag{4.1}$$

where δ_2 is a 2-dimensional delta function at the location of the core of the vortex pancake. When we consider the interaction between stacks of pancake vortices in layered high-temperature superconductors, the relevant solutions are of the form

$$U(\boldsymbol{s}) = K_0\left(\frac{\|\boldsymbol{s}\|}{\lambda_{\mathrm{m}}}\right), \tag{4.2}$$

where K_0 is the modified Bessel function of the second kind [63, §9.6], λ_{m} is the magnetic penetration depth and $\|\boldsymbol{s}\|$ is the distance between the stacks of pancakes [62, 64, 65, 66].

Since λ_{m} can be several orders of magnitude larger than $\|\boldsymbol{s}\|$ [66], the K_0 potential has a very long range character. The inter-particle potential is repulsive, which yields a relatively homogeneous distribution of particles, so we can divide the unit simulation cell into a regular hierarchy of meshes as described in [4, 44, 67, 68]. The particles are grouped together in boxes and are represented by a set of coefficients which allow their effect on particles in distant boxes to be calculated. Our approach follows standard fast multipole methods where the system energy is computed by summing the interactions between boxes which are sufficiently well-separated [57]. Previous studies on multipole methods for the Helmholtz equation have focused on applications in electromagnetic scattering [69, 70, 71]; by contrast, here we develop an algorithm suitable for particle simulations governed by an inter-particle potential resulting from its solution.

In line with the usual terminology in the literature, at a given level of refinement, l, a box b is divided into 4 child boxes at level $l+1$. Box b is referred to as the parent of these boxes. Two boxes are nearest neighbours if their intersection is not empty at a given level. Boxes are 'well-separated' if they are at the same level l, but are not nearest neighbours. The interaction set of a box b is the set of boxes which are the children of

i	*i*	*i*	*i*	*i*	*i*
i	*i*	*i*	*i*	*i*	*i*
i	*i*	*n*	*n*	*n*	*i*
i	*i*	*n*	*b*	*n*	*i*
i	*i*	*n*	*n*	*n*	*i*
i	*i*	*i*	*i*	*i*	*i*

Figure 4.1: The interaction set, i, of a box, b. Nearest neighbours are marked with n.

the nearest neighbours of the parent of b, but are not themselves nearest neighbours of b – see fig. 4.1. The highest level of refinement used is L, which we initially set to $\lceil \log_4 N \rceil$ (see §4.5), where N is the number of particles in the unit cell.

In our proofs we require the following formulæ, which are vectorised forms of the Gegenbauer addition formulæ [72, 73]. The vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are as shown in fig. 4.2. For $\boldsymbol{a} = (a_1, a_2)$, we employ the notation $\Theta(\boldsymbol{a}) = \arg(a_1 + \mathrm{i}a_2)$ (where $\mathrm{i} = \sqrt{-1}$) and throughout we use $K_\nu(a) \equiv K_\nu(|\boldsymbol{a}|)$:

$$K_\nu(|\boldsymbol{a}-\boldsymbol{b}|)\, e^{\mathrm{i}\nu(\Theta[\boldsymbol{a}-\boldsymbol{b}]-\Theta[\boldsymbol{a}])} = \sum_{n=-\infty}^{\infty} K_{\nu+n}(a) I_n(b) e^{\mathrm{i}n(\Theta[\boldsymbol{a}]-\Theta[\boldsymbol{b}])};\ |\boldsymbol{a}| > |\boldsymbol{b}|, \tag{4.3}$$

$$I_\nu(|\boldsymbol{a}-\boldsymbol{b}|)\, e^{\mathrm{i}\nu(\Theta[\boldsymbol{a}-\boldsymbol{b}]-\Theta[\boldsymbol{a}])} = \sum_{n=-\infty}^{\infty} (-1)^n I_{\nu+n}(a) I_n(b) e^{\mathrm{i}n(\Theta[\boldsymbol{a}]-\Theta[\boldsymbol{b}])}, \tag{4.4}$$

$$K_\nu(|\boldsymbol{a}+\boldsymbol{b}|)\, e^{\mathrm{i}\nu(\Theta[\boldsymbol{a}]-\Theta[\boldsymbol{a}+\boldsymbol{b}])} = \sum_{n=-\infty}^{\infty} K_{\nu+n}(a) I_n(b) e^{\mathrm{i}n(\pi-\Theta[\boldsymbol{a}]+\Theta[\boldsymbol{b}])};\ |\boldsymbol{a}| > |\boldsymbol{b}|, \tag{4.5}$$

$$I_\nu(|\boldsymbol{a}+\boldsymbol{b}|)\, e^{\mathrm{i}\nu(\Theta[\boldsymbol{a}]-\Theta[\boldsymbol{a}+\boldsymbol{b}])} = \sum_{n=-\infty}^{\infty} (-1)^n I_{\nu+n}(a) I_n(b) e^{\mathrm{i}n(\pi-\Theta[\boldsymbol{a}]+\Theta[\boldsymbol{b}])}, \tag{4.6}$$

where the modified Bessel functions of order ν, of the first and second kinds respectively, are denoted I_ν and K_ν [63, §9.6].

4.2 Multipole formulæ

In this section, we introduce the formulæ that we require to design and implement the method. These are based upon the Gegenbauer addition formulæ as discussed in §4.1. In §4.2.1 we derive an expression for the energy of an interaction; in §4.2.2 we develop an expression for the energy of the interaction between two well-separated boxes; and

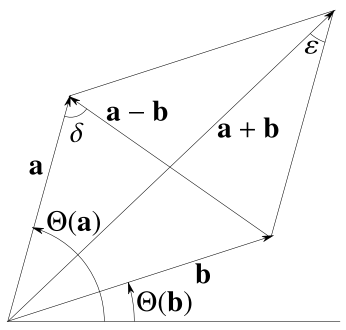


Figure 4.2: Vectorised Bessel functions: definitions of $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{a}+\boldsymbol{b}$, $\boldsymbol{a}-\boldsymbol{b}$, $\delta = \Theta\left[\boldsymbol{a}-\boldsymbol{b}\right] - \Theta\left[\boldsymbol{a}\right]$ and $\varepsilon = \Theta\left[\boldsymbol{a}\right] - \Theta\left[\boldsymbol{a}+\boldsymbol{b}\right]$

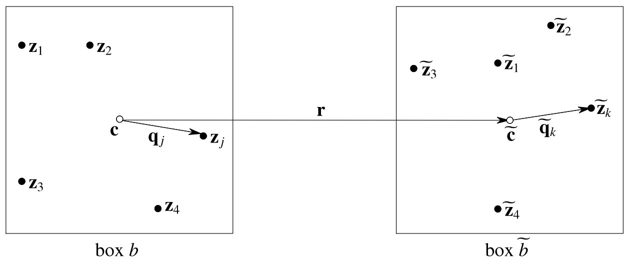


Figure 4.3: Well-separated boxes b and $\widetilde{b}$ at level l containing n_j and n_k particles, located at z_j and $\widetilde{z}_k$ respectively. The vectors $\boldsymbol{q}_j$ and $\widetilde{\boldsymbol{q}}_k$ go from the centres of the boxes, at $\boldsymbol{c}$ and $\widetilde{\boldsymbol{c}}$ respectively, to the locations of the particles.

we furnish a means of translating expansions in §4.2.3.

4.2.1 Interaction energy

In fig. 4.3, we consider two well-separated boxes b and $\widetilde{b}$, at level l, centred at $\boldsymbol{c}$ and $\widetilde{\boldsymbol{c}}$. They are separated by $\boldsymbol{r} = \widetilde{\boldsymbol{c}} - \boldsymbol{c}$ and contain n_j and n_k particles respectively, located at z_j and $\widetilde{z}_k$, where $j = 1, \ldots, n_j$ and $k = 1, \ldots, n_k$.

The interaction energy between two of the particles can be written as:

$$K_0\left(\left|\widetilde{z}_k - z_j\right|\right) = \sum_{m=-\infty}^{\infty} K_m\left(r\right) e^{\mathrm{i}m\Theta[\boldsymbol{r}]} \sum_{n=-\infty}^{\infty} I_{m+n}\left(\widetilde{q}_k\right) e^{\mathrm{i}(m+n)(\pi - \Theta[\widetilde{\boldsymbol{q}}_k])} I_n\left(q_j\right) e^{\mathrm{i}n\Theta[\boldsymbol{q}_j]}, \quad (4.7)$$

where $\boldsymbol{q}_j = z_j - \boldsymbol{c}$ and $\widetilde{\boldsymbol{q}}_k = \widetilde{z}_k - \widetilde{\boldsymbol{c}}$ are vectors from the centre of each box to the particles.

Proof

Write the energy of the two particles as

$$K_0\left(\left|\widetilde{\boldsymbol{z}}_k - \boldsymbol{z}_j\right|\right) = K_0\left(\left|\boldsymbol{r} + \widetilde{\boldsymbol{q}}_k - \boldsymbol{q}_j\right|\right). \tag{4.8}$$

Using (4.5) this becomes:

$$K_0\left(\left|\widetilde{\boldsymbol{z}}_k - \boldsymbol{z}_j\right|\right) = \sum_{m=-\infty}^{\infty} K_m\left(r\right) I_m\left(\left|\widetilde{\boldsymbol{q}}_k - \boldsymbol{q}_j\right|\right) e^{\mathrm{i}m\left(\pi - \Theta[\boldsymbol{r}] + \Theta[\widetilde{\boldsymbol{q}}_k - \boldsymbol{q}_j]\right)}, \tag{4.9}$$

which requires $|\boldsymbol{r}| > \left|\widetilde{\boldsymbol{q}}_k - \boldsymbol{q}_j\right|$ for convergence. This is guaranteed since we only apply the formula between boxes which are well-separated. From (4.4), we write the final term as:

$$I_m\left(\left|\widetilde{\boldsymbol{q}}_k - \boldsymbol{q}_j\right|\right) e^{\mathrm{i}m\left(\Theta[\widetilde{\boldsymbol{q}}_k - \boldsymbol{q}_j] - \Theta[\widetilde{\boldsymbol{q}}_k]\right)} = \sum_{n=-\infty}^{\infty} \left(-1\right)^n I_{m+n}\left(\widetilde{q}_k\right) I_n\left(q_j\right) e^{\mathrm{i}n\left(\Theta[\widetilde{\boldsymbol{q}}_k] - \Theta[\boldsymbol{q}_j]\right)}. \tag{4.10}$$

Using (4.10) in (4.9) and re-arranging yields:

$$\begin{aligned} K_0\left(\left|\widetilde{\boldsymbol{z}}_k - \boldsymbol{z}_j\right|\right) &= \sum_{m=-\infty}^{\infty} K_m\left(r\right) e^{\mathrm{i}m(\pi + \Theta[\widetilde{\boldsymbol{q}}_k] - \Theta[\boldsymbol{r}])} \times \\ &\qquad \sum_{n=-\infty}^{\infty} \left(-1\right)^n I_{m+n}\left(\widetilde{q}_k\right) I_n\left(q_j\right) e^{\mathrm{i}n\left(\Theta[\widetilde{\boldsymbol{q}}_k] - \Theta[\boldsymbol{q}_j]\right)}. \end{aligned} \tag{4.11}$$

Equation (4.7) then follows by replacing m by $-m$ and $-n$ by n and using $\left(-1\right)^\nu = e^{\mathrm{i}\pi\nu}$, $e^{\mathrm{i}\pi\nu} = e^{-\mathrm{i}\pi\nu}$, $I_\nu\left(x\right) = I_{-\nu}\left(x\right)$ and $K_\nu\left(x\right) = K_{-\nu}\left(x\right)$ [63].

Note that if the inter-particle potential is governed by a Bessel function of order higher than 0, then we use a modified version of (4.5):

$$\begin{aligned} K_\nu\left(|\boldsymbol{a} + \boldsymbol{b}|\right) &= \left|K_\nu\left(|\boldsymbol{a} + \boldsymbol{b}|\right) e^{\mathrm{i}\nu(\Theta[\boldsymbol{a}] - \Theta[\boldsymbol{a}+\boldsymbol{b}])}\right| \\ &= \left|\sum_{n=-\infty}^{\infty} K_{\nu+n}\left(a\right) I_n\left(b\right) e^{\mathrm{i}n(\pi - \Theta[\boldsymbol{a}] + \Theta[\boldsymbol{b}])}\right|; \ |\boldsymbol{a}| > |\boldsymbol{b}|, \end{aligned} \tag{4.12}$$

and so

$$\begin{aligned} K_\nu\left(\left|\widetilde{\boldsymbol{z}}_k - \boldsymbol{z}_j\right|\right) &= \left|\sum_{m=-\infty}^{\infty} K_{m+\nu}\left(r\right) e^{\mathrm{i}m(\pi + \Theta[\widetilde{\boldsymbol{q}}_k] - \Theta[\boldsymbol{r}])} \times \right. \\ &\qquad \left. \sum_{n=-\infty}^{\infty} \left(-1\right)^n I_{m+n}\left(\widetilde{q}_k\right) I_n\left(q_j\right) e^{\mathrm{i}n\left(\Theta[\widetilde{\boldsymbol{q}}_k] - \Theta[\boldsymbol{q}_j]\right)}\right|. \end{aligned} \tag{4.13}$$

4.2.2 Multipole expansion

We can compute the total interaction between all of the particles in well-separated boxes b and $\widetilde{b}$ using:

$$\sum_{k=1}^{n_k}\sum_{j=1}^{n_j} K_0\left(\left|\widetilde{\boldsymbol{z}}_k - \boldsymbol{z}_j\right|\right) = \sum_{m=-\infty}^{\infty} K_m\left(r\right) e^{\mathrm{i}m\Theta[\boldsymbol{r}]} \sum_{n=-\infty}^{\infty} \widetilde{Q}_{m+n} Q_n, \tag{4.14}$$

where:

$$\begin{aligned} \widetilde{Q}_{m+n} &= \sum_{k=1}^{n_k} I_{m+n}\left(\widetilde{q}_k\right) e^{\mathrm{i}(m+n)(\pi-\Theta[\widetilde{\boldsymbol{q}}_k])}, \\ Q_n &= \sum_{j=1}^{n_j} I_n\left(q_j\right) e^{\mathrm{i}n\Theta\left[\boldsymbol{q}_j\right]}, \end{aligned} \tag{4.15}$$

are analogous to the multipole expansions for each of the boxes.

Proof

This follows from expanding the energy given by (4.7) and changing the order of summation on the right hand side of (4.14).

4.2.3 Translation of expansions

In fig. 4.4, we show the parents of boxes b and $\widetilde{b}$, at level $l-1$, centred at $\boldsymbol{c}^{(l-1)}$ and $\widetilde{\boldsymbol{c}}^{(l-1)}$, and separated by $\boldsymbol{r}^{(l-1)} = \widetilde{\boldsymbol{c}}^{(l-1)} - \boldsymbol{c}^{(l-1)}$. We can compute the contribution from the particles in boxes b and $\widetilde{b}$ to the interaction energy between the parent boxes using the multipole expansions for boxes b and $\widetilde{b}$:

$$\begin{aligned} E_{b\widetilde{b}} = & \sum_{m=-\infty}^{\infty} K_m\left(r^{(l-1)}\right) e^{\mathrm{i}m\Theta\left[\boldsymbol{r}^{(l-1)}\right]} \sum_{n=-\infty}^{\infty}\sum_{\mu=-\infty}^{\infty} \widetilde{Q}^{(l)}_{m+n+\mu} I_\mu\left(\widetilde{a}\right) e^{\mathrm{i}\mu\Theta[\widetilde{\boldsymbol{a}}]} \times \\ & \sum_{\lambda=-\infty}^{\infty} Q^{(l)}_{n+\lambda} I_\lambda\left(a\right) e^{\mathrm{i}\lambda(\pi-\Theta[\boldsymbol{a}])}, \end{aligned} \tag{4.16}$$

where $\boldsymbol{a} = \boldsymbol{c}^{(l-1)} - \boldsymbol{c}^{(l)}$ and $\widetilde{\boldsymbol{a}} = \widetilde{\boldsymbol{c}}^{(l-1)} - \widetilde{\boldsymbol{c}}^{(l)}$ are vectors from the centre of the child box to the centre of the parent box. To find the total energy between all particles in the parent boxes, we sum the contributions from all of the children of the parents of b and $\widetilde{b}$.

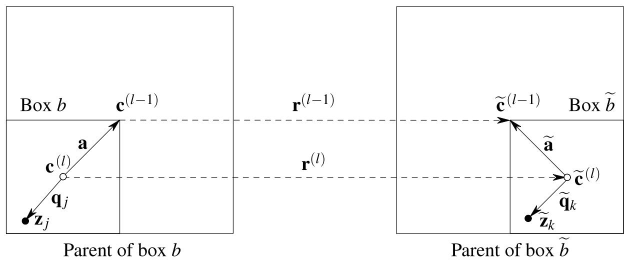


Figure 4.4: Translation of multipole expansions from a child box at level l to its parent at level $l-1$. The vectors $\boldsymbol{q}_j$ and $\widetilde{\boldsymbol{q}}_k$ go from the centres of the boxes at level l to the locations of particles at z_j and $\widetilde{z}_k$, whilst $\boldsymbol{a}$ and $\widetilde{\boldsymbol{a}}$ go from the centres of the boxes at level l to the centres $\boldsymbol{c}^{(l-1)}$ and $\widetilde{\boldsymbol{c}}^{(l-1)}$ of their respective parents at level $l-1$.

Proof

Using (4.14), write the interaction between particles in the parent box as:

$$\sum_{k\in\tilde{b}}\sum_{j\in b} K_0\left(\left|\widetilde{z}_k - \boldsymbol{z}_j\right|\right) = \sum_{m=-\infty}^{\infty} K_m\left(r^{(l-1)}\right) e^{\mathrm{i}m\Theta\left[\boldsymbol{r}^{(l-1)}\right]} \sum_{n=-\infty}^{\infty} \widetilde{Q}_{m+n}^{(l-1)} Q_n^{(l-1)}, \tag{4.17}$$

where the terms:

$$Q_n^{(l-1)} = \sum_{j=1}^{n_j} I_n\left(q_j^{(l-1)}\right) e^{\mathrm{i}n\Theta\left[\boldsymbol{q}_j^{(l-1)}\right]}, \quad \text{and} \tag{4.18}$$

$$\widetilde{Q}_{m+n}^{(l-1)} = \sum_{k=1}^{n_k} I_{m+n}\left(\widetilde{q}_k^{(l-1)}\right) e^{\mathrm{i}(m+n)\left(\pi - \Theta\left[\widetilde{\boldsymbol{q}}_k^{(l-1)}\right]\right)}, \tag{4.19}$$

use the distance from each particle to the centre of the parent boxes. Since

$$\begin{aligned} \boldsymbol{q}_j^{(l-1)} &= \boldsymbol{z}_j - \boldsymbol{c}^{(l-1)} \\ &= \boldsymbol{z}_j - \boldsymbol{c}^{(l)} + \boldsymbol{c}^{(l)} - \boldsymbol{c}^{(l-1)} \\ &= \boldsymbol{q}_j^{(l)} - \boldsymbol{a}, \end{aligned} \tag{4.20}$$

we can use (4.4) to write:

$$\begin{aligned} e^{-\mathrm{i}n\Theta\left[\boldsymbol{q}_j^{(l)}\right]} I_n\left(q_j^{(l-1)}\right) e^{\mathrm{i}n\Theta\left[\boldsymbol{q}_j^{(l-1)}\right]} &= I_n\left(\left|\boldsymbol{q}_j^{(l)} - \boldsymbol{a}\right|\right) e^{\mathrm{i}n\left(\Theta\left[\boldsymbol{q}_j^{(l)} - \boldsymbol{a}\right] - \Theta\left[\boldsymbol{q}_j^{(l)}\right]\right)} \\ &= \sum_{\lambda=-\infty}^{\infty} (-1)^{\lambda} I_{n+\lambda}\left(q_j^{(l)}\right) I_{\lambda}(a) e^{\mathrm{i}\lambda\left(\Theta\left[\boldsymbol{q}_j^{(l)}\right] - \Theta\left[\boldsymbol{a}\right]\right)}. \end{aligned} \tag{4.21}$$

Substituting into (4.18) yields

$$\begin{aligned}
Q_n^{(l-1)} &= \sum_{j=1}^{n_j} e^{\mathrm{i}n\Theta\left[\boldsymbol{q}_j^{(l)}\right]} \sum_{\lambda=-\infty}^{\infty} (-1)^{\lambda} I_{n+\lambda}\left(q_j^{(l)}\right) I_{\lambda}(a)\, e^{\mathrm{i}\lambda\left(\Theta\left[\boldsymbol{q}_j^{(l)}\right]-\Theta[\boldsymbol{a}]\right)} \\
&= \sum_{\lambda=-\infty}^{\infty} \sum_{j=1}^{n_j} I_{n+\lambda}\left(q_j^{(l)}\right) e^{\mathrm{i}(n+\lambda)\left(\Theta\left[\boldsymbol{q}_j^{(l)}\right]\right)} I_{\lambda}(a)\, e^{\mathrm{i}\lambda(\pi-\Theta[\boldsymbol{a}])} \\
&= \sum_{\lambda=-\infty}^{\infty} Q_{n+\lambda}^{(l)} I_{\lambda}(a)\, e^{\mathrm{i}\lambda(\pi-\Theta[\boldsymbol{a}])}. \qquad (4.22)
\end{aligned}$$

Combining (4.22) and a similar proof for $\widetilde{Q}_{m+n}^{(l-1)}$ with (4.17) yields (4.16).

4.3 Algorithm

1. For each box at the most-refined level, L, and for $n = 0, \ldots, k_{\text{trunc}}$, compute

$$\begin{aligned}
Q_n^{(L)} &= \sum_{j=1}^{n_b} I_n(q_j)\, e^{\mathrm{i}n\Theta[\boldsymbol{q}_j]}, \quad \text{and} \\
\widetilde{Q}_n^{(L)} &= \sum_{j=1}^{n_b} I_n(q_j)\, e^{\mathrm{i}n(\pi-\Theta[\boldsymbol{q}_j])} = (-1)^n \left(Q_n^{(L)}\right)^*, \qquad (4.23)
\end{aligned}$$

   where $\boldsymbol{q}_j = \boldsymbol{z}_j - \boldsymbol{c}$ is a vector from the centre of the box to the particle j and $(Q_n)^*$ is the complex conjugate of Q_n. The number of particles in the box under consideration is n_b.

2. Shift the multipole expansion for each box up to its parent at the next level up using:

$$Q_n^{(l-1)} = \sum_{\text{children}} \sum_{\lambda=-\infty}^{\infty} Q_{n+\lambda}^{(l)} I_{\lambda}(a)\, e^{\mathrm{i}\lambda(\pi-\Theta[\boldsymbol{a}])}, \qquad (4.24)$$

   where $\boldsymbol{a} = \boldsymbol{c}^{(l-1)} - \boldsymbol{c}^{(l)}$ is a vector from the centre of the child box currently under consideration at level l to the centre of its parent at level $l-1$, and the sum runs over all such children of each box at level $(l-1)$. Repeat this shift for all levels until the coefficient for the entire unit cell, $Q_n^{(1)}$, is calculated. For each value of $Q_n^{(l)}$ calculated, apply the last relation in (4.23) to find the corresponding $\widetilde{Q}_n^{(l)}$.

3. Compute the interaction energies. In contrast to other multipole algorithms, we do not shift the expansions down the tree; instead we directly compute the interaction energies between pairs of boxes at each level and sum them. The work required by the two approaches is comparable, though our method is slightly more amenable to efficient parallelisation of the algorithm. For every box $\widetilde{b}$ in the interaction set of a box b at level l, calculate the contribution to the total energy $E^{\left(b,\widetilde{b},l\right)}$:

$$\begin{aligned} E^{\left(b,\widetilde{b},l\right)} &= \sum_{k=1}^{n_k}\sum_{j=1}^{n_j} K_0\left(\left|\widetilde{\boldsymbol{z}}_k - \boldsymbol{z}_j\right|\right) \\ &= \sum_{m=-\infty}^{\infty} K_m\left(r^{(l)}\right) e^{\mathrm{i}m\Theta\left[\boldsymbol{r}^{(l)}\right]} \sum_{n=-\infty}^{\infty} \widetilde{Q}^{(l)}_{m+n} Q^{(l)}_n , \end{aligned} \tag{4.25}$$

where n_j and n_k are the numbers of particles in boxes b and $\widetilde{b}$ respectively. Calculate $E^{\,\mathrm{fmm}}$, the contribution to the total energy due to multipole expansions across all the levels l using

$$E^{\,\mathrm{fmm}} = \sum_l \sum_{b_l} \sum_{\substack{\widetilde{b}\in\Omega_b \\ \widetilde{b}>b}} E^{\left(b,\widetilde{b},l\right)}, \tag{4.26}$$

where Ω_b denotes the interaction set of box b and b_l denotes 'boxes at level l'. Note that we need only compute the interaction energy between each pair once.

4. Compute nearest neighbour interaction energies directly. For each particle, j, in each box at level L, compute the interaction energy with particles, k, in the same box b and in each of the nearest neighbour boxes nn:

$$E^{\,\mathrm{direct}} = \sum_{j_L} \sum_{\substack{k\,\in\, nn\cup b \\ k>j}} K_0\left(\left|\boldsymbol{z}_k - \boldsymbol{z}_j\right|\right), \tag{4.27}$$

where j_L represents box j at level L and it is again possible to ensure that each interaction energy is computed only once.

5. Compute the total energy. Finally, add together the interaction energy from the multipole expansions and the directly computed energies:

$$E^{\,\mathrm{tot}} = E^{\,\mathrm{fmm}} + E^{\,\mathrm{direct}}. \tag{4.28}$$

Due to the rapid convergence of the Gegenbauer addition formulæ, the infinite summations in the above steps may be truncated at k_{trunc} between 5 and 20 terms while maintaining good overall accuracy, as illustrated in fig. 4.8. A further factor of two in performance can be obtained by using symmetry to convert all summations from $k = -\infty \ldots \infty$ to the range $k = 0 \ldots \infty$. We now analyse the algorithmic complexity.

Step	Operations	Description
1	$\mathcal{O}\left(2Nk_{\text{trunc}}\right)$	Compute $2k_{\text{trunc}}$ terms for each of N particles and sum.
2	$\mathcal{O}\left(N(k_{\text{trunc}}^2 + 3k_{\text{trunc}})\right)$	There are $\sim N$ boxes to be shifted up the quadtree and each shift requires $k_{\text{trunc}}^2 + 3k_{\text{trunc}}$ operations.
3	$\sim \mathcal{O}\left(\frac{27}{2}Nk_{\text{trunc}}^2\right)$	Each box has at most 27 entries in its interaction set, but each interaction is computed only once for each of the N boxes, which halves the number of operations required. The factor k_{trunc}^2 is from the $\mathcal{O}(k_{\text{trunc}}^2)$ correlation required.
4	$\sim \mathcal{O}\left(4N\right)$	Each box has 8 nearest neighbours, and with sufficient subdivision there should be ~ 1 particle per box. Computing each pairwise interaction only once yields an average $\mathcal{O}(4N)$ computations.
5	1	Adding together the two components of the energy.

The total algorithm therefore scales with $\mathcal{O}(N)$ for constant k_{trunc} and requires $\mathcal{O}(k_{\text{trunc}}N)$ memory. The performance of the method has been optimised by employing recurrence relations [74] for the trigonometric terms and a vectorised Bessel function. Furthermore, our sequences converge rapidly so we require values of k_{trunc} between 5 and 20: the additional memory required to obtain numerically stable $\mathcal{O}(k_{\text{trunc}} \log k_{\text{trunc}})$ correlation does not improve the performance significantly. Since the force between the particles is repulsive, the particle distribution in simulations tends to be homogeneous, so it is not necessary to adopt an adaptive hierarchical grid in this case.

4.4 Periodic boundary conditions

It has been shown [47] that it is essential to sum the interactions of the particles over infinite periodic repeats of the unit cell to handle the long range nature of the interactions. This avoids introducing artificial effects into the simulation that may otherwise be given rise to by truncation of the interaction range. We write the energy of the infinitely tiled system as:

$$U(\boldsymbol{s}) = K_0^*(\boldsymbol{s}) = \sum_{m_x,m_y} K_0\left(\left|\boldsymbol{s} + L_x m_x \hat{\boldsymbol{x}} + L_y m_y \hat{\boldsymbol{y}}\right|\right), \tag{4.29}$$

where m_x and m_y are integers, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are unit vectors in the x and y directions, and L_x and L_y are the lengths of the edges of the simulation cell. The energy of the system is bounded as more unit cells are included in the calculation, since $\lim_{s\to\infty} K_\nu(s) = 0$. We can compute the interaction between well-separated unit cells using

$$E^{\text{ infinite}} = \sum_{m_x,m_y} \sum_{m=-\infty}^{\infty} K_m\left(\left|m_x L_x \hat{\boldsymbol{x}} + m_y L_y \hat{\boldsymbol{y}}\right|\right) e^{im\Theta\left[m_x L_x \hat{\boldsymbol{x}} + m_y L_y \hat{\boldsymbol{y}}\right]} \sum_{n=-\infty}^{\infty} \widetilde{Q}_{m+n}^{(0)} Q_n^{(0)}, \tag{4.30}$$

where $Q_n^{(0)}$ and $\widetilde{Q}_{m+n}^{(0)}$ are multipole expansions of the unit cell at the highest level. Following the procedure in [47] we reverse the order of computation and pre-compute the coefficients:

$$S_m = \sum_{m_x,m_y} K_m\left(\left|m_x L_x \hat{\boldsymbol{x}} + m_y L_y \hat{\boldsymbol{y}}\right|\right) e^{im\Theta\left[m_x L_x \hat{\boldsymbol{x}} + m_y L_y \hat{\boldsymbol{y}}\right]}. \tag{4.31}$$

In equations (4.30) and (4.31) the values of m_x and m_y are chosen so as to add cells to the summation in rings of increasing radius, until the sum converges.

Finally it is necessary to compute the contribution to the energy from image particles in the nearest neighbours of the unit cell. This is achieved by computing:

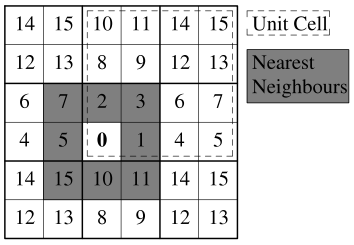


Figure 4.5: Interaction set of box 0 at level 3 of the mesh, using periodic boundary conditions.

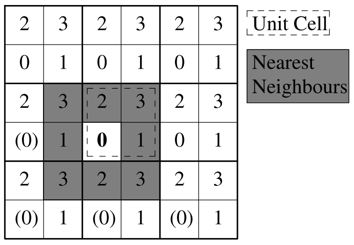


Figure 4.6: Interaction set of box 0 at level 2 of the mesh, using periodic boundary conditions. To avoid double counting of pairwise interactions, the images of box 0 shown in brackets are ignored.

1. the energy from the interaction set of each box in levels $3 \dots L$ using periodic boundary conditions, as shown in fig. 4.5, and

2. the interaction energy due to periodic repeats at level 2, as shown in fig. 4.6.

For the latter energy, it is necessary to explicitly exclude some of the self-interactions between boxes to avoid double counting: these have their box numbers in brackets in fig. 4.6. The total system energy is therefore given by $E^{\text{tot}} = E^{\text{infinite}} + E^{\text{fmm}} + E^{\text{direct}}$, where E^{fmm} and E^{direct} now incorporate the energy contribution from periodic image particles in the nearest neighbours of the unit cell.

4.5 Results

We have implemented the method described here for systems whose inter-particle potential is governed by the zeroth order modified Bessel function K_0, and demonstrated

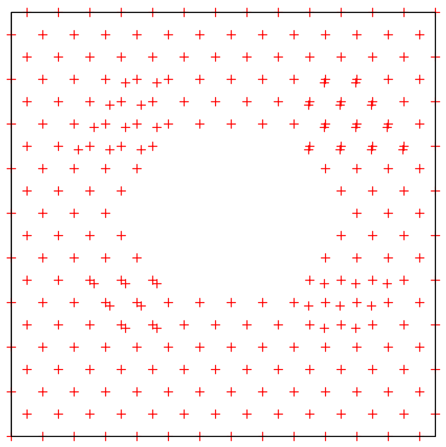


Figure 4.7: One of the inhomogeneous particle distributions used to test the effect of particle location on the accuracy of the FMM. Particles were displaced away from the central region.

its performance using a set of particles in a unit cell. Table 4.1 shows the accuracy of the multipole method for three values of k_{trunc}, the number of retained terms in our series expansions. The data were calculated by simulating a single unit cell without periodic boundary conditions containing N particles in a hexagonal configuration, representative of the relatively homogeneous and regular particle distributions obtained in realistic simulations. Similar results are obtained when the particles are randomly distributed. The inter-particle distance is fixed to be 1 unit for all N and all calculations use a tree with 4 levels. We show the fractional error per particle compared to the direct evaluation when the summations are truncated at $k_{\text{trunc}} = 5$, 10 and 20, and find it to be, on average, of orders 10^{-4}, 10^{-6}, and 10^{-7} respectively. The errors do not exhibit a monotonic trend with increasing N. We speculate that this may be due to varying numbers of particles falling into each cell at the finest grid level as N is changed, which would affect the numbers of particles involved in the direct vs. the multipole calculation.

We tested the accuracy of the multipole method applied to inhomogeneous particle distributions, such as that illustrated in fig. 4.7, and found that for $k_{\text{trunc}} = 5$, 10 and 20 the error is of orders 10^{-4}, 10^{-5}, and 10^{-7} respectively. In these cases, the accuracy

	5 coefficients		10 coefficients		20 coefficients	
N	$\lvert\delta_{\text{fmm}}\rvert \times 10^{-4}$	t_{run}	$\lvert\delta_{\text{fmm}}\rvert \times 10^{-6}$	t_{run}	$\lvert\delta_{\text{fmm}}\rvert \times 10^{-7}$	t_{run}
280	2.1	0.04 s	0.8	0.09 s	1.6	0.20 s
468	4.9	0.05 s	0.9	0.10 s	0.3	0.21 s
900	2.6	0.06 s	6.5	0.11 s	3.8	0.21 s
988	3.4	0.06 s	11.0	0.11 s	1.3	0.21 s
1080	3.8	0.06 s	15.0	0.11 s	2.9	0.21 s
1176	4.1	0.06 s	0.8	0.11 s	0.8	0.21 s
1320	2.4	0.07 s	8.7	0.11 s	2.5	0.22 s
1426	1.3	0.07 s	13.0	0.11 s	0.7	0.21 s
1700	1.9	0.07 s	8.9	0.12 s	4.5	0.22 s
2128	2.5	0.08 s	9.1	0.12 s	1.0	0.23 s

Table 4.1: Accuracy of fast multipole method for a unit cell containing a regular arrangement of N particles, showing the fractional error $|\delta(\text{fmm})|$ in energy per particle as the number of coefficients is increased, at 4 levels of refinement. Errors are calculated relative to the naïve method. The runtime for each FMM result is also shown.

compares favourably to that for homogeneous distributions. We note that in the application of the method to simulations of stacks of pancake vortices in high-temperature superconductors, such a configuration is unlikely to arise because the inter-particle potential is repulsive, which usually gives rise to relatively homogeneous particle distributions. To maintain high accuracies for increasingly inhomogeneous systems, it may be necessary to adopt an adaptive meshing scheme such as that given in [54].

We performed numerical experiments to determine the effect of truncating the infinite summations. In the correlation routine, at the innermost loop level we found the largest- and smallest-magnitude contributions to the result, $\widetilde{L}$ and $\widetilde{S}$ respectively. At the outer loop level, we take the ratio $\widetilde{R} = \widetilde{S}/\widetilde{L}$, and for any value of k_{trunc} we define $\varepsilon_{\text{series}} = \max\left(\widetilde{R}\right)$, where the maximum is taken across all calls to the correlation routine in the simulation. In figure 4.8 we plot this ratio as a function of k_{trunc}. It can be seen that the ratio becomes comparable to machine precision ε_{m} at $k_{\text{trunc}} = 15$ for single precision, and $k_{\text{trunc}} = 25$ for double precision. The values of this ratio depend upon λ, and figure 4.9 shows the results of a numerical study of the effect of this parameter on the ratio $\varepsilon_{\text{series}}$. We are interested in cases where $\lambda \gg |\boldsymbol{s}|$ [66] and in these large-λ cases it can be seen that $\varepsilon_{\text{series}}$ is very small even for the relatively low value of $k_{\text{trunc}} = 10$.

In fig. 4.10 we show the effect of using different numbers of tree levels in the simulation, for various numbers of particles N and for two different numbers of retained terms in our multipole-analogue series, $k_{\text{trunc}} = 5$ and $k_{\text{trunc}} = 20$. The $\mathcal{O}(N)$ scaling of the algorithm is evident and our implementation of the method is faster than the $\mathcal{O}(N^2)$ algorithm for $N \gtrsim 1100$. For $k_{\text{trunc}} = 20$, the crossover is around $N \sim 2500$, provided the level of refinement is chosen appropriately.

The performance of our method for the infinitely tiled periodic system is shown in Figure 4.11. The accuracy of these results is checked by ensuring that the energy per particle for the infinitely tiled system is constant as N is increased. Typical per particle errors were of similar magnitude to those in Table 4.1. The performance improvement of using the infinite summation technique over a naïve implementation, which directly

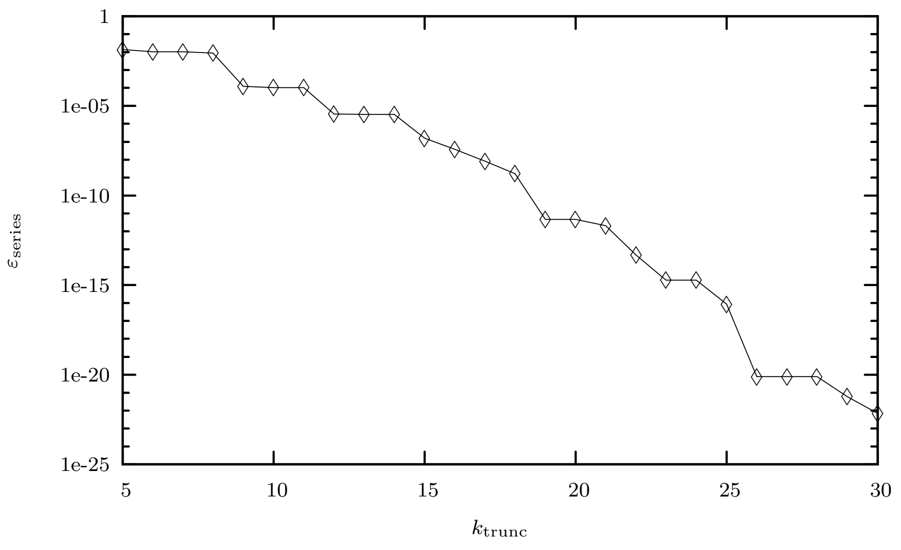


Figure 4.8: The value of $\varepsilon_{\text{series}}$ as a function of k_{trunc}, illustrating the justification for truncating the infinite summations with relatively few terms. For single precision, $\varepsilon_{\text{series}}$ is comparable to the machine precision ε_{m} for $k_{\text{trunc}} = 15$; for double precision this occurs for $k_{\text{trunc}} = 25$. We held $\lambda = 1$ in this plot.

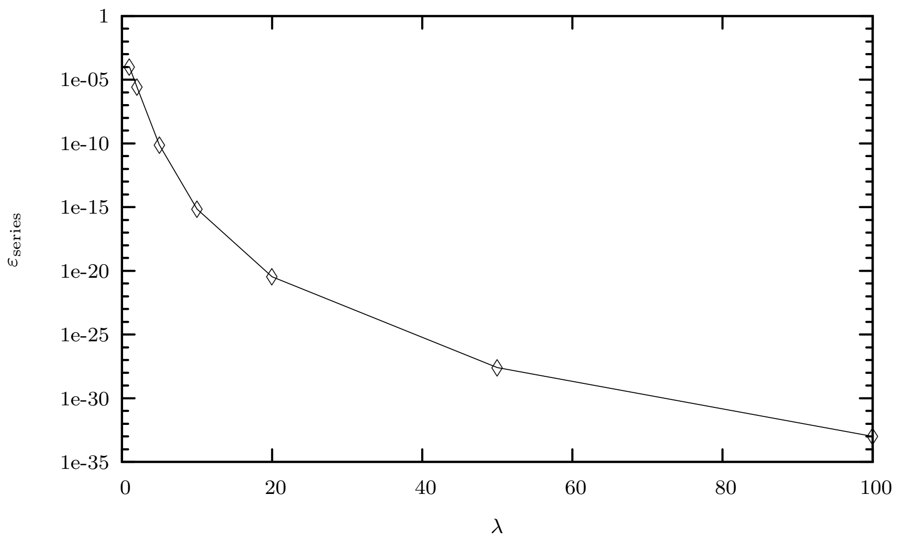
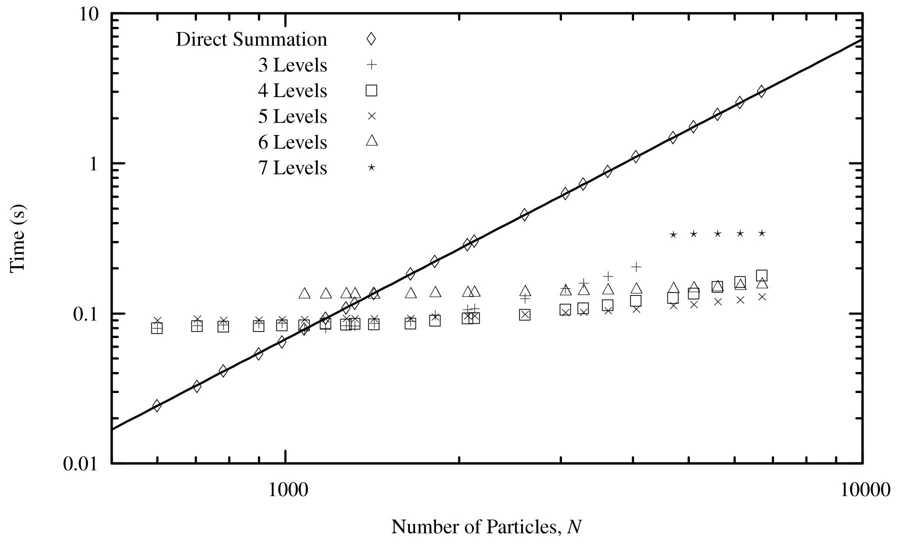
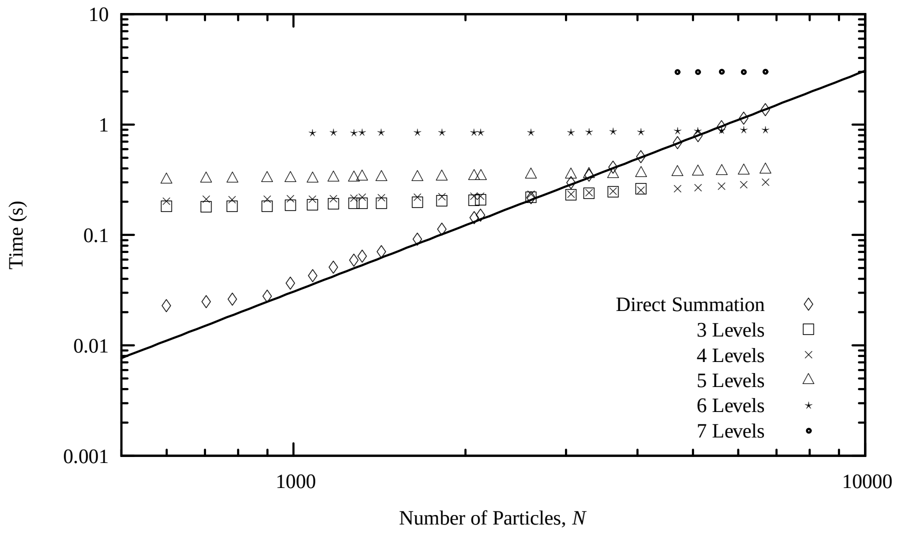


Figure 4.9: The variation of the error due to series cut-off $\varepsilon_{\text{series}}$ as a function of λ, the magnetic penetration depth. We truncated the series at $k_{\text{trunc}} = 10$ terms in this plot.



(a) Runtime with $k_{\text{trunc}} = 5$



(b) Runtime with $k_{\text{trunc}} = 20$

Figure 4.10: Time to evaluate system energy for particles in a hexagonal lattice in a unit cell without periodic boundary conditions using direct and multipole methods, with different numbers of levels in the multipole algorithm (a tree with L levels has 4^L boxes at its lowest level).

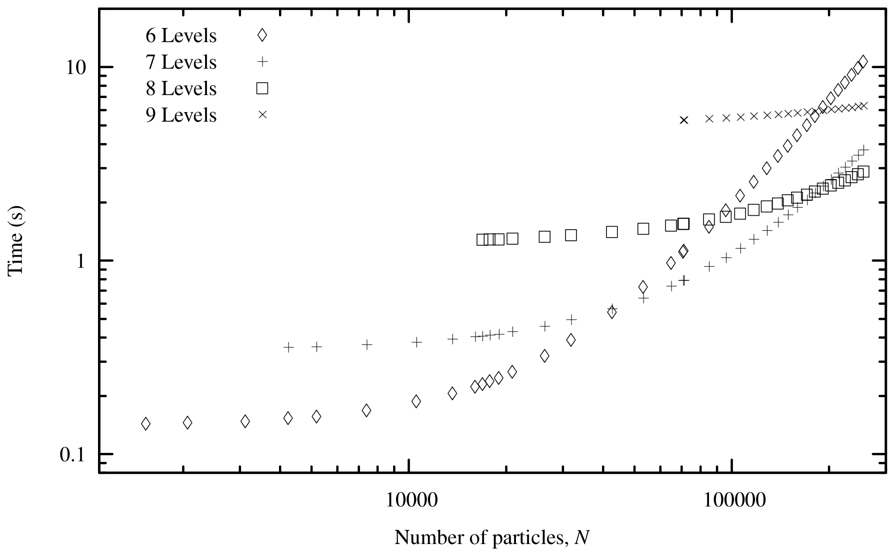


Figure 4.11: Time taken to evaluate system energy for an infinitely tiled periodic hexagonal lattice using the multipole method with cut-off of infinite sums after $k_{\text{trunc}} = 5$ terms, and different numbers of levels of refinement.

sums the energies from particles in unit cells in shells of increasing radius has been detailed elsewhere [47]. Again, the $\mathcal{O}(N)$ scaling of the algorithm is evident when there are relatively few particles in each box at the lowest level. As the number of levels is decreased the direct summation in equation (4.27) becomes dominant, which ultimately scales as $\mathcal{O}(N^2)$, as is evident from the lines showing the performance when 6 and 7 levels are used with large N. For practical applications of the method in simulations, it is best to optimise the number of levels to employ in the tree hierarchy by timing the implementation on the machine to be used. We recommend such timings begin with $\log_4 N$ levels, so that, on average, there is one particle per box at the most refined level.

4.6 Conclusions

We have described and implemented an $\mathcal{O}(N)$ algorithm suitable for molecular dynamics or Monte Carlo simulation of systems where the inter-particle potential is governed by a modified Bessel function, K_ν. The method naturally handles an infinitely tiled periodic system.

Our implementation of the method shows speed advantages for $N \gtrsim 1100$ with $k_{\text{trunc}} = 5$ or $N \gtrsim 2500$ for $k_{\text{trunc}} = 10$ when applied to the unit cell, and for $N \gtrsim 300$ in infinitely-tiled simulations with $k_{\text{trunc}} = 5$.

Whilst the algorithm is efficient and the implementation makes use of several optimisations, there may be potential to further speed up the runtime by utilising one of the acceleration technologies mentioned in §2. It is expected that molecular dynamics or Monte Carlo simulations calling this algorithm could save significant amounts of computational time, either by running multiple threads, each of which would run the algorithm described for a different vortex stack configuration, or by parallelising the implementation of the algorithm itself and calling it with different vortex stack configurations sequentially. The calculation of expansions for the lowest-level boxes may be carried out independently, and the translation of expansions up the tree could make use of any optimised hardware facilities for reduction. Which parallelisation scheme would deliver better performance overall would depend upon the specifics of the architecture in use, and its available memory and bandwidth.

Chapter 5

Meshless methods

5.1 Introduction

There are many engineering problems which give rise to a set of partial differential equations (PDEs) along with some boundary conditions when modelled mathematically. These problems were traditionally solved using various mesh-based methods, including the finite element method (FEM), finite difference method (FDM) and finite volume method (FVM). We begin this chapter by briefly introducing these methods. Whilst the methods are popular and well-established, they suffer some disadvantages inherent in their mesh-based nature, and for this reason there has been interest in alternatives, including the meshless methods [75, 76]. Here, we introduce meshless methods, and also the radial basis functions that many meshless methods rely upon. We go on to describe a novel meshless method developed to solve an eigenvalue problem on a periodic domain; this method is based on the radial basis function finite difference (RBF-FD) formulation. In §6, we draw on the material presented here in developing a hybrid meshless weak-strong form method suitable for modelling photonic crystals, and also develop RBF-FD methods for the same purpose, based upon the RBF-FD method formulated in this chapter.

5.2 Mesh based methods

There are three main classes of established discretization techniques that are routinely applied to problems of solving PDEs in science and engineering. These are the finite element method (FEM), finite difference method (FDM) and the finite volume method (FVM). In the following subsections, we give a very brief introduction to each of the methods, and provide references for more detailed information. These mesh-based methods provide the context against which the meshless methods – the focus of this chapter – were first developed, and therefore a brief introduction to each of the methods is in order. We also note where these methods have been applied to photonic crystal modelling, in advance of the development of new meshless methods for this purpose in §6.

5.2.1 Finite element method

In the FEM, an approximate solution to a partial differential equation (PDE) is sought by means of a variational problem, involving integrating the differential equation over the problem domain. The finite elements after which the method is named are the non-overlapping subdomains into which the problem must be divided; the solution of the PDE is then approximated by a polynomial function on each element. The individual element equations are then systematically recombined into a global system of equations, which may be solved using various techniques subject to the initial conditions of the original problem to obtain a numerical answer.

The seminal paper on the FEM was authored by Turner *et al.* [77], in which examples are given of the use of simple finite elements such as the triangular plate and pin-jointed bar for the analysis of aircraft structures. The method has since been successfully applied to many problems, including heat conduction [78], electric and magnetic fields [79], seepage flow [80] and fluid dynamics [81], as well as the structural mechanics for which it was originally formulated. There now exists a large body of literature on the subject, including reference books such as [81, 82, 83], and the method has also been applied to photonic crystal modelling [84, 85, 86, 87].

Despite the wide success of the method, it has some drawbacks [75], which generally stem from the mesh, that is, the topological map connecting the individual elements together. These issues include the possibility of mesh distortion in Lagrangian type computations, which may have severely damaging consequences for the accuracy of the method [88]. Moreover, where problems have a distinct local character or high gradients, a very fine mesh is required. Such a mesh can be computationally expensive to generate and update. Remeshing adaptively is a formidable task, especially in cases of explosion/fragmentation, impact/penetration and flow passing obstacles. It is also difficult to map thermodynamic state variables from one mesh to another without introducing numerical errors, which means that remeshing should be avoided when possible.

Usually, FEM interpolation fields are functions with C^0 smoothness, and it is difficult to construct higher order fields for arbitrary geometries using unstructured meshes in multiple dimensions; this can also result in poorer accuracy [89].

In mechanics applications there is an additional limitation, in that when FEMs are used to simulate material disintegration, the disintegration corresponds to the disintegration of the FEM subdivision. Therefore, the possible disintegration patterns are limited to those embedded in the way that the domain is subdivided before the simulation begins.

5.2.2 Finite difference method

In an FDM, the unknown function $u(\boldsymbol{x})$ is represented by its values at some discrete set of points, which lie on the nodes of a mesh. Then, a finite-difference representation is substituted into the equation to be solved, expressing the derivatives in terms of the known function values at the nodes. This allows the formulation of the problem as a matrix equation $\mathbf{A} \cdot \boldsymbol{u} = \boldsymbol{b}$, where the matrix $\mathbf{A}$ is tridiagonal with fringes and $\boldsymbol{b}$ encodes information on the (known) values of $u(\boldsymbol{x})$ or its derivative on the boundary points.

The method still relies upon the creation of a grid of nodes, and requires the topological relations between them be known, so that the finite difference formulæ for the derivatives can be evaluated. In this sense, the FDM potentially suffers from the disadvantages of computationally-expensive mesh generation that applies to the FEM. Moreover, care has to be taken to ensure the sparseness of the matrix **A**, whose structure is governed by the problem at hand, in order to avoid generating prohibitively large matrix problems [74].

The finite difference method has been applied to the problem of photonic crystal modelling by a group from MIT, and is implemented in the well-known finite difference time domain software Meep [90].

5.2.3 Finite volume method

The finite volume method offers a discretization that calculates values at discrete points on a mesh which has been set up on the geometry of the problem at hand. Therefore, the drawbacks arising from the mesh (and possible necessary remeshing) in the FEM apply also to the FVM. The solution proceeds by considering the small volumes surrounding each node, and using the divergence theorem to convert volume integrals containing a divergence term into surface integrals. The method is naturally conservative, since the terms are evaluated as fluxes at each surface of the volume, and the flux entering a given volume is identical to that leaving the adjacent volume. These methods are particularly attractive, therefore, for simulating e.g. elliptic, parabolic, or hyperbolic conservation laws. Considerable detail on these methods is given by Eymard, *et al.* [91], and the FVM has been also applied to modelling photonic crystal devices [92, 93].

5.2.4 Other mesh-based methods

Additional mesh-based methods have been introduced. One example is the discontinuous Galerkin methods, where features from both finite element and finite volume methods are combined. These methods have been gradually developed by a number of workers, and a self-contained treatment of the method, its applications to various problems, and a rigorous mathematical analysis of the underpinning ideas is presented in a recently-published textbook [94]. Another example is given by the immersed boundary method [95], which was introduced to study problems in fluid-structure interaction, having been developed for the study of flow patterns around heart valves [96].

5.3 Meshless methods

The difficulties arising from mesh generation and remeshing in the methods outlined above have motivated research into alternative methods for solving PDEs in engineering and scientific problems. One group of methods that is enjoying some popularity are the so-called meshless methods, which are detailed in the following sections. The methods have been developed for particular applications within the discipline of engineering, and we briefly mention a few representative examples of such applications. We also give details of some of the common radial basis functions (RBFs) that are used with many meshless methods, and consider how meshless methods may be classified by their formulation.

5.3.1 Introduction

Meshless methods are a class of methods for solving PDEs in engineering problems. In a meshless method, the problem is discretized at a number of nodes, positioned arbitrarily in the problem domain, where there is no need for any special connections or relationships between the nodes. Meshless methods naturally facilitate solving problems involving discontinuities, large and complicated geometries, and large deformations, whilst avoiding the inherent problems of generating and updating meshes including the topological information required for traditional, mesh-based methods.

Meshless methods have been defined as follows [97]:

> [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 domain discretization.

Meshless methods enjoy a history traceable back to early collocation methods published in the 1930s for the purpose of computing excited electronic energy bands in metals [98, 99]. Many types of meshless method have since been introduced, such as the diffuse element method (DEM) [100], the element free Galerkin (EFG) method [101], meshfree collocation methods – on which there exists a body of literature including Kansa's 1990 paper introducing the RBF (radial basis function) collocation method of solving elliptic, hyperbolic and parabolic PDEs [102] and some further developments [103, 104], and meshfree weak-strong form methods, which were developed chiefly by Liu and Gu from 2002 onwards [104, 105]. As recently as 2013, new forms of meshless method have been introduced, such as the direct meshless local Petrov-Galerkin (DMLPG) method from Mirzaei and Schaback [106], which uses a generalised moving least squares approximation and promises lower computational costs and higher accuracies. These methods are broadly divisible into three categories, based upon whether they use a weak or strong formulation, or some hybrid approach, as we shall explore in §5.3.4.

Despite the relative novelty of applying meshless methods to PDEs in engineering, they have already been shown to be suitable for a variety of application areas, including simulations of crack growth and propagation (one of many examples is given in [107]), strain localisation (e.g. [108]), a number of fluid dynamics/fluid-structure interaction problems [109, 110], and heat flows [111]. More recent work has demonstrated that meshless methods can be applied to solve eigenvalue problems on periodic domains [112], and also showed that they can be applied to bandgap calculations for photonic crystals [113].

The meshless local Petrov-Galerkin (MLPG) method was applied in 2012 [114] to boundary-value problems arising in the analysis of two-dimensional electromagnetic wave propagation and scattering. The results for the TM polarisation were found to be in good agreement with other numerical studies. Additionally, recent work has shown that finite cloud [115] meshless methods may be applied to solve vectorial mode fields in microstructured optical waveguides [116, 117].

In §6 the application of meshless methods to photonic crystals is explored in more detail before we go on to develop some novel meshless methods for photonic crystal modelling.

5.3.2 Radial basis functions

The excellent performance of radial basis functions for scattered data interpolation, which will be discussed in §5.3.3, motivates their use in developing meshless schemes for solving PDEs. Here, an overview of radial basis functions is given. A continuous function $\phi : \mathbb{R}^d \to \mathbb{R}$ is a radial basis function if $\phi(\boldsymbol{x}) = \phi(\boldsymbol{y})$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfying $\|\boldsymbol{x}\| = \|\boldsymbol{y}\|$, where the Euclidean norm is denoted $\| \cdot \|$, $\mathbb{R}^d$ is a d-dimensional space on $\mathbb{R}$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$.

In this section, we give the equations for various common RBFs that are seen in the literature and will be used later, along with some of their derivatives. The following paragraphs introduce the notation $\phi \equiv \phi(r)$ where $r = \|\boldsymbol{r}\|$ for any RBF evaluated for a distance r from the origin. The partial derivative of an RBF ϕ with respect to some variable v is denoted $\phi_v \equiv \partial\phi/\partial v$, for second derivatives $\phi_{v_1 v_2} = \partial^2\phi/\partial v_1 \partial v_2$, and we have the Laplace operator ∇^2 where, in 2D space, $\nabla^2\phi \equiv \partial^2\phi(r)/\partial x^2 + \partial^2\phi(r)/\partial y^2$, with $r = \sqrt{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2}$. All the RBFs given below have an adjustable shape parameter, and the illustrations show that the general trend across all the families of RBFs mentioned is that an increase of the shape parameter leads to a broader, flatter RBF shape.

Globally supported RBFs

The globally supported RBFs (GSRBFs) meet the definition in §5.3.2 and have a domain extending from $-\infty$ to ∞. As we shall see in §5.3.3, some GSRBFs have been proven to always lead to solvable interpolation problems. Some GSRBFs commonly found in the literature include the Gaussian, multiquadric, inverse multiquadric, and thin plate spline functions, which we examine below. Here, we denote the shape parameter c, in harmony with the notation used in the discussions of compactly-supported RBFs in §5.3.2, but we remark that in much of the literature, this parameter is denoted σ

instead.

Gaussian The Gaussian family of RBFs is illustrated in figs. 5.1(a)-5.1(c) for three values of the shape parameter, c. The RBF and some of its derivatives are given in (5.1-5.4).

$$\phi = \exp(-r^2/c) \tag{5.1}$$

$$\phi_x = -\frac{2x}{c}\exp(-r^2/c) \tag{5.2}$$

$$\phi_y = -\frac{2y}{c}\exp(-r^2/c) \tag{5.3}$$

$$\nabla^2\phi = \frac{4}{c}\exp(-r^2/c)\left(\frac{r^2}{c} - 1\right) \tag{5.4}$$

Multiquadric The multiquadric (MQ) family of RBFs is illustrated in figs. 5.1(d)-5.1(f). The RBF and some of its derivatives are given in (5.5-5.8).

$$\phi = (c^2 + r^2)^{1/2} \tag{5.5}$$

$$\phi_x = \frac{x}{(c^2 + r^2)} \tag{5.6}$$

$$\phi_y = \frac{y}{(c^2 + r^2)} \tag{5.7}$$

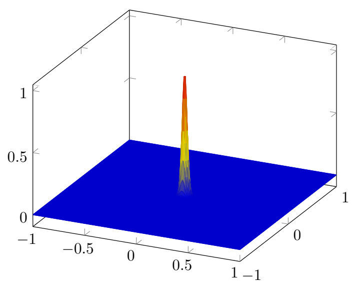
$$\nabla^2\phi = \frac{r^2 + 2c^2}{(r^2 + c^2)^{3/2}} \tag{5.8}$$

Inverse multiquadric The inverse multiquadric (IMQ) family of RBFs is illustrated in figs. 5.1(g)-5.1(i). The RBF and some of its derivatives are given in (5.9-5.12).

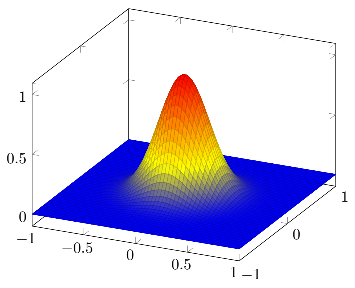
$$\phi = (c^2 + r^2)^{-1/2} \tag{5.9}$$

$$\phi_x = \frac{-x}{(c^2 + r^2)^{3/2}} \tag{5.10}$$

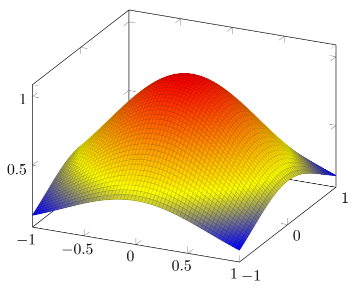
$$\phi_y = \frac{-y}{(c^2 + r^2)^{3/2}} \tag{5.11}$$



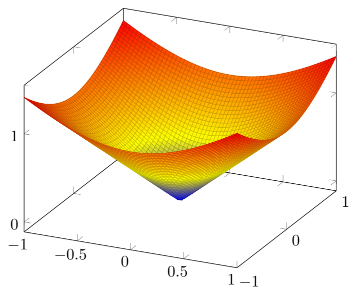
(a) Gaussian, $c = 0.01$



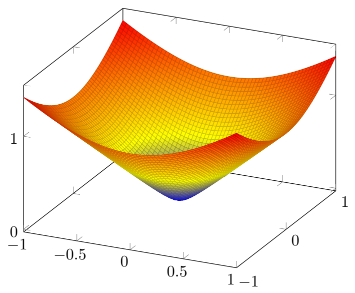
(b) Gaussian, $c = 0.1$



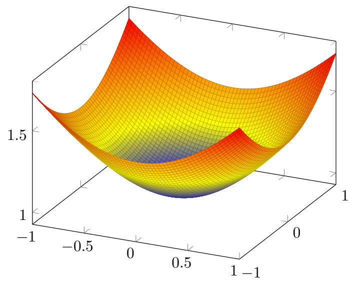
(c) Gaussian, $c = 1$



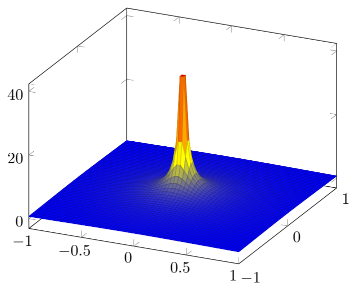
(d) Multiquadric, $c = 0.01$



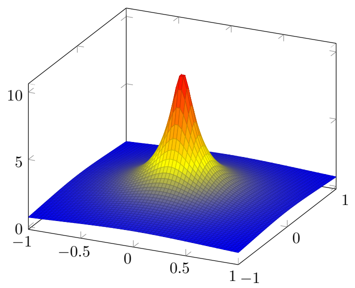
(e) Multiquadric, $c = 0.1$



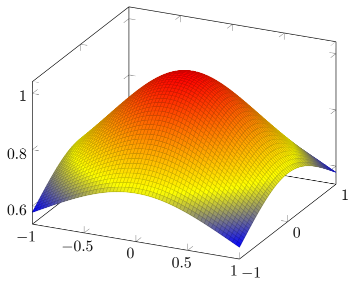
(f) Multiquadric, $c = 1$



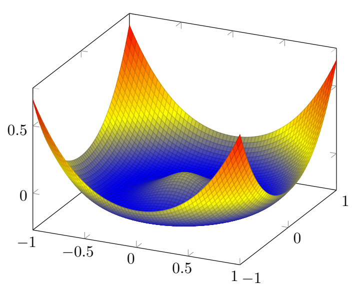
(g) Inverse multiquadric, $c = 0.01$



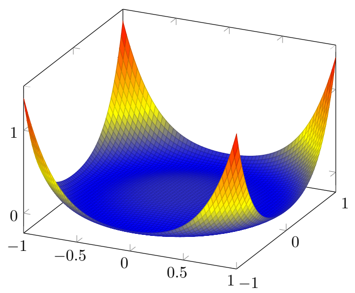
(h) Inverse multiquadric, $c = 0.1$



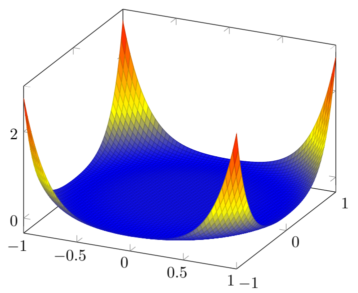
(i) Inverse multiquadric, $c = 1$



(j) Thin plate spline, $c = 1$



(k) Thin plate spline, $c = 2$



(l) Thin plate spline, $c = 3$

Figure 5.1: Various globally supported RBFs, illustrated for different values of the shape parameter c

$$\nabla^2\phi = \frac{3\left(x^2+y^2\right)}{\left(c^2+r^2\right)^{5/2}} - \frac{2}{\left(c^2+r^2\right)^{3/2}} \tag{5.12}$$

Thin plate spline The thin plate spline (TPS) family of RBFs is illustrated in figs. 5.1(j)-5.1(l). The RBF and some of its derivatives are given in (5.13-5.16).

$$\phi = r^{2c}\log(r) \tag{5.13}$$

$$\phi_x = xr^{2c-2}\left(1+2c\log r\right) \tag{5.14}$$

$$\phi_y = yr^{2c-2}\left(1+2c\log r\right) \tag{5.15}$$

$$\nabla^2\phi = 4cr^{2c-2}\left(1+c\log r\right) \tag{5.16}$$

Compactly supported RBFs

Compactly supported radial basis functions (CSRBFs) can also give rise to nonsingular interpolation problems. Their compactness can lead to increased sparseness and better conditioning of the system matrices, which makes them attractive for computational applications. Perhaps the most popular CSRBFs in the literature are those of Wu [118] and Wendland [119]. Both authors construct compactly supported, positive definite functions, which use a univariate polynomial within their support domain. The simplest of these is a cut-off polynomial; that is, a polynomial piece on $[0,1]$ which vanishes on $[1,\infty)$. More recently, Buhmann [120] proposed a new, larger class of smooth radial functions of compact support. Some typical CSRBFs used in the literature are illustrated in fig. 5.2.

A function which has k first continuous derivatives is called a C^k function. A CSRBF has the general form

$$\phi(r) = (1-r)_+^n\, p(r), \quad \text{for } k \geq 1, \tag{5.17}$$

where where $n = 2k+1$, $p(r)$ is a prescribed polynomial, and we have introduced the notation:

$$(1-r)_+^n = \begin{cases} (1-r)^n, & \text{for } 0 \le r < 1, \\ 0, & \text{for } r \ge 1. \end{cases} \tag{5.18}$$

Wu's CSRBFs Wu [118] introduced a series of positive definite compactly supported RBFs for various degrees of smoothness. In (5.19-5.26) we give two of the most popular forms used in the literature, expressed with r scaled by a shape parameter c so that $\phi(r)$ vanishes for $r \ge c$. Equations (5.19-5.22) give Wu's C^2 function and its derivatives:

$$\phi = \frac{(c-r)_+^5}{c^9}\left(8c^4 + 40rc^3 + 48r^2c^2 + 25r^3c + 5r^4\right) \tag{5.19}$$

$$\phi_x = \frac{x(c-r)_+^4}{c^9}\left(-144c^3 - 261rc^2 - 192r^2c^2 - 45r^3\right) \tag{5.20}$$

$$\phi_y = \frac{y(c-r)_+^4}{c^9}\left(-144c^3 - 261rc^2 - 192r^2c^2 - 45r^3\right) \tag{5.21}$$

$$\nabla^2\phi = \frac{9\,(c-r)_+^3}{c^9}\left(-32c^4 + 9rc^3 + 123r^2c^2 + 135r^3c + 45r^4\right) \tag{5.22}$$

Equations (5.23-5.26) give Wu's smoother C^4 function and selected derivatives:

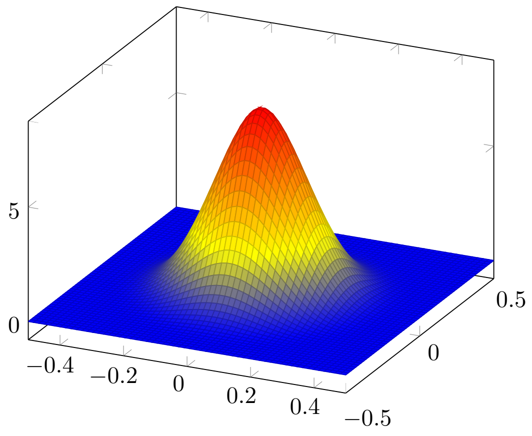
$$\phi = \frac{(c-r)_+^6}{c^{11}}\left(6c^5 + 36rc^4 + 82r^2c^3 + 72r^3c^2 + 30r^4c + 5r^5\right) \tag{5.23}$$

$$\phi_x = \frac{x(c-r)_+^5}{c^{11}}\left(-88c^4 - 440rc^3 - 528r^2c^2 - 550r^3c - 110r^4\right) \tag{5.24}$$

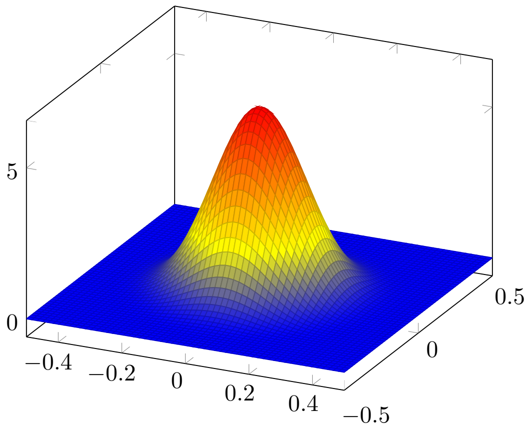
$$\phi_y = \frac{y(c-r)_+^5}{c^{11}}\left(-88c^4 - 440rc^3 - 528r^2c^2 - 550r^3c - 110r^4\right) \tag{5.25}$$

$$\nabla^2\phi = \frac{11\,(c-r)_+^4}{c^{11}}\left(-16c^5 - 64rc^4 + 128r^2c^3 + 307r^3c^2 + 220r^4c + 55r^5\right) \tag{5.26}$$

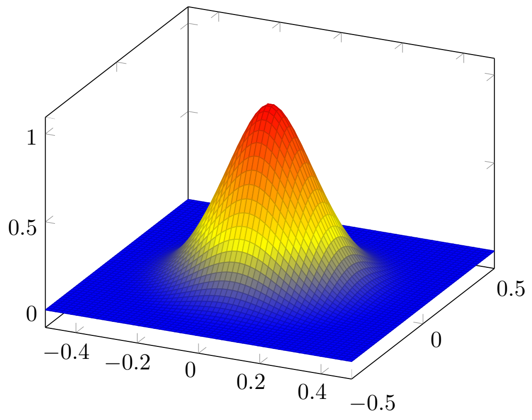
Wendland's CSRBFs In a paper published shortly after Wu's, Wendland [119] introduced a class of new CSRBFs which are of minimal degree given for a given smoothness, and showed that such functions are unique up to a constant factor. Wendland also showed Wu's functions to be special cases of his functions [120]. In (5.27-5.30) we



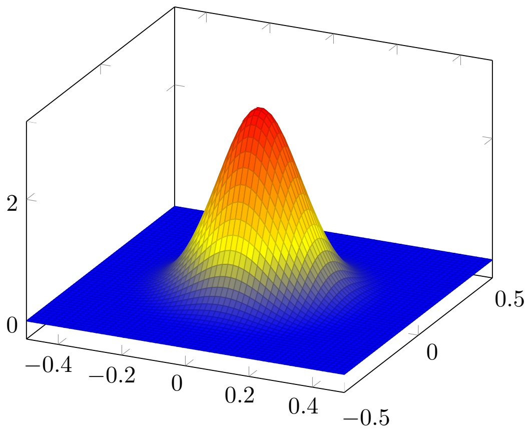
(a) Wu's C^2, $c = 0.5$



(b) Wu's C^4, $c = 0.5$



(c) Wendland's C^2, $c = 0.5$



(d) Wendland's C^4, $c = 0.5$

Figure 5.2: Compactly supported RBFs with shape parameter $c = 0.5$, from both Wu's and Wendland's families, and with two different smoothnesses. Note that the functions are identically zero for $r = \sqrt{x^2 + y^2} \geq c$.

give Wendland's C^2 function and its derivatives:

$$\phi = \frac{(c-r)_+^4}{c^5}(c+4r) \tag{5.27}$$

$$\phi_x = \frac{20x}{c^5}(c-r)_+^3 \tag{5.28}$$

$$\phi_y = \frac{20y}{c^5}(c-r)_+^3 \tag{5.29}$$

$$\nabla^2\phi = -\frac{20}{c^5}(c-r)_+^2(2c-5r) \tag{5.30}$$

Wendland's C^4 function and some of its derivatives are given in (5.31-5.34):

$$\phi = \frac{(c-r)_+^6}{c^8}\left(3c^2+18rc+35r^2\right) \tag{5.31}$$

$$\phi_x = \frac{56x(c-r)_+^5}{c^8}(-c-5r) \tag{5.32}$$

$$\phi_y = \frac{56y(c-r)_+^5}{c^8}(-c-5r) \tag{5.33}$$

$$\nabla^2\phi = -\frac{112}{c^8}(c-r)_+^4\left(c^2+4cr+20r^2\right) \tag{5.34}$$

5.3.3 RBF interpolation

RBFs have long been synonymous with scattered data interpolation, and are the foundations of a well-established and successful technique in the theory of multivariate function approximation. Given data $\{\boldsymbol{x}_i, f_i\} \in \mathbb{R}^n \times \mathbb{R}$, $1 \le i \le N$ specifying values of a function $f : \mathbb{R}^n \to \mathbb{R}$ on a finite set of distinct centres $\{\boldsymbol{x}_i\}_{i=1}^N \in \mathbb{R}^n$, the interpolant $F(\boldsymbol{x})$ approximating the function $f(\boldsymbol{x})$ is given by:

$$F(\boldsymbol{x}) = \sum_{j=1}^{N} \gamma_j \phi(\|\boldsymbol{x}-\boldsymbol{x}_j\|) + \beta, \tag{5.35}$$

where $\boldsymbol{x}$ and $\boldsymbol{x}_j$ are points in $\mathbb{R}^n$, the RBF is ϕ, the Euclidean norm on n-dimensional space is indicated by $\|\cdot\|$, and N is the total number of points. The coefficients γ_j and β may be found by setting

$$F(\boldsymbol{x}_i) = f(\boldsymbol{x}_i), \quad \text{for } i = 1, \ldots, N. \tag{5.36}$$

and imposing that $\sum_{j=1}^{N} \gamma_j = 0$. This gives rise to a symmetric linear system of equations,

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{e} \\ \boldsymbol{e}^T & 0 \end{bmatrix} \begin{bmatrix} \gamma \\ \beta \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ 0 \end{bmatrix}, \tag{5.37}$$

with $\boldsymbol{\Phi}_{ij} = \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$ for $i, j = 1, \ldots, N$, $\boldsymbol{u} = [f_1, \cdots, f_N]^T$, and $\boldsymbol{e}_i = 1$ for $i = 1, \ldots, N$.

The interpolation problem is well-posed if and only if the coefficient matrix in (5.37) is nonsingular, *i.e.* its inverse exists. It has been shown [121] that (5.37) can always be solved when using multiquadric interpolation and the following two conditions are met:

1. The points $\boldsymbol{x}_i$ are distinct, and
2. The degree of the appended polynomial is chosen to be the order of strictly conditionally positive definiteness of the RBF used.

5.3.4 Classes of meshless method

Meshless methods may be classified on various different characteristics of their formulation and implementation, including the way the domain is represented (domain or boundary), the function approximation scheme employed, and the formulation procedure used. In this section we expand on the latter and specifically split the methods into those based on strong forms, weak forms, and combinations of both.

In a strong form (or collocation) method, the approximate unknown function is assumed to be sufficiently smooth to be differentiable at least up to the order of the PDEs to be solved. Strong form methods are formulated by obtaining a set of discretized system equations using the strong forms of the governing equations and boundary conditions, directly discretized at the nodes. These methods, therefore, enjoy a high computational efficiency and are truly meshless (in contrast to weak form methods which require a set of background cells over which the requisite integration may be performed). They are also simple in algorithmic terms and may be implemented with relative ease. However, there are many situations in which they have significant downsides, such as a lack of robustness and stability, or in which they yield relatively inaccurate results.

Although the weak form methods have enjoyed more research effort than the strong form methods, there are many examples of strong form methods in the literature, including the collocation methods of Kansa, Wu, Zhang and Song *et al.*, etc [102, 103, 122], as well as Oñate *et al.*'s finite point method [123] and the generalised finite difference method [124].

The weak form methods do not require the same high degree of consistency of the unknown function as the strong form. In order to relax this requirement, an integral operation is introduced to the system equation, based on a mathematical or physical principle. The weak form thus requires more computation, but can usually produce highly stable discretized system equations giving rise to more accurate results than the strong form [76].

In formulating the weak form methods, the PDEs and boundary conditions are transformed into a set of weak form integral equations. The system equations are derived using a numerical integration process with the weak forms, over sets of background cells that are set up either locally or globally in the problem domain.

Significant contributions in the history of weak form methods include work by Nayroles *et al.*, who introduced the diffuse element method (DEM) using a (global) Galerkin weak form [100], and the introduction by Belytschko *et al.* of the element-free Galerkin (EFG) method [101] which was based upon the DEM. The local weak form methods based on Petrov-Galerkin formulations were developed initially by Atluri and Zhu [125].

It is possible to combine the weak and strong form methods, so as to have the advantages of the stability of the weak form formulation in the vicinity of natural boundaries, to obtain a stabilised solution, and the relative computational efficiency of the strong form method further from the boundaries. The meshless weak-strong form method was developed by Liu and Gu in 2002 [104]. They applied the local weak form to all nodes that are on or near boundaries with derivative boundary conditions, and the strong form for the other, so-called collocatable, nodes in the problem. This method can provide stable and accurate solutions in mechanics problems, using fewer background cells for the integration than competing weak-form methods.

There also exist other methods such as smooth particle hydrodynamics [126, 127], in which strong form equations are discretized at the particles but the function approximation is performed with a weak form.

5.3.5 Disadvantages of meshless methods

Although meshless methods possess attractive qualities, as detailed earlier in this section, they are of course not without drawbacks. Perhaps the chief drawbacks attending the meshless methods developed and applied in this thesis are those of long runtimes and poorly conditioned system matrices.

The problem of long runtimes arises in RBF-based methods largely due to the necessity to evaluate many RBFs and their derivatives in the course of assembling the system matrices. Numerical integration in the weak-form Galerkin methods is also a time-consuming process, requiring a large number of integration points for sufficient accuracy. However, computational acceleration technologies such as GPUs (see §2) are becoming increasingly popular and affordable, and have been successfully applied to ameliorate this disadvantage of meshless methods to some degree. One recent example is the use of multiple GPUs to accelerate a radial basis function finite difference (RBF-FD) PDE solver, where an unoptimised implementation on a GPU achieved a $9\times$ speedup compared to a CPU-only code [128]. Meshless methods also tend to suffer from a high memory requirement compared to the usual mesh based methods such as the FEM, FDM and FVM.

The problem of ill-conditioned, dense system matrices arises in many RBF methods, and for strong-form methods, the ill-conditioning increases with the number of collocation points, leading – in some cases – to an unsolvable system. One way to ameliorate this sort of issue is to solve many smaller sub-regions by collocation techniques, rather than one larger global domain problem which would require many collocation points [129]. Moreover, using schemes such as RBF-FD or a compactly-supported RBF-based method allows the sparseness of the matrix to be controlled via some adjustable shape parameter.

Another potential problem with meshless methods is that the development of the mathematical theory on meshless methods, including aspects such as stability and convergence order estimates, has been described as “far from satisfactory” [130]. However, some work has been carried out on estimating the bounds of the smallest eigenvalue of the collocation matrix and how the collocation matrix may be stabilised by smoothing [131], and on stability estimates for the meshless unsymmetric collocation method [130].

5.4 A meshless RBF-FD method for solving an eigenvalue problem with a periodic domain

Here we formulate a new meshless method using the radial basis function finite difference (RBF-FD) technique. We begin with an introduction to our problem (the elliptic Helmholtz equation) and to the RBF-FD technique, and then give our formulation. This is followed by results from our implementation, and we then go on to formulate a higher order RBF-FD scheme for the problem. The work in this section also forms the background for the RBF-FD method for calculating photonic band structure which we formulate in §6.5.

5.4.1 Introduction

As we stated in §5.1, many problems in engineering result in sets of partial differential equations, and sets of boundary conditions. In 2008, Hart *et al.* proposed a new meshless method to solve an eigenvalue problem with periodic boundary conditions [112]. In this case, CSRBFs are utilised and the shape parameter is kept less than or equal to half of the length of the domain to satisfy the boundary conditions. The compact sup-

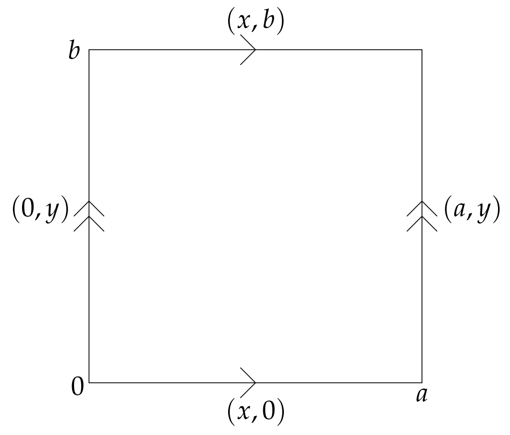


Figure 5.3: System domain with periodic boundary conditions.

port of these RBFs gives rise to a well-conditioned system matrix (compared to those generated by global RBF methods) with adjustable sparsity.

In this section we begin by describing the problem and its analytical solutions, and then introduce the formulation of a novel meshless method for the same problem, based upon an RBF-FD strong form formulation. The "local" nature of the RBF-FD method increases sparseness and improves the conditioning of the linear system [132, §1], analogously to the use of CSRBFs in Hart's work [112]. We give some numerical results demonstrating that the method is in good agreement with the analytical solution, and go on to formulate a higher-order scheme for the RBF-FD method to increase accuracy.

The method is formulated for the elliptic Helmholtz equation:

$$\nabla^2 u + \lambda^2 u = 0, \tag{5.38}$$

where ∇^2 is the Laplace operator, λ is a constant and the unknown function is u, defined on n-dimensional Euclidean space $\mathbb{R}^n$. For two dimensions, we have $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$.

This problem has analytical solutions, which may be obtained after imposing the periodic boundary conditions:

$$u(x, 0) = u(x, b), \quad u(0, y) = u(a, y), \tag{5.39}$$

and

$$u'(x,0) = u'(x,b), \quad u'(0,y) = u'(a,y), \tag{5.40}$$

where the domain extends from $(0,0)$ to (a,b) as shown in fig. 5.3.

Separating variables leads to the analytical solution of:

$$u(x,y) = X(x)Y(y), \tag{5.41}$$

with

$$X(x) = A\cos(lx) + B\sin(lx), \tag{5.42}$$

and

$$Y(y) = C\cos\left(y\sqrt{\lambda^2 - l^2}\right) + D\sin\left(y\sqrt{\lambda^2 - l^2}\right). \tag{5.43}$$

In (5.42-5.43), l is some constant and the boundary conditions may be enforced to give a set of solutions λ_{nk}^2 for l:

$$\lambda_{nk}^2 = \left(\frac{2\pi n}{b}\right)^2 + \left(\frac{2\pi k}{a}\right)^2, \quad \text{for } n,k = 0,1,2,\ldots. \tag{5.44}$$

For a square domain extending from $(0,0)$ to (a,a), this reduces to

$$\lambda_{nk} = \frac{2\pi}{a}\sqrt{n^2 + k^2}. \tag{5.45}$$

5.4.2 Formulation

Radial basis functions may be applied in finite-difference mode, which may be seen as analogous to the generalised finite difference schemes [133], but using arbitrary or random point positions rather than a fixed grid system. The idea of using RBFs with a local collocation, as in finite differences, reduces the number of connections (the so-called support) for each node, hence producing a sparse and better-conditioned matrix (unlike the global RBF methods which produce dense and ill-conditioned matrices when the number of nodes increases). The scheme was introduced independently by Tolstykh and Shirobokov [134], and Wright and Fornberg [132], in the literature. Chinchapatnam *et al.* later developed RBF-FD methods for application to the incompressible Navier-Stokes equations [110].

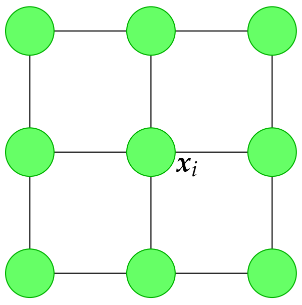


Figure 5.4: Schematic representation of an RBF-FD stencil around the node $\boldsymbol{x}_i$, with 9 nodes in total. Nodes 🟢 are those at which we use the values of u.

In the RBF-FD method, the complete domain is represented by a set of N nodes, which may be arbitrarily located. For each node, an influence domain (or stencil) is defined, which consists of M nodes near to the node under consideration. Such a stencil is represented schematically in fig. 5.4. Because the boundary conditions in this problem are periodic, distances between nodes are calculated using the minimum image distance (that is, the minimum of the distance between nodes within the unit cell, or that between a node in the unit cell and the appropriate node in a periodic repeat of the unit cell, see *e.g.* [135, §A.1]), rather than the absolute distance. At each node, a local RBF interpolation problem is set up to determine the RBF-FD coefficients to represent the interpolant, and the weights to represent its derivatives. The diagram in fig. 5.5 illustrates an influence domain for node $\boldsymbol{x}_1$ under these periodic boundary conditions. The nodes are shown arranged on a regular rectangular spacing for visual clarity, but this is not a requirement of the algorithm.

In a conventional finite difference scheme the derivative of the function $u(x, y)$ with respect to x at some grid point (i, j) can be estimated using a central difference expression like:

$$\left.\frac{\partial u}{\partial x}\right|_{(i,j)} \approx \sum_{k \in \{i-1,i,i+1\}} w_{(k,j)} u(k, j), \tag{5.46}$$

where the function value at the grid point (k, j) is $u(k, j)$ and the coefficients $w_{(k,j)}$ can be obtained via a Taylor series or a polynomial interpolation. The set of nodes $\{(i - 1, j), (i, j), (i + 1, j)\}$ are often referred to as the stencil in finite difference literature.

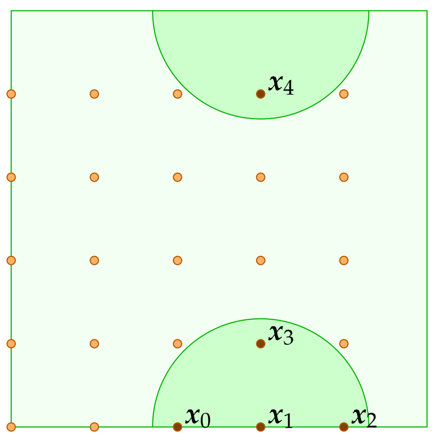


Figure 5.5: The nodes of the system, and an RBF-FD influence domain or stencil. The unit cell is the outer square. The small circles illustrate the nodes (shown uniformly spaced for visual clarity). A stencil for node x_1, which includes those nodes within a radius of $r_p = 0.26a$, is shown with a darker background.

Using conventional interpolation techniques requires that the nodes are situated on a structured grid, as explained in §5.2.

The RBF-FD concept is to use a formula like (5.46), but to compute the weights using the RBF interpolation technique rather than the traditional polynomial or Taylor series approaches. This brings several advantages, namely that the method is truly meshless, requiring no information about the connectivity of the nodes, as well as utilising the accuracy of RBF interpolants in approximating derivatives, and overcoming the problem of well-posedness in the polynomial interpolation scheme, since RBF interpolation is well posed in multidimensional problems.

We now turn our attention to formulating the RBF-FD method for this problem. Recall that the standard RBF interpolation problem seeks an interpolant of the form in (5.35), with coefficients that may be determined from (5.37). The RBF interpolation problem here is of the standard form:

$$u(\boldsymbol{x}) \approx s(\boldsymbol{x}) = \sum_{j=1}^{M} \gamma_j \phi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) + \beta, \tag{5.47}$$

where $\phi(\|\cdot\|)$ is a (globally supported) RBF, M is the number of nodes in the influence domain, and β is a constant.

In Lagrangian form, (5.47) can be written as

$$\bar{s}(\boldsymbol{x}) = \sum_{j=1}^{M} \chi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)u(\boldsymbol{x}_j), \tag{5.48}$$

where $\chi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)$ is of the form (5.47) and satisfies the usual cardinal conditions, i.e.,

$$\chi(\|\boldsymbol{x}_k - \boldsymbol{x}_j\|) = \begin{cases} 1, & \text{if } k = j, \\ 0, & \text{if } k \neq j, \end{cases} \quad k = 1, \cdots, N. \tag{5.49}$$

The goal is to write the approximations of function derivatives as a linear combination of function values, as in (5.46). Here, we derive the RBF-FD approximation for an arbitrary linear operator $\mathcal{L}$ operating on $u(\boldsymbol{x})$. The function $u(\boldsymbol{x})$ is represented at any node as an RBF interpolant whose centres are on the node and the $M - 1$ surrounding nodes. We pick node $\boldsymbol{x}_1$ (see fig. 5.5) for this example, although the process is repeated for all the nodes in the domain. We approximate the differential operator at the node by applying the operator to the Lagrangian form of the RBF interpolant:

$$\mathcal{L}u(\boldsymbol{x}_1) \approx \mathcal{L}\bar{s}(\boldsymbol{x}_1) = \sum_{j=1}^{M} \mathcal{L}\chi(\|\boldsymbol{x}_1 - \boldsymbol{x}_j\|)u(\boldsymbol{x}_j) \tag{5.50}$$

Equation (5.50) can be rewritten as a FD formula of the form

$$\mathcal{L}u(\boldsymbol{x}_1) \approx \sum_{j=1}^{M} w_{(1,j)}u(\boldsymbol{x}_j), \tag{5.51}$$

where the RBF-FD weights $\{w_{(1,j)}\}_{j=1}^{M}$ are formally given by the operator $\mathcal{L}$ applied on the Lagrange form of the basis functions, *i.e.*,

$$w_{(1,j)} = \mathcal{L}\chi(\|\boldsymbol{x}_1 - \boldsymbol{x}_j\|), \tag{5.52}$$

We compute the weights by applying Gaussian elimination to solve the linear system

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{e} \\ \boldsymbol{e}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathcal{L}\boldsymbol{\varphi}_1 \\ 0 \end{bmatrix}, \tag{5.53}$$

where $\boldsymbol{e}_i = 1$, and $\mathcal{L}\boldsymbol{\varphi}_1$ denotes the column vector $\mathcal{L}\boldsymbol{\varphi} = [\mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_1\|), \mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_2\|), \cdots, \mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_M\|)]^T$ evaluated at the node $\boldsymbol{x}_1$. The matrix $\boldsymbol{\Phi}$ is given by $\boldsymbol{\Phi}_{ij} = \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$ and μ is a scalar value which enforces the condition

$$\sum_{j=1}^{M} w_{(1,j)} = 0,$$

ensuring that the stencil is exact for all constants.

The above procedure is formulated with an arbitrary linear differential operator $\mathcal{L}$, and so the weights $\boldsymbol{w}$ corresponding to any such operator may be found this way.

To apply this technique to the eigenvalue problem at hand, it is first necessary to calculate the weights $\{w_{(1,i)}^{(xx)}\}_{i=1}^{M}$ for $\partial^2 u/\partial x^2|_{\boldsymbol{x}=\boldsymbol{x}_1}$ and $\{w_{(1,i)}^{(yy)}\}_{i=1}^{M}$ for $\partial^2 u/\partial y^2|_{\boldsymbol{x}=\boldsymbol{x}_1}$. The periodic boundary conditions are enforced in the definition of the stencil for the node $\boldsymbol{x}_1$, as shown in fig. 5.5, and in using the minimum image distances in evaluating functions arguments. Thus, at the node $\boldsymbol{x}_1$, we identify the set $\Omega_1 = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}$ as the stencil (with reference to fig. 5.5), and expand (5.53) as follows:

$$\begin{bmatrix} \phi(\|\boldsymbol{x}_0 - \boldsymbol{x}_0\|) & \phi(\|\boldsymbol{x}_0 - \boldsymbol{x}_1\|) & \cdots & \phi(\|\boldsymbol{x}_0 - \boldsymbol{x}_4\|) & 1 \\ \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|) & \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_1\|) & \cdots & \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_4\|) & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi(\|\boldsymbol{x}_4 - \boldsymbol{x}_0\|) & \phi(\|\boldsymbol{x}_4 - \boldsymbol{x}_1\|) & \cdots & \phi(\|\boldsymbol{x}_4 - \boldsymbol{x}_4\|) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} w_{(1,0)}^{(xx)} \\ w_{(1,1)}^{(xx)} \\ \vdots \\ w_{(1,4)}^{(xx)} \\ \mu \end{bmatrix} = \begin{bmatrix} \phi_{,xx}(\boldsymbol{x}_1, \boldsymbol{x}_0) \\ \phi_{,xx}(\boldsymbol{x}_1, \boldsymbol{x}_1) \\ \vdots \\ \phi_{,xx}(\boldsymbol{x}_1, \boldsymbol{x}_4) \\ 0 \end{bmatrix}, \tag{5.54}$$

where $\phi_{,xx}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\partial^2}{\partial x^2} \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)|_{\boldsymbol{x}=\boldsymbol{x}_j}$, with an analogous system for the $\partial^2/\partial y^2$ operator to determine $\{w_{(1,i)}^{(yy)}\}_{i=1}^{M}$. These equations are solved using Gaussian elimination. Then, the procedure is repeated at all other nodes in the system, and the weights are stored. The RBF-FD weights are clearly functions only of the relative positions of the nodes and the RBF employed. Thus, such weights may be pre-calculated if the inter-node distances are known in advance.

The discretization of (5.38) leads to an eigenvalue problem:

$$\mathbf{L}\boldsymbol{u} = -\lambda^2 \boldsymbol{u}, \tag{5.55}$$

with the eigenvalues λ^2 corresponding to the various λ values in the analytical solution (5.45). The $N \times N$ matrix $\mathbf{L}$ is constructed using the RBF-FD weights calculated at each node, and its sparsity can be varied by changing the number of nodes used in the stencils when calculating these weights. The elements $\mathbf{L}_{ij}$ of this matrix can be expressed as follows:

$$\mathbf{L}_{ij} = \begin{cases} w^{(xx)}_{(i,j)} + w^{(yy)}_{(i,j)}, & \text{if } j \in \Omega_i, \\ 0, & \text{otherwise,} \end{cases} \tag{5.56}$$

where the notation Ω_i represents the set of nodes comprising the stencil for node at $\boldsymbol{x}_i$.

With the matrix assembled, an eigensolver routine is called to solve the problem for the eigenvalues λ which are then compared to the analytical solutions to determine the accuracy of the method. For the results in §5.4.4, the eigensolver chosen was a Krylov-Schur iterative algorithm provided as part of the SLEPc library [136].

5.4.3 Formulation of a higher-order RBF-FD scheme

In §5.4.2, we formulated an RBF-FD method for solving an eigenvalue problem on a periodic domain, and in §5.4.4 we shall give some numerical results from this method. For a fixed stencil size, it will be observed that the accuracy of the method increases for larger numbers of nodes (smaller inter-node distances h), and for increasing shape parameters. However, increasing the shape parameter too severely will be found found to produce ill-conditioning of either the eigenvalue problem (5.55) or the linear system (5.54).

Another way to increase the accuracy of an RBF-FD method, without changing the size of the stencil or the number of nodes used, is to employ a higher-order RBF-FD scheme, using ideas from Hermite interpolation. In this section, we will introduce and formulate such a scheme.

Wright and Fornberg [132] proposed to keep the stencil size fixed and include in the RBF-FD approximation of the derivative some linear combination of the derivatives of u at the surrounding nodes, as illustrated schematically in fig. 5.6. This work was built upon Collatz's original *Mehrstellenverfahren* [137], which was developed by Lele into compact finite difference formulæ [138]. In a higher-order finite difference scheme,

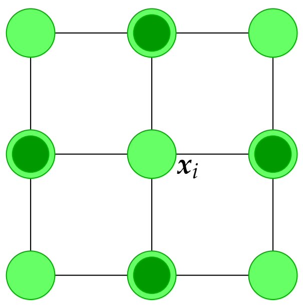


Figure 5.6: A schematic representation of a higher-order RBF-FD stencil for node x_i, with nine nodes in total. Nodes ● are those at which we use the values of u whilst nodes ● are those at which we also consider the additional information, $\partial u/\partial x$.

including this derivative information might transform (5.46) into:

$$\left.\frac{\partial u}{\partial x}\right|_{(i,j)} \approx \sum_{k\in\{i-1,i,i+1\}} w_{(k,j)} u_{(k,j)} + \sum_{k\in\{i-1,i+1\}} \widetilde{w}_{(k,j)} \left.\frac{\partial u}{\partial x}\right|_{(k,j)}, \tag{5.57}$$

where the additional term includes the derivative information, with its own set of weights $\{\widetilde{w}_{(k,j)}\}$, without changing the size of the stencil employed.

To formulate the higher order RBF-FD method, we first construct an interpolant analogous to (5.35), and then impose conditions similar to (5.36), giving rise to a block linear system of equations reminiscent of (5.37).

Taking $\mathcal{L}$ to be an arbitrary linear differential operator and the unknown function to be $u(\boldsymbol{x})$, we require the function values $\{u(\boldsymbol{x}_i)\}$ at each of the N nodes $\{\boldsymbol{x}_i\}_{i=1}^N$. We introduce $\boldsymbol{\eta}$, a vector containing $m \leq N$ of the numbers $\{1,\ldots,N\}$. The derivative information we require is the data $\mathcal{L}u(\boldsymbol{x}_{\eta_l})$ at the nodes $\{\boldsymbol{x}_{\eta_l}\}_{l=1}^m$, which are a subset of the nodes $\{\boldsymbol{x}_i\}_{i=1}^N$. Then, the interpolant may be expressed:

$$u(\boldsymbol{x}) \approx s(\boldsymbol{x}) = \sum_{i=1}^{N} \gamma_i \phi(\|\boldsymbol{x}-\boldsymbol{x}_i\|) + \sum_{l=1}^{m} \widetilde{\gamma}_l \mathcal{L}_2\phi(\|\boldsymbol{x}-\boldsymbol{x}_{\eta_l}\|) + \beta, \tag{5.58}$$

where $\mathcal{L}_2\phi(\|\cdot\|)$ is a basis function derived by applying the operator $\mathcal{L}$ to the basis function $\phi(\|\cdot\|)$ as a function of the second variable, and and β is a constant as in the lower-order case.

The conditions we enforce to obtain the unknowns are:

$$\begin{aligned} s(\boldsymbol{x}_i) &= u(\boldsymbol{x}_i), & i &= 1,\dots,N, \\ \mathcal{L}s(\boldsymbol{x}_{\eta_l}) &= \mathcal{L}u(\boldsymbol{x}_{\eta_l}), & l &= 1,\dots,m, \\ \sum_{i=1}^{N} \gamma_i &= 0. \end{aligned} \tag{5.59}$$

As before, these conditions lead to a system of block linear equations:

$$\begin{bmatrix} \boldsymbol{\Phi} & \mathcal{L}_2\boldsymbol{\Phi} & \boldsymbol{e} \\ \mathcal{L}\boldsymbol{\Phi} & \mathcal{L}\mathcal{L}_2\boldsymbol{\Phi} & \boldsymbol{0} \\ \boldsymbol{e}^T & \boldsymbol{0}^T & 0 \end{bmatrix} \begin{bmatrix} \gamma \\ \widetilde{\gamma} \\ \beta \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \mathcal{L}\boldsymbol{u} \\ 0 \end{bmatrix}, \tag{5.60}$$

with

$$\begin{aligned} \boldsymbol{\Phi}_{ij} &= \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), & i &= 1,\dots,N, \quad j = 1,\dots,N, \\ \mathcal{L}_2\boldsymbol{\Phi}_{ij} &= \mathcal{L}_2\phi(\|\boldsymbol{x}_i - \boldsymbol{x}_{\eta_j}\|), & i &= 1,\dots,N, \quad j = 1,\dots,m, \\ \mathcal{L}\boldsymbol{\Phi}_{ij} &= \mathcal{L}\phi(\|\boldsymbol{x}_{\eta_i} - \boldsymbol{x}_j\|), & i &= 1,\dots,m, \quad j = 1,\dots,N, \\ \mathcal{L}\mathcal{L}_2\boldsymbol{\Phi}_{ij} &= \mathcal{L}\mathcal{L}_2\phi(\|\boldsymbol{x}_{\eta_i} - \boldsymbol{x}_{\eta_j}\|), & i &= 1,\dots,m, \quad j = 1,\dots,m, \\ \boldsymbol{e}_i &= 1 & i &= 1,\dots,N. \end{aligned} \tag{5.61}$$

We write the interpolant (5.58) in the Lagrange form,

$$\bar{s}(\boldsymbol{x}) = \sum_{i=1}^{N} \chi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) u(\boldsymbol{x}_i) + \sum_{l=1}^{m} \widetilde{\chi}(\|\boldsymbol{x} - \boldsymbol{x}_{\eta_l}\|) \mathcal{L}u(\boldsymbol{x}_{\eta_l}), \tag{5.62}$$

with $\chi(\|\boldsymbol{x} - \boldsymbol{x}_i)\|)$ and $\widetilde{\chi}(\|\boldsymbol{x} - \boldsymbol{x}_{\eta_l}\|)$ of the form of (5.58). These terms satisfy the cardinal conditions,

$$\chi(\|\boldsymbol{x}_k - \boldsymbol{x}_i\|) = \begin{cases} 1, & \text{if } k = i, \\ 0, & \text{if } k \neq i, \end{cases} \qquad k = 1,\dots,N, \tag{5.63}$$

and

$$\mathcal{L}\chi(\|\boldsymbol{x}_{\eta_k} - \boldsymbol{x}_i\|) = 0, \quad k = 1,\dots,m, \tag{5.64}$$

and for the $\widetilde{\chi}$ terms, we have

$$\widetilde{\chi}(\|\boldsymbol{x}_k - \boldsymbol{x}_{\eta_l}\|) = 0, \quad k = 1, \ldots, N, \tag{5.65}$$

and

$$\mathcal{L}\widetilde{\chi}(\|\boldsymbol{x}_{\eta_k} - \boldsymbol{x}_{\eta_l}\|) = \begin{cases} 1, & \text{if } k = l, \\ 0, & \text{if } k \neq l, \end{cases} \qquad k = 1, \ldots, m. \tag{5.66}$$

We can then write a higher-order RBF-FD discretization of $\mathcal{L}u(\boldsymbol{x}_1)$ based upon the Lagrangian form, as we did in the initial RBF-FD scheme in (5.50):

$$\mathcal{L}u(\boldsymbol{x}_1) \approx \mathcal{L}\bar{s}(\boldsymbol{x}_1) = \sum_{i=1}^{n} \mathcal{L}\chi(\|\boldsymbol{x}_1 - \boldsymbol{x}_i\|)u(\boldsymbol{x}_i) + \sum_{l=1}^{\widetilde{m}} \mathcal{L}\widetilde{\chi}(\|\boldsymbol{x}_1 - \boldsymbol{x}_{\eta_l}\|)\mathcal{L}u(\boldsymbol{x}_{\eta_l}). \tag{5.67}$$

Here, the set of nodes $\{\boldsymbol{x}_i\}_{i=1}^{n}$ are those in the stencil for node $\boldsymbol{x}_1$ at which we use the value $u(\boldsymbol{x})$, whilst the set $\{\boldsymbol{x}_{\eta_l}\}_{l=1}^{\widetilde{m}}$ are those at which we also use the derivative information $\mathcal{L}u(\boldsymbol{x})$. Introducing the weights for the higher-order RBF-FD scheme as $\{w^{\mathcal{L}}_{(1,i)}\}_{i=1}^{n}$ and $\{\widetilde{w}^{\mathcal{L}}_{(1,l)}\}_{l=1}^{\widetilde{m}}$, we may rewrite (5.67) as a compact FD formula,

$$\mathcal{L}u(\boldsymbol{x}_1) \approx \sum_{i=1}^{n} w^{\mathcal{L}}_{(1,i)} u(\boldsymbol{x}_i) + \sum_{l=1}^{\widetilde{m}} \widetilde{w}^{\mathcal{L}}_{(1,l)} \mathcal{L}u(\boldsymbol{x}_{\eta_l}), \tag{5.68}$$

where we have

$$\begin{aligned} w^{\mathcal{L}}_{(1,i)} &= \mathcal{L}\chi(\|\boldsymbol{x}_1 - \boldsymbol{x}_i\|), \\ \widetilde{w}^{\mathcal{L}}_{(1,l)} &= \mathcal{L}\widetilde{\chi}(\|\boldsymbol{x}_1 - \boldsymbol{x}_{\eta_l}\|), \end{aligned} \tag{5.69}$$

and the superscript $\mathcal{L}$ indicates the operator for which these weights were calculated.

We compute the weights in practise by applying Gaussian elimination to solve the linear system

$$\begin{bmatrix} \boldsymbol{\Phi} & \mathcal{L}_2\boldsymbol{\Phi} & \boldsymbol{e} \\ \mathcal{L}\boldsymbol{\Phi} & \mathcal{L}\mathcal{L}_2\boldsymbol{\Phi} & \boldsymbol{0} \\ \boldsymbol{e}^T & \boldsymbol{0}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^{\mathcal{L}} \\ \widetilde{\boldsymbol{w}}^{\mathcal{L}} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathcal{L}^*\boldsymbol{\varphi}_1 \\ \mathcal{L}^*\widetilde{\boldsymbol{\varphi}}_1 \\ 0 \end{bmatrix}, \tag{5.70}$$

where $\mathcal{L}^*\boldsymbol{\varphi}_1$ denotes the column vector $\mathcal{L}^*\boldsymbol{\varphi} = [\mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_1\|), \cdots, \mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_n\|)]^T$, and $\mathcal{L}^*\widetilde{\boldsymbol{\varphi}}_1$ is the column vector $\mathcal{L}^*\widetilde{\boldsymbol{\varphi}}_1 = [\mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_{\eta_1}\|), \cdots, \mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_{\eta_{\widetilde{m}}}\|)]^T$, both evaluated at the node $\boldsymbol{x}_1$. Here, μ is a scalar value which enforces the condition

$$\sum_{i=1}^{n} w^{\mathcal{L}}_{(1,i)} = 0,$$

ensuring that the stencil is exact for all constants.

The above formulation is given for node $\boldsymbol{x}_1$, and the procedure is carried out for each node $\boldsymbol{x}_i$, $i = 1, \ldots, n$, in order to calculate all the required RBF-FD weights.

With the weights known, the discretization of (5.38) at the node $\boldsymbol{x}_1$ gives:

$$\sum_{j=1}^{n} w^{\mathcal{L}}_{(1,j)} u_j + \sum_{l=1}^{\widetilde{m}} \widetilde{w}^{\mathcal{L}}_{(1,\eta_l)} \mathcal{L} u_{\eta_l} = -\lambda^2 u(\boldsymbol{x}_1) \tag{5.71}$$

This can be rewritten in a form suggestive of a generalised eigenvalue problem (where we have substituted from (5.55)):

$$\sum_{j=1}^{n} w^{\mathcal{L}}_{(1,j)} u_j = -\lambda^2 \left[u(\boldsymbol{x}_1) - \sum_{l=1}^{\widetilde{m}} \widetilde{w}^{\mathcal{L}}_{(1,\eta_l)} u_{\eta_l} \right] \tag{5.72}$$

Considering all the N nodes in the system, the generalised eigenvalue problem formed is:

$$\mathbf{L}\boldsymbol{u} = -\lambda^2 \mathbf{G}\boldsymbol{u} \tag{5.73}$$

with the $\mathbf{L}$ matrix given by

$$\mathbf{L}_{ij} = \begin{cases} w^{\nabla^2}_{(i,j)}, & \text{if } j \in \Omega_i, \\ 0, & \text{otherwise,} \end{cases} \tag{5.74}$$

and the $\mathbf{G}$ matrix given by

$$\mathbf{G}_{ij} = \begin{cases} 1, & \text{if } i = j, \\ -\widetilde{w}^{\nabla^2}_{(i,j)}, & \text{if } j \in \eta_i, \\ 0, & \text{otherwise,} \end{cases} \tag{5.75}$$

where Ω_i represents the stencil for node i, i.e. the nodes at which we consider $u(\boldsymbol{x})$. Again, we note that by adjusting the size of the stencil Ω_i we are able to control the sparseness of the matrix $\mathbf{L}$, whilst adjusting the size of the subset $\Omega_i^{\nabla^2}$ adjusts the sparseness of $\mathbf{G}$.

With the matrices assembled, the generalised eigenvalue problem (5.73) is passed to a vendor-optimised eigensolver and the calculated eigenvalues λ are compared to the analytical solution.

5.4.4 Results

We implemented the algorithm using Gaussian elimination to solve the linear systems (5.53) and a Krylov-Schur iterative eigensolver [136] to find the eigenvalues of (5.55). Here we present results showing how the accuracy of the method varies with various of its parameters. We set the domain size $a = b = 1$ and we use eigenvalues scaled by π, i.e., $\lambda_{\mathrm{s}} = \lambda/\pi$.

We defined the relative error between an analytical and a numerical result to be:

$$\epsilon_{\mathrm{r}} = \frac{\|\lambda_{\mathrm{a}} - \lambda_{\mathrm{n}}\|}{\|\lambda_{\mathrm{a}}\|}, \tag{5.76}$$

with the analytical eigenvalue denoted λ_{a} and the numerical results λ_{n}. We compare the numerical and analytical results for the scaled eigenvalue $\lambda_{\mathrm{s}} = 2$ which has a four-fold degeneracy ($n = 0, k = \pm 1$ and $k = 0, n = \pm 1$ in (5.45)). The reported values of ϵ_{r} are the mean of ϵ_{r} calculated for the four $\lambda_{\mathrm{s}} = 2$ results.

In fig. 5.7 we show the relative error ϵ_{r} as a function of the total number of nodes N in the system. The stencil employed in each case was a circle with radius set so that 29 nodes were admitted on the uniformly spaced grid at each value of N. We used the multiquadric RBF with shape parameter $c = 0.2$. Results are shown for three different node layouts: uniformly distributed nodes on a square grid, nodes distributed according to a Sobol sequence, and uniformly distributed nodes perturbed randomly within a radius of $0.3h$ from their original position, where h is the distance between adjacent uniform nodes.

It can be seen that increasing the number of nodes leads to a decreasing relative error. The difference in relative error between the three node layouts is less than an order of magnitude in all cases, but for large numbers of nodes the uniform distribution consistently yields the best accuracies, whilst for small and moderate numbers of

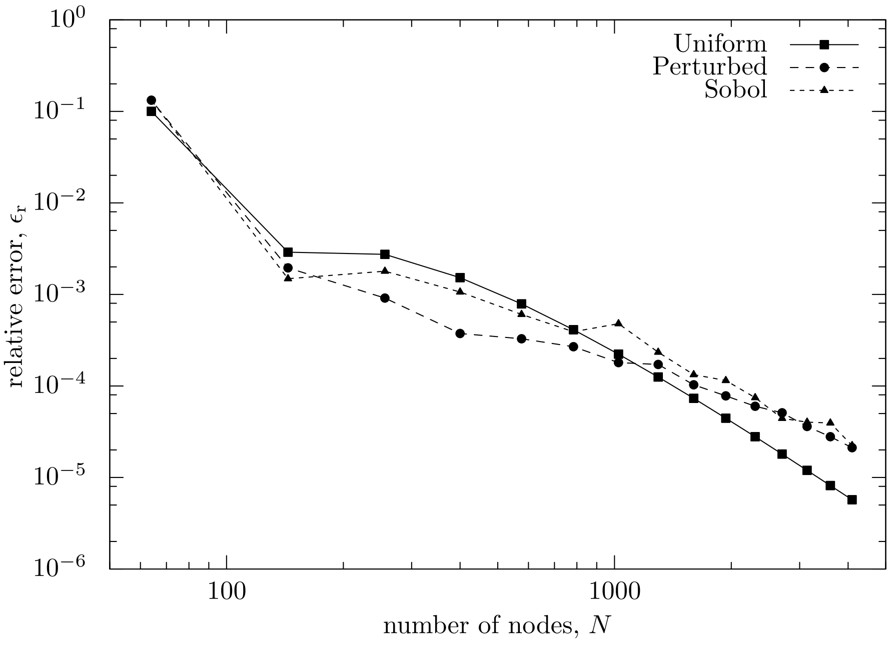


Figure 5.7: Mean relative error ϵ_r as a function of total number of nodes N for three point layouts, with a stencil of fixed shape (standard order method, multiquadric RBF, $c = 0.2$).

nodes, the perturbed pattern often performs best.

In fig. 5.8 we illustrate the mean relative error ϵ_r as a function of the number of nodes N for the higher order and the standard order methods, using uniformly distributed nodes. As in fig. 5.7 we used the multiquadric RBF, with $c = 0.2$ and an influence domain that admits 29 nodes at each resolution, and in the higher order method we included all nodes other than i from Ω_i in η_i.

For all the numbers of nodes that we considered, the higher-order method produced better overall relative errors, and it demonstrates a considerably more rapid convergence toward the analytical solution as the number of nodes increases when compared to the standard-order method.

Convergence rates for the standard and higher order methods are determined according to fig. 5.9. The error is modelled in terms of the mesh spacing (*i.e.*, the inter-

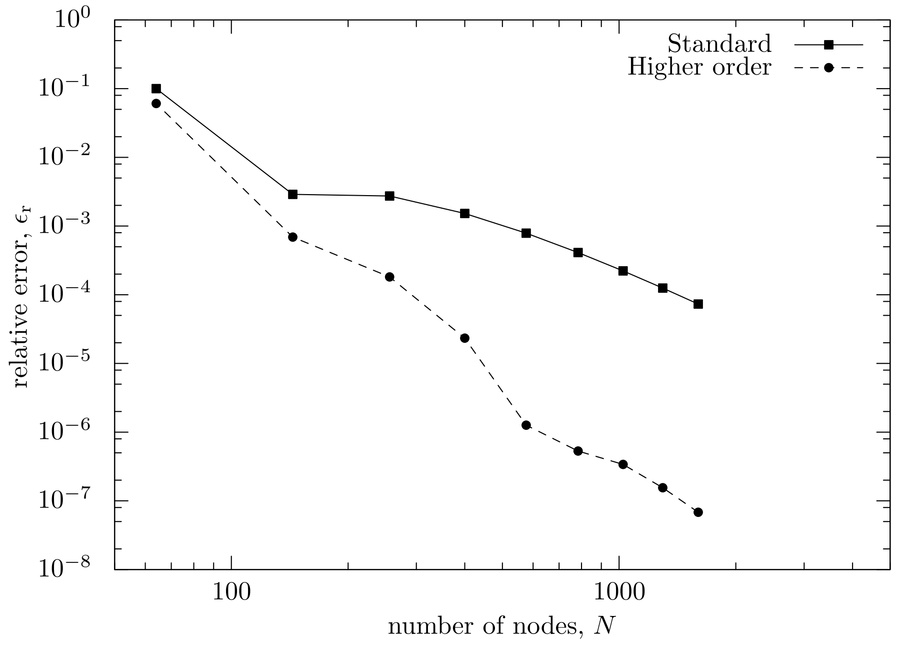


Figure 5.8: Comparison of mean relative error ϵ_r as a function of total number of nodes N for the higher order and standard order methods, using uniformly distributed nodes (multiquadric RBF, $c = 0.2$).

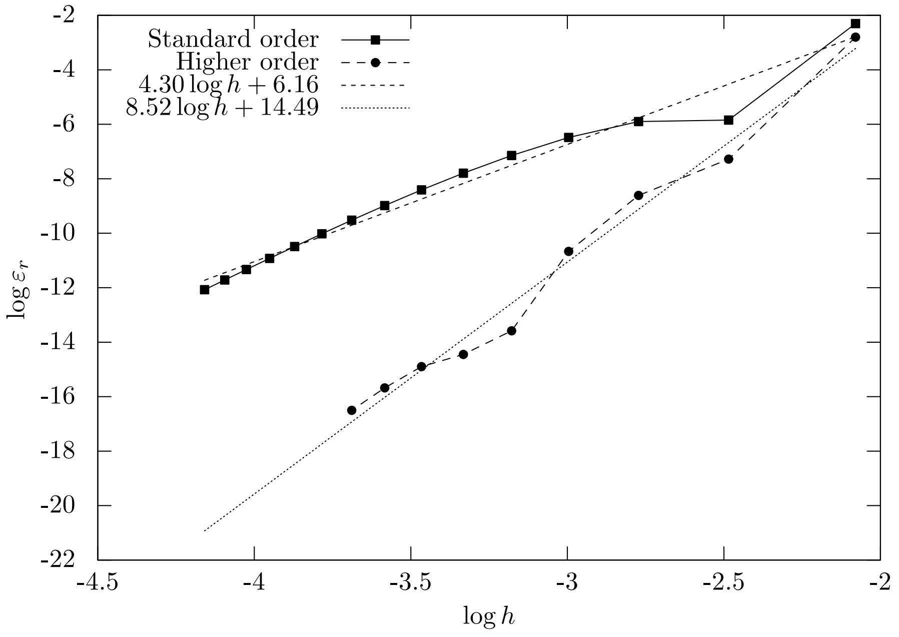


Figure 5.9: Determination of the orders of convergence for the standard and higher order methods by plotting $\log \epsilon_{\mathrm{r}}$ against $\log h$ (using uniformly distributed nodes with the multiquadric RBF and shape parameter $c = 0.2$).

node distance) h as a power law,

$$\epsilon_{\mathrm{r}}(h) \approx Ch^p,$$

with C a constant scale factor and p defined as the order of convergence. Taking logarithms, we have $\log \epsilon_{\mathrm{r}} = \log C + p \log h$. Plotting $\log \epsilon_{\mathrm{r}}$ against $\log h$ and applying the nonlinear least-squares Marquardt-Levenberg algorithm [139, 140] to fit straight lines to the data yields the lines shown in the the figure. We therefore find that the orders of convergence for the standard-order method and the higher-order method are $\mathcal{O}(h^{4.3})$ and $\mathcal{O}(h^{8.5})$ respectively. These compare favourably to the convergence rates for the central difference form of the finite difference method, which is $\mathcal{O}(h^2)$, and to the CSRBF-based methods, which are of order $\mathcal{O}(h^3)$ when using C2 functions and $\mathcal{O}(h^5)$ with C4 functions [112].

In fig. 5.10, we show the runtimes of the RBF-FD method code for both standard and higher orders, using the parameters that were used to generate the data for fig. 5.8.

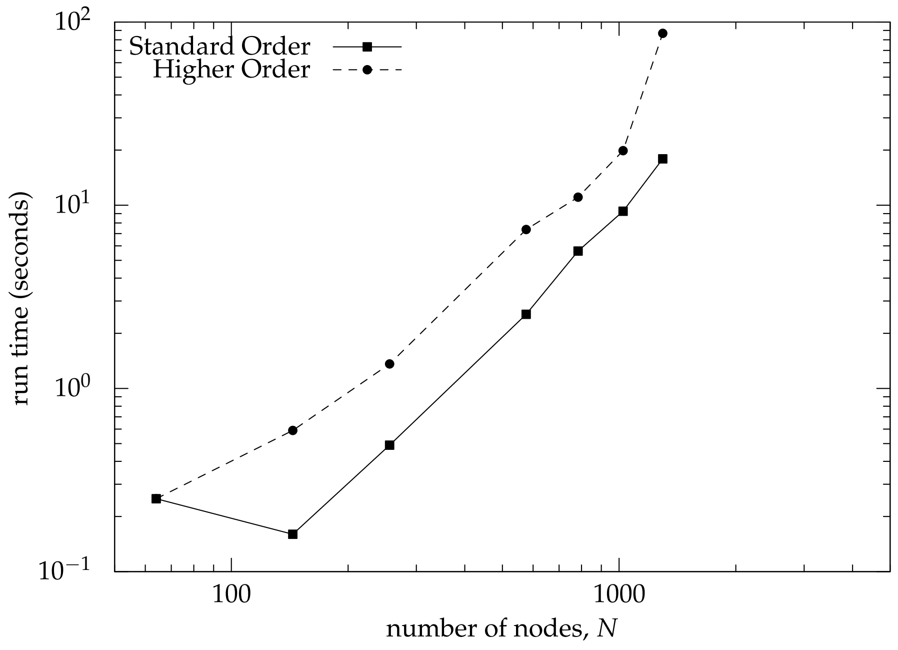


Figure 5.10: Approximate runtimes for the RBF-FD code using standard and higher order, with the same parameters used in generating the results shown in fig. 5.8.

This is given as approximate guidance only; the code that was used to generate these results was a debug version which benefited neither from compiler optimisations nor an implementation that prioritised speed of execution. Moreover, the timings were carried out on a workstation on which various other programs were running so the RBF-FD code may have been allocated less CPU time than wall-clock time by the scheduler. We see that for all numbers of nodes considered (other than 64), the higher order method takes longer than the standard order method, in some cases significantly longer.

In fig. 5.11 we show the results of varying the multiquadric shape parameter c with a fixed number of nodes $N = 3136$ and fixed stencil radius $r = 0.053625$, using uniformly distributed nodes, so that the stencil always contains 29 nodes. We observe that the relative error drops monotonically for increasing shape parameters up to a threshold $c = 0.28$ and has an oscillatory nature for large values of c, with many of the

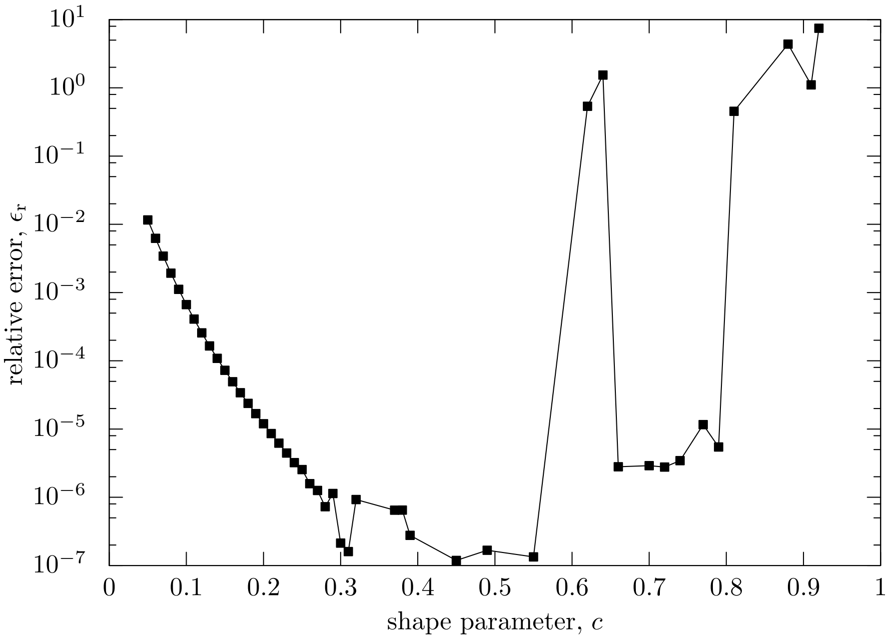


Figure 5.11: Mean relative error ϵ_r as a function of multiquadric shape parameter c using a fixed number of uniformly-distributed nodes (standard order).

larger values giving rise to very high relative errors.

Further, we examined the performance of the method using various different RBFs, as discussed in §5.3.2. In fig. 5.12 we show ϵ_r as a function of the total number of nodes N, using regularly-spaced nodes, for six different combinations of RBF and shape parameter. It can be seen that, in general, larger shape parameters give rise to better accuracy, which follows the trend illustrated in fig. 5.11 for the multiquadric RBF. However, as the number of nodes increases, errors in the Gaussian elimination solution of the linear system (5.53) became more significant, ultimately compromising accuracy for some sets of parameters (where data points are missing the eigensolver failed to converge on at least four nonzero eigenvalues). In fig. 5.13 we show the results of the same investigations using Sobol-pattern nodes. The trends are mostly similar, but we see considerably larger maximum relative errors, and for some combinations of RBF and shape parameter, increasing the resolution above a threshold value causes a considerable loss of accuracy. For the Gaussian RBF, with $N > 1000$, the larger shape

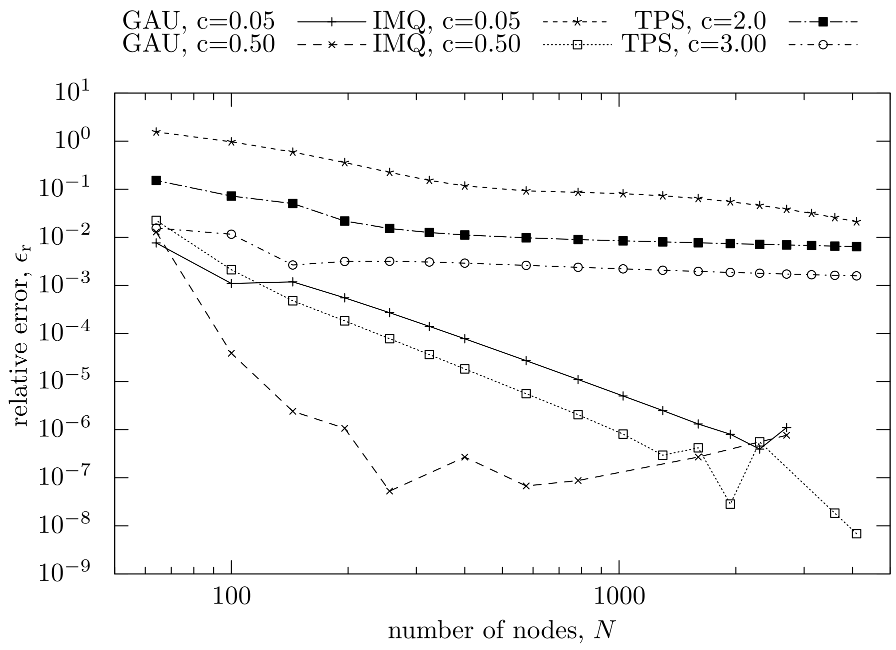


Figure 5.12: Mean relative error ϵ_r as a function of the total number of nodes N for various RBFs and shape parameters (uniformly spaced nodes, standard order).

parameter gave rise to worse accuracies than a smaller one. We note that in this case the eigensolver failed to find the requested number of the smallest eigenvalues, and speculate that the cause is likely to be an increasingly poor condition number of the eigenvalue problem. It is possible that a suitable preconditioner might improve the results in these cases, as discussed in *e.g.* [141]. In both cases, the inverse multiquadric with shape parameter $c = 0.50$ and the Gaussian with shape parameter $c = 0.05$ gave the most consistently good results.

5.5 Summary and conclusions

This chapter has introduced the field of meshless methods for partial differential equations, beginning by outlining the context in which meshless methods began to attract considerable research interest. A brief introduction was provided to the traditional, mesh-based methods including the finite element method (FEM), finite differ-

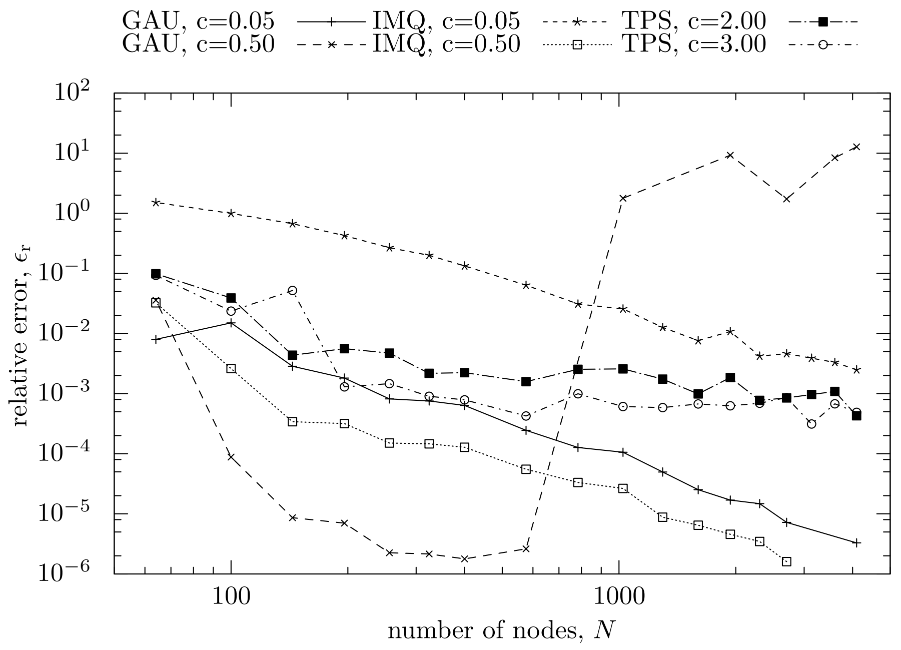


Figure 5.13: Mean relative error ϵ_r as a function of the total number of nodes N for various RBFs and shape parameters (Sobol nodes, standard order).

ence methods (FDM), and finite volume method (FVM). Each of these methods inevitably has certain drawbacks, and in particular, some drawbacks stem intrinsically from the mesh-based nature of the methods. Among them are an inability to simulate material disintegration in mechanics applications (except along paths embedded in the mesh of the original problem), and the requirement for a very fine mesh to capture local character imposing a high computational burden through necessary re-meshing, along with the potential introduction of inaccuracies when mapping thermodynamic state variables between meshes.

Meshless methods were then introduced as a class of methods for solving PDEs where the problem is discretized at a number of nodes in the problem domain, without need for any special connection or relationship between nodes. Although the application of meshless methods in engineering is a relatively young field of research compared to the traditional finite element, finite difference and finite volume methods, the methods have already become established as suitable for a variety of applications including fluid dynamics, crack growth and propagation, and heat flow problems.

The radial basis functions (RBFs) that many meshless methods are based upon were detailed, starting with the definition of an RBF and then giving examples of popular globally- and compactly- supported RBFs seen in the literature. The classification of meshless methods was considered, particularly a classification in terms of whether the methods are based on weak or strong forms, or a hybrid of both.

The chapter concluded by giving the formulation of a novel RBF-FD meshless method for solving an eigenvalue problem on a periodic domain. The system matrix is made sparse and well-conditioned in our method by employing a stencil or influence domain whose radius is tunable, and the method formulated for our sample problem is capable of good accuracy. The chapter also showed how ideas from Hermite interpolation may be incorporated into the RBF-FD method to increase the accuracy available with a fixed size of RBF-FD stencil. Results from our implementations of the RBF-FD and higher order RBF-FD schemes indicate that these are promising methods for solving eigenvalue problems on a periodic domain, and confirm that the higher order scheme delivers enhanced accuracy without increasing the template size.

The material in this chapter forms the background to the novel meshless methods formulated and reported on in §6 for the specific application of modelling the bandgap characteristics of two-dimensional photonic crystals.

Chapter 6

Meshless methods for photonic crystal modelling

This chapter begins with a brief overview of the field of photonic crystals. The subject area is first introduced in §6.1, then the mathematical and physical background to modelling 2D photonic crystals is summarised in §6.2. Recently, other work has shown that meshless methods can successfully be used in photonic crystal modelling. These developments are summarised in §6.3. Following this, we draw upon the background material presented in §5 to introduce a new meshless local weak-strong form method for photonic crystal modelling in §6.4, which builds upon the existing methods. We then go on to formulate a new radial basis function finite difference method, based on that in §5.4 to solve the transverse magnetic mode problem in photonic crystal modelling in §6.5. The chapter ends with a summary and conclusions.

The work presented in §6.4.3 was carried out in collaboration with others. Specifically, the cloud worker infrastructure responsible for enqueueing parameter sets, invoking the meshless method using those parameters, and recording the results was provided by Steven Johnston, who also created the diagram that fig. 6.9 is based upon. However, I developed the meshless method itself.

6.1 Introduction to photonic crystals

'Photonic crystal' is a term that describes a periodic dielectric structure that prevents the propagation of specific wavelengths of electromagnetic radiation. Considerable interest in such structures arose after two important publications in 1987 [142, 143], although Lord Rayleigh had analysed a 1D case in 1887 [144]. The photonic band gap structure present in photonic crystals is analogous to the electron energy bands and band gaps in semiconductor physics. In nature, photonic band gap structures create splendid optical effects, including the iridescent blue in the wings of the Morpho butterfly.

Typical photonic crystals fabricated in laboratories may consist of silicon pillars arranged in a lattice, the pillars being surrounded by air. Photonic crystals are an interesting area of research because of their wide field of potential applications in filtering, focusing and dispersing light. Suitably designed photonic crystals could profitably be applied, for example, to lasers, telecommunications, other light sources, and to optical electronics or computing.

Because fabricating photonic crystals can be a tedious and expensive process [145], it is desirable to be able to accurately model the crystals and check that they behave as intended, before committing to manufacture them. With an increasingly complex array of shapes and lattices being considered for photonic crystals (*e.g.* [146]), it is appropriate to investigate novel methods of modelling their behaviour. It was shown by Hart *et al.* [147] that meshless methods are a viable means of modelling photonic crystals.

6.2 Mathematical and physical background

In this section, we detail specific aspects of photonic crystals and how they are modelled. We begin by recounting some ideas from crystallography in §6.2.1 before considering the Maxwell equations in §6.2.2, and examining harmonic solutions to the equations to the equations in §6.2.3. With those solutions in hand, we then illustrate how a photonic band gap may be given rise to in §6.2.4, and briefly note the convention for expressing the size of such band gaps in §6.2.5. Following this, in §6.2.6, we move from the simplest, 1D example system to a 2D photonic crystal of the kind that we shall model in later sections.

6.2.1 Prerequisite crystallography

Since photonic crystals are part of the broader class of crystals, many ideas from crystallography and solid state physics are called into requisition in the description and analysis of photonic crystals. Here we outline some of the most important of these concepts. Considerably more detailed background is given in various textbooks, including, *e.g.* [148, 149].

Lattices The 'essence of crystallinity' is said to be the regular geometrical arrangement of atoms in space [148]. These arrangements of atoms are described by reference to perfect, infinite arrays of points in space, which are termed lattices. In a lattice, the points are so arranged that every point has identical surroundings. Positions on a lattice relative to an arbitrarily chosen origin are traditionally expressed as combinations of lattice vectors. For example, on a 2D lattice with basis vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, a position vector is $\boldsymbol{R} = n_1\boldsymbol{a} + n_2\boldsymbol{b}$ where n_1 and n_2 are arbitrary integers. In the case of the photonic crystals discussed in this chapter, their discrete translational symmetry is such that the structure repeats itself on each lattice point, so that for some physical entity $\phi(\boldsymbol{r})$ we have $\phi(\boldsymbol{r}) = \phi(\boldsymbol{r} + \boldsymbol{R})$.

Lattice cells For any lattice, each point has some given volume associated with it, which is the primitive cell. The primitive cell may be formed either with a vertex at the lattice point, or with the lattice point at its centre. In the latter case it is known as the Wigner-Seitz cell. For the two dimensional case, the Wigner-Seitz cell is the area enclosed by the perpendicular bisectors of the lines linking a given lattice point to its nearest neighbours.

The reciprocal lattice Considering a 2D case, a lattice with basis vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ can be seen as an assembly of identical lines; these lines may be chosen in one of an infinity of ways. To describe the choice of lines in use, Miller indices are employed. Within the unit cell, the intercepts x and y of the chosen lines with the coordinate axes are noted. These are then expressed in terms of the basis vectors, as $x/\|\boldsymbol{a}\|$ and $y/\|\boldsymbol{b}\|$. Taking the inverse and expressing the resulting ratios as the lowest pair of integers hk gives the 2D Miller index for the line. If we characterise the family of lines (hk) by the unit normal $\boldsymbol{n}_{hk}$ and the inter-line spacing d_{hk}, we can define a reciprocal lattice in terms of the vectors $\boldsymbol{G}_{hk} = 2\pi\boldsymbol{n}_{hk}/d_{hk}$. If we take $\boldsymbol{G}_{hk} = h\boldsymbol{A} + k\boldsymbol{B}$ then the directions of $\boldsymbol{A}$ and $\boldsymbol{B}$ are the directions perpendicular to $\boldsymbol{b}$ and $\boldsymbol{a}$ respectively. The magnitudes of $\boldsymbol{A}$ and $\boldsymbol{B}$ are the reciprocals of the spacing between lines in the real lattice in the directions of $\boldsymbol{B}$ and $\boldsymbol{A}$ respectively.

The Brillouin zones The Wigner-Seitz cell of the reciprocal lattice is known as the first Brillouin zone. This zone has a physical significance because it is the part of the reciprocal lattice space in which all the wavevectors $\boldsymbol{k}$ are unique. Where the first Brillouin zone features rotational symmetry, a smaller section, known as the irreducible Brillouin zone, can be seen to contain all unique $\boldsymbol{k}$ vectors instead. For a square lattice, the reciprocal lattice is square also. We illustrate the reciprocal lattice, the first Brillouin zone, and the irreducible Brillouin zone for this case in fig. 6.1. We label the critical points Γ, X and M, which are located around the perimeter of the irreducible Brillouin zone, at $(0, 0)$, $(\pi/a, 0)$, and $(\pi/a, \pi/a)$ respectively, where a is the lattice constant.

The Bloch-Floquet theorem Bloch showed [150] that the electrons in a conductor are scattered only by impurities, not by the ions which are aligned periodically on a lattice. Considering Schrödinger's equation and a wavefunction $\boldsymbol{\Psi}_{\boldsymbol{k}}(\boldsymbol{r})$ which solves the equation, Bloch extended a theorem first advanced by Floquet [151] to show that $\boldsymbol{\Psi}_{\boldsymbol{k}}(\boldsymbol{r})$ may be written as the product of a plane wave $\exp(\mathrm{i}\boldsymbol{k} \cdot \boldsymbol{r})$ and a spatially periodic envelope function $U_{\boldsymbol{k}}(\boldsymbol{r})$. We will see in §6.2.2 that the solutions to the Maxwell equations may

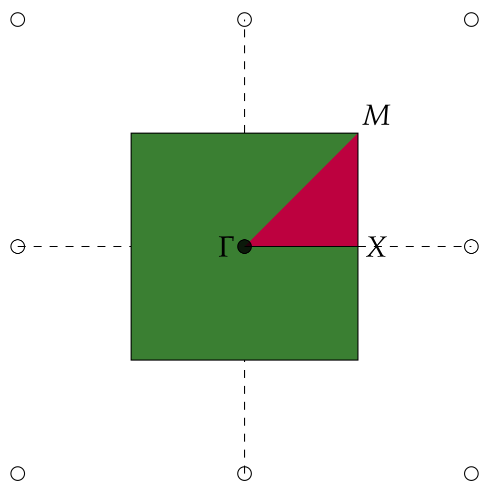


Figure 6.1: The reciprocal lattice of a square real-space lattice, illustrating the construction of the first Brillouin zone (green) around the black-shaded reciprocal lattice point, and the irreducible Brillouin zone (purple).

be written in terms of Bloch states, and due to the lattice periodicity, we must have

$$U_{\boldsymbol{k}}(\boldsymbol{r}) = U_{\boldsymbol{k}}(\boldsymbol{r} + \boldsymbol{R}) \tag{6.1}$$

for a lattice vector $\boldsymbol{R}$. Moreover, different values of $\boldsymbol{k}$ do not necessarily give different modes. For a system periodic in y only, the Bloch states with wavenumber k_y and $k_y + 2m\pi/a$ are identical from a physical point of view, and thus $\omega(k_y) = \omega(k_y + 2m\pi/a)$. The only set of non-redundant values of the wavenumber that we need to consider are those in the first Brillouin zone; in this case, $-\pi/a < k_y \leq \pi/a$.

6.2.2 The Maxwell equations

To model the effect of photonic crystals on light, it is necessary to solve Maxwell's equations. Expressed in their differential, macroscopic form, in SI units, they are:

$$\begin{aligned}
\nabla \cdot \boldsymbol{B} &= 0, \\
\nabla \cdot \boldsymbol{D} &= \rho, \\
\nabla \times \boldsymbol{E} &= -\frac{\partial \boldsymbol{B}}{\partial t}, \\
\nabla \times \boldsymbol{H} &= \boldsymbol{J} + \frac{\partial \boldsymbol{D}}{\partial t},
\end{aligned} \tag{6.2}$$

where $\boldsymbol{E}$ and $\boldsymbol{H}$ are the macroscopic electric and magnetic fields, and $\boldsymbol{D}$ and $\boldsymbol{B}$ are the displacement and magnetic induction fields respectively. The free charge density is ρ and the current density is $\boldsymbol{J}$. Here we are interested in analysing the propagation of light through the crystal, so we can set $\rho = 0$ and $\boldsymbol{J} = 0$, which precludes any light sources (or electric currents) within the material.

In line with standard practice [152, 153], we further assume that the field strengths present are sufficiently low that we remain within the linear regime. The linear regime is that in which the components of the displacement field $\boldsymbol{D}$ are related linearly to the components of the electric field vector $\boldsymbol{E}$. At sufficiently high field strengths, in so-called nonlinear media, this assumption would be invalid and various nonlinear effects, including frequency doubling (second harmonic generation) may be observed in experiments, as described in, for example [154, §13.4] or [155, §26].

We also take the dielectric constant to be isotropic, and perfectly periodic with respect to the spatial coordinates. Thus, we can write the following constitutive equations:

$$\begin{aligned}
\boldsymbol{D}(\boldsymbol{r}) &= \varepsilon_0 \varepsilon(\boldsymbol{r}) \boldsymbol{E}(\boldsymbol{r}), \\
\boldsymbol{B}(\boldsymbol{r}) &= \mu_0 \mu(\boldsymbol{r}) \boldsymbol{H}(\boldsymbol{r}),
\end{aligned} \tag{6.3}$$

where we introduced the vacuum and relative permittivities ε_0 and $\varepsilon(\boldsymbol{r}, \omega)$, and the vacuum and relative magnetic permeabilities μ_0 and $\mu(\boldsymbol{r})$ respectively. The angular frequency of the light is ω. We ignore any material dispersion, setting $\varepsilon(\boldsymbol{r}) = \varepsilon(\boldsymbol{r}, \omega)|_{\omega=\omega_0}$

for the frequency of interest, ω_0. In the materials we are interested in, the magnetic permittivity is very close to unity, so we also take $\mu(\boldsymbol{r}) = 1$. A final assumption is that the material is lossless, so that $\varepsilon(\boldsymbol{r})$ is always real and positive.

Under these assumptions, the Maxwell equations (6.2) become the following:

$$\begin{aligned}
\nabla \cdot \boldsymbol{H}(\boldsymbol{r}, t) &= 0, \\
\nabla \cdot [\varepsilon(\boldsymbol{r})\boldsymbol{E}(\boldsymbol{r}, t)] &= 0, \\
\nabla \times \boldsymbol{E}(\boldsymbol{r}, t) &= -\mu_0 \frac{\partial \boldsymbol{H}(\boldsymbol{r}, t)}{\partial t}, \\
\nabla \times \boldsymbol{H}(\boldsymbol{r}, t) &= \varepsilon_0 \varepsilon(\boldsymbol{r}) \frac{\partial \boldsymbol{E}(\boldsymbol{r}, t)}{\partial t}.
\end{aligned} \tag{6.4}$$

6.2.3 Harmonic solutions and the eigenvalue equation

We seek harmonic solutions to (6.4) of the form

$$\begin{aligned}
\boldsymbol{E}(\boldsymbol{r}, t) &= \boldsymbol{E}(\boldsymbol{r}) e^{-\mathrm{i}\omega t}, \\
\boldsymbol{H}(\boldsymbol{r}, t) &= \boldsymbol{H}(\boldsymbol{r}) e^{-\mathrm{i}\omega t},
\end{aligned} \tag{6.5}$$

where $\mathrm{i} = \sqrt{-1}$ and we have written the solutions as Bloch states, the products of plane waves and the spatial patterns or mode profiles $\boldsymbol{E}(\boldsymbol{r})$ and $\boldsymbol{H}(\boldsymbol{r})$. Fourier analysis shows us that we may build arbitrary solutions by combining harmonic solutions.

Substituting the trial solutions (6.5) into (6.4) gives the relations,

$$\begin{aligned}
\nabla \times \boldsymbol{E}(\boldsymbol{r}) - \mathrm{i}\omega\mu_0 \boldsymbol{H}(\boldsymbol{r}) &= 0, \\
\nabla \times \boldsymbol{H}(\boldsymbol{r}) + \mathrm{i}\omega\varepsilon_0\varepsilon(\boldsymbol{r})\boldsymbol{E}(\boldsymbol{r}) &= 0.
\end{aligned} \tag{6.6}$$

Eliminating $\boldsymbol{E}$ and noting that $c = 1/\sqrt{\varepsilon_0\mu_0}$ yields the following eigenvalue equation entirely in terms of $\boldsymbol{H}$:

$$\nabla \times \left(\frac{1}{\varepsilon(\boldsymbol{r})} \nabla \times \boldsymbol{H}(\boldsymbol{r}) \right) = \left(\frac{\omega}{c} \right)^2 \boldsymbol{H}(\boldsymbol{r}), \tag{6.7}$$

whose solutions must additionally conform to the divergence criteria in the first two lines of (6.4). There is a physical interpretation of these conditions, which is that the electromagnetic waves are transverse: for a wave $\boldsymbol{H}(\boldsymbol{r}) = \boldsymbol{a}\exp(\mathrm{i}\boldsymbol{k} \cdot \boldsymbol{r})$, with a wavevector $\boldsymbol{k}$, the divergence criteria would require that $\boldsymbol{a} \cdot \boldsymbol{k} = 0$. This is equivalent to saying that the medium contains no point sources or sinks of the displacement and magnetic fields.

The equation above is an eigenvalue equation, and may be re-written in terms of an operator $\boldsymbol{\Theta}$ to emphasise its nature:

$$\boldsymbol{\Theta} \boldsymbol{H}(\boldsymbol{r}) = \left(\frac{\omega}{c}\right)^2 \boldsymbol{H}(\boldsymbol{r}), \tag{6.8}$$

where we have defined

$$\boldsymbol{\Theta} = \nabla \times \left(\frac{1}{\varepsilon(\boldsymbol{r})} \nabla \times\right). \tag{6.9}$$

The operator $\boldsymbol{\Theta}$ introduced in (6.9) is a linear, Hermitian operator: it has real eigenvalues and any linear combination of solutions is itself a valid solution. The eigenvalues $(\omega/c)^2$ of (6.8) are proportional to the squares of the frequencies of the modes, whose spatial field patterns are given by the eigenvectors $\boldsymbol{H}(\boldsymbol{r})$.

6.2.4 Photonic band gaps in one-dimensional photonic crystals

Lord Rayleigh was probably first to show that a one-dimensional photonic crystal consisting of a stack of layers of alternating dielectric constants may have a band gap; that is, a range of wavelengths across which its reflectivity is large [144]. This multilayer configuration is homogeneous in two axes and periodic along the other; it is the simplest photonic crystal. Rayleigh considered light travelling along the z-direction, normal to the layers. He used the technique of considering the sums of multiple reflections and refractions which occur at the interfaces between the materials as a plane wave travels through the structure to analyse the system [156]. The one-dimensional photonic crystal is illustrated in fig. 6.2. The origin of the photonic band gap is intimately related to the periodicity of the structure, which can be shown not to support any extended electromagnetic modes (*i.e.* modes with a real wave vector) with frequencies inside a certain range. This range of frequencies is the band gap.

To understand the origin of the band gap, we follow the approach set out in [152] and consider waves propagating in the z-direction so that the only important compon-

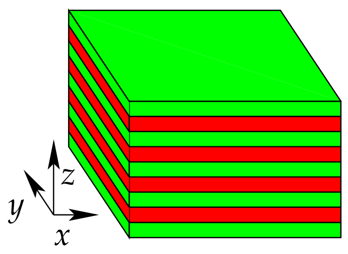


Figure 6.2: The one-dimensional photonic crystal. Rayleigh treated light travelling in the z-direction, normal to the layers. The two colours represent materials with different dielectrics. After fig. 1 in [152, §4].

ent of the wavenumber is $k = \boldsymbol{k}_z$. In a bulk medium, which is not periodic, we may assign an arbitrary periodicity of a. The dispersion relation for such a material is given by the light line

$$\omega(k) = \frac{ck}{\sqrt{\varepsilon}}, \tag{6.10}$$

because the speed of light is reduced by the index of refraction $n = \sqrt{\varepsilon}$. We specified periodic boundary conditions, and require that k repeat itself outside of the first Brillouin zone, so that the light line must fold back into the Brillouin zone when it reaches an edge. There is no photonic band gap in this material, as shown in fig. 6.3 in dashed lines. If, however, we consider a material consisting of periodically stacked layers with alternating dielectric constants, the dispersion relation changes somewhat at the edge of the Brillouin zone; a gap emerges between the upper and lower branches of the lines, as shown with solid lines in fig. 6.3. This is the photonic band gap, which contains no modes with real wave vectors.

The first band gap arises for $k = \pi/a$, which is a mode with wavelength $2a$, where a is the periodicity of the crystal. This mode can be spatially localised in the crystal in either of two configurations: the nodes of the standing wave can be centred in the crystal layers with high dielectric, or the layers with low dielectric. From the variational theorem in electromagnetics, we know that modes with lower frequencies have more energy in regions of high dielectric, whilst the opposite holds for higher frequency modes. Thus, at the edge of the Brillouin zone, the mode which has most energy in

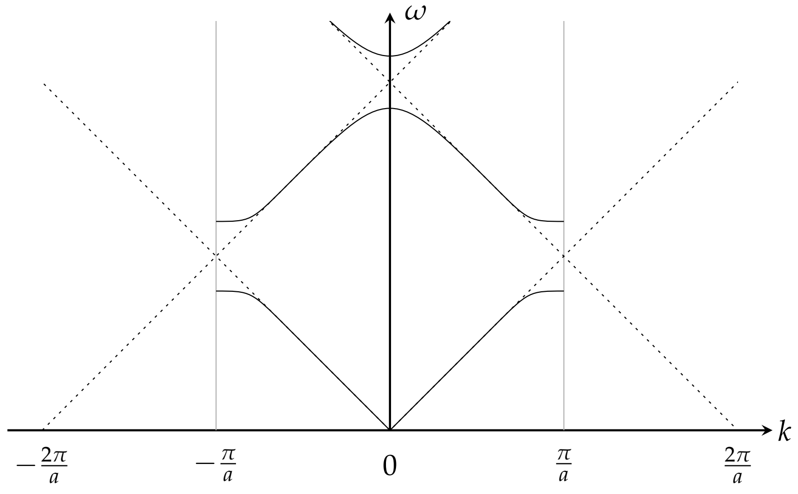


Figure 6.3: Schematic dispersion relation for a one-dimensional photonic crystal for light travelling normal to the layers. Dashed lines represent the case for a homogeneous material; solid lines for a periodically layered material with layer thickness $0.5a$. The vertical lines illustrate the boundary of the first Brillouin zone. Adapted from fig. 2 in [152, §4].

the high-dielectric region lies just below the gap, which gives it a lower frequency. The mode with most energy in the low-dielectric region has a higher frequency and sits just above the gap.

It is noted, however, that modes with complex wave vectors $k + \mathrm{i}\kappa$ may exist within the band gap. Such evanescent states decay exponentially into the crystal, on a length scale of $1/\kappa$, as shown:

$$\boldsymbol{H}(\boldsymbol{r}) = e^{ikz}\boldsymbol{u}(z)e^{-\kappa z}. \tag{6.11}$$

In a perfect, infinite photonic crystal, these modes cannot be excited since they are divergent as $z \to \pm\infty$ (dependent upon the sign of κ), and would require infinite energy. In real crystals, imperfections may halt the exponential growth and these modes may be energised.

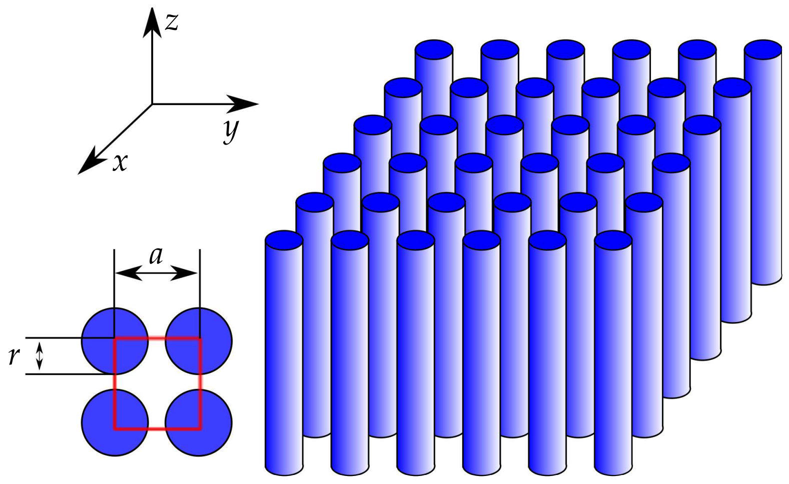


Figure 6.4: The two-dimensional photonic crystal. We treat light travelling in the $\boldsymbol{xy}$ plane. The left hand inset shows a view of the lattice from above, defining the radius r of the rods, the lattice constant a, and the unit cell (in red). After fig. 1 in [152, §5], with thanks to Richard Boardman for assistance in preparing this figure.

6.2.5 Scale invariance in the Maxwell equations

Because the results of the Maxwell equations are scalable, attempts to directly measure the frequency range of the band gap are meaningless for any comparison purposes, since a photonic crystal with a band gap whose width is $\Delta\omega$ may be expanded in its physical dimensions by some factor x, after which its band gap width will become $\Delta\omega/x$. A more useful and conventional approach is to express the band gap in terms of the gap-midgap ratio, $\Delta\omega/\omega_{\mathrm{m}}$ where ω_{m} is the frequency at the middle of the band gap.

6.2.6 Two-dimensional photonic crystals

A two-dimensional photonic crystal is homogeneous along one axis, which we take as the $\boldsymbol{z}$ axis, and periodic along the other two, $\boldsymbol{x}$ and $\boldsymbol{y}$. One such crystal structure, a square lattice of cylindrical dielectric columns of radius r and lattice constant a, is illustrated in fig. 6.4.

Because of the periodicity, for a 2D photonic crystal, the dielectric function obeys the relationship $\varepsilon(\boldsymbol{r}) = \varepsilon(\boldsymbol{r} + \boldsymbol{R})$ for all lattice vectors $\boldsymbol{R}$.

If we restrict our analysis to modes that have $\boldsymbol{k}_z = 0$, i.e., those that propagate strictly in the xy plane, there is a mirror symmetry in the z direction. This means that we may define the TM (transverse magnetic) polarisation to be the one in which the magnetic field is confined to the xy plane, having no component in the z direction, and the electric field has no component along an axis other than z; and the TE (transverse electric) mode is the one in which the electric field is confined to the xy plane, with no z component, and the magnetic field has no components other than along the z direction. The symmetry allows the splitting of (6.8) into two equations, one for each polarisation. The TE and TM modes can have markedly different band structures, in some cases there are band gaps in for one polarisation but not the other. For a band gap to be considered complete, it must be the coincidence of gaps in the TE and TM modes.

A band diagram is shown in fig. 6.5 for a 2D photonic crystal consisting of a triangular lattice of holes ($\varepsilon = 1$) in a substrate ($\varepsilon = 12$). It features a complete band gap, shown in yellow.

6.3 Meshless methods for photonic crystal modelling: background

In this section we begin by reviewing some existing meshless methods that have been applied to photonic crystal modelling and bandgap calculations, these being the meshless local strong- and weak-form methods developed by Hart *et al.*, for producing band diagrams for 2D photonic crystals [147]. This section is a prelude to §6.4 and §6.5, in which we will develop meshless methods for calculating the band diagrams of 2D photonic crystals by a meshless local weak-strong form method and and RBF-FD method respectively.

The eigenvalue problem (6.8) can be split up into two equations, one for each possible polarisation state, as discussed in §6.2.6. The transverse magnetic (TM) and trans-

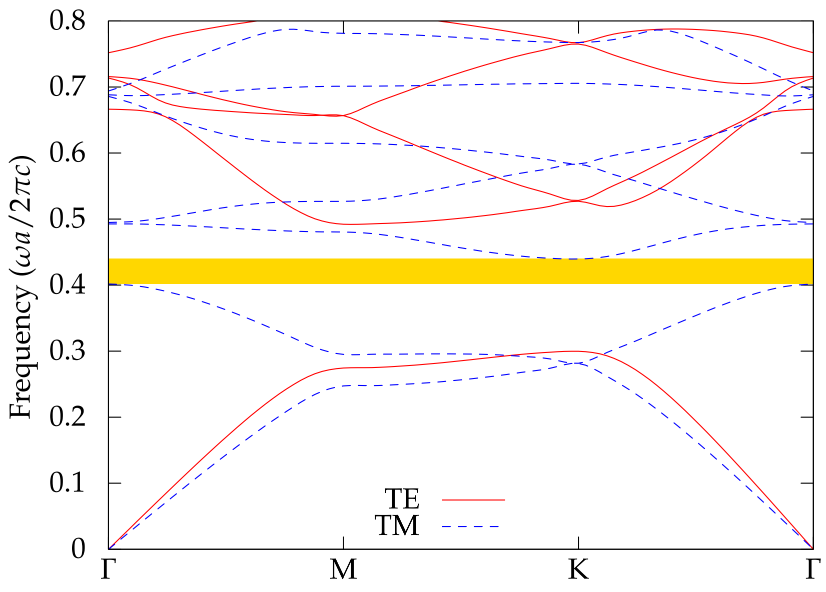


Figure 6.5: Band diagram of 2D photonic crystal with a triangular lattice of holes ($\varepsilon = 1$) in a substrate ($\varepsilon = 12$). The structure has a complete band gap, shown in yellow (calculations performed using the freely available MPB software package [141]).

verse electric (TE) mode equations are:

$$-\frac{1}{\varepsilon}\Delta\varphi = \lambda\varphi, \tag{6.12}$$

$$-\nabla \cdot \frac{1}{\varepsilon}\nabla\varphi = \lambda\varphi, \tag{6.13}$$

respectively, where, in two dimensions,

$$\Delta = \begin{pmatrix} \frac{\partial^2}{\partial x^2} \\ \frac{\partial^2}{\partial y^2} \end{pmatrix}.$$

The photonic crystal is modelled as being of infinite extent with periodic boundary conditions on the unit cell, under which conditions the wave function can be expressed as:

$$\varphi = \exp(\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x})u(\boldsymbol{x}), \tag{6.14}$$

with wave vector $\boldsymbol{k} = (k_1, k_2)$ and $\boldsymbol{x} = (x, y)$.

6.3.1 The plane-wave expansion method as a reference for comparison

In order to compute a suitable comparison result, to assess the accuracy of the meshless methods developed here, we followed Hart *et al.* [147] in utilising an established, optimised, plane-wave expansion method (PWEM), implemented as the freely available MIT Photonic Bands (MPB) software [141], which computes the definite-frequency eigenstates of Maxwell's equations in arbitrary periodic dielectric structures, using preconditioned block-iterative eigensolvers in a planewave basis. The algorithm was demonstrated by its authors to have convergence of order $\mathcal{O}(h^2)$ for a mesh spacing of h [141], and although the accuracy of the results inevitably varies with the number of sampling values employed, MPB was found to have a relative error of the lowest eigenvalue at the M-point of better than 10^{-4} for approximately 10^5 sampling points [157, §16.3]. The MPB program has been used to calculate some of the band diagrams found in [152].

In the present work, the plane wave expansion method was run with a resolution of approximately 4×10^6 sampling points, to generate the reference results for a sample geometry of photonic crystal, against which the meshless methods will be compared. This sample geometry consists of a square unit cell, of side length $a = 1$, containing a round rod of radius $r = 0.2a$, with a dielectric $\varepsilon_{\text{rod}} = 8.9$, in air ($\varepsilon_{\text{air}} = 1$). Where run times for the PWEM are given in later sections of this work, it should be noted that although MPB includes support for parallelisation via the MPI standard and libraries, we used a single-threaded MPB.

6.3.2 Meshless local strong form method

For the TM mode, Hart *et al.* solved (6.12) by a meshless local-strong form method (MLSFM) (see §5.3.4) employing compactly supported radial basis functions [147]. They used selected compactly supported RBFs from Wendland [119] and Wu [118], specifically, the $C2$ and $C4$ functions of each of those families.

The average relative error for a band diagram for a set of rods of radius $r = 0.2a$ in air was given as 1%, and the results of the method were shown to be in good agreement with the results from the PWEM. The MLSFM was used with uniformly spaced nodes and was found to have a convergence rate of $\mathcal{O}(h^2)$ where h is the distance between nearest-neighbour nodes [147].

6.3.3 Meshless local weak form method

Because of the differential operator acting upon $1/\varepsilon(\boldsymbol{x})$ in (6.13), a strong-form collocation method such as the one formulated for the TM mode is unsuitable here. At the boundary of the rod and surround, there is a discontinuity in $\varepsilon(\boldsymbol{x})$ which would cause the derivative to diverge. For this reason, Hart *et al.* developed a meshless local weak form method (MLWFM) (see §5.3.4) using Galerkin's method with CSRBFs [147]. Galerkin's method is naturally suited to handling discontinuous media, and uses an integral to calculate a discretized approximation to the true solution of the problem. Hart *et al.* developed an MLWFM for both the TM and TE polarisations, but noted that the increased computation burden of the MLWFM means that the MLSFM is preferred for the TM polarisation.

The average relative error for a band diagram for a set of rods of radius $r = 0.2a$ in air was stated as 1%, and the results of the method were showed to be in good agreement with the results from the established PWEM. The method was implemented with nodes uniformly spaced, and found to have a convergence rate of $\mathcal{O}(h^4)$ where h is the spacing between nearest-neighbour nodes [147].

6.3.4 Other meshless methods for photonic crystal modelling

Additional meshless methods for photonic crystal band structure calculations have been proposed. The moving least squares (MLS) method was proposed by Jun, Cho and Im [158]. They presented results initially for the electromagnetic Kronig-Penney problem, for which analytical solutions are possible. The agreement between the analytic solutions and the MLS results was very good. They went on to furnish results for the TE and TM modes in a 2D photonic crystal, compared against results from a plane wave expansion method (PWEM). They observed that the agreement is again good, but that the MLS converges more rapidly than the PWEM, making the method very attractive.

The meshless local Petrov-Galerkin (MLPG) method was employed by Nicomedes *et al.* in 2012 [114]. Their approach is based on a form of the MLPG known as the local boundary integral equation method (LBIE), with the periodicity enforced through the shape functions rather than imposed upon the unit cell. They achieved results for the TM polarisation which are in good agreement with other numerical studies, but did not treat the TE polarisation.

Additionally, recent work has shown that meshless methods may be applied to solve vectorial mode fields in microstructured optical waveguides [116, 117]. In this work, a finite cloud method was applied to solve for both "transverse components of the magnetic field as well as the effective index of refraction for the waveguides." The finite cloud method was first presented in 2001 [115] as a truly meshless technique for solving partial differential equations.

6.4 A meshless local weak-strong form method for 2D bandgap calculations

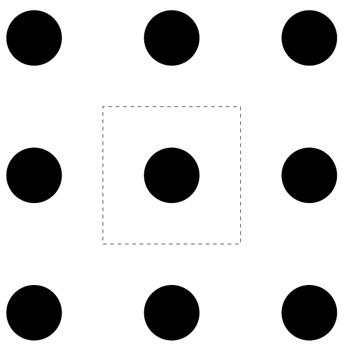
Building on the work of Hart *et al.* [147], summarised in §6.3, we propose a new meshless method to solve the Maxwell equations with periodic boundary conditions for photonic crystal modelling. The new method is a hybrid meshless local weak-strong form method (MLWSFM) using compactly supported radial basis functions, and we formulate the method for both TE and TM polarisations. It retains the generic advantages of meshless methods whilst improving on the MLWFM by reducing the amount of computationally-expensive numerical integration required. We applied a cloud-based architecture for performing distributed parameter sweeps during the development and verification of this algorithm, which conferred a number of advantages as discussed in the general case in §2.4. We discuss the application of cloud computing to the development of this algorithm in §6.4.3.

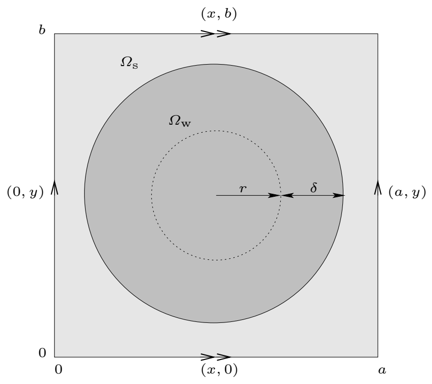
6.4.1 Meshless method formulation

The MLWSFM was formulated with two subdomains, as illustrated in fig. 6.6(b). The figure shows the subdomains, namely Ω_w and Ω_s. For nodes i with $\boldsymbol{x}_i \in \Omega_\mathrm{w}$, the weak form method is used; where $\boldsymbol{x}_i \in \Omega_\mathrm{s}$, the strong form method is used. Note that, should any node fall upon the border, we define it to be within Ω_s. Moreover, the overall domain of the system $\Omega = \Omega_\mathrm{s} \cup \Omega_\mathrm{w}$ is made periodic by imposing the conditions $u(x,0) = u(x,b)$ and $u(0,y) = u(a,y)$, where a and b are the lengths of the edges of the domain. In the next paragraphs, we formulate meshless local strong- and weak-form methods for the TE and TM modes, which will be the foundations of the MLWSFM.

TM mode, strong form We begin by formulating a local strong form method for TM mode. Substituting (6.14) into (6.12), we have:

$$-(\nabla + i\boldsymbol{k}) \cdot (\nabla + i\boldsymbol{k})u = \varepsilon(\boldsymbol{x})\lambda u, \tag{6.15}$$


(a) Part of the photonic crystal modelled



(b) The subdomains used

Figure 6.6: The physical system modelled and the subdomains used. (a) is a section through the square array of dielectric rods (with radius $r = 0.2a$). The dashed box indicates the computational unit cell. (b) shows the subdomains within the unit cell (the unit cell is represented by the dashed line in (a)). Ω_{w} is of radius $r + \delta$, and is centred on the rod (shown with a dotted line). Ω_{s} is the subdomain outside of Ω_{w}.

where λ is the spectral parameter and $\varepsilon(\boldsymbol{x})$ is the dielectric constant (see §6.2.2). Substituting

$$u(\boldsymbol{x}) = \sum_{j=1}^{n} \gamma_j \phi_j(\boldsymbol{x}), \tag{6.16}$$

where $\phi(\|\cdot\|)$ is a CSRBF, $\phi_j(\boldsymbol{x}) = \phi(\|\boldsymbol{x}-\boldsymbol{x}_j\|)$ and $n = n_\mathrm{s} + n_\mathrm{w}$ is the total number of nodes of the strong and weak form domains, with n_s the number of nodes in Ω_s and n_w the number of nodes in Ω_w, we can formulate the following matrices **A** and **B**, of order $n_\mathrm{s} \times n$, which we will later assemble into a generalised eigenvalue problem:

$$\mathbf{A}_{ij} = -(\nabla + i\boldsymbol{k}) \cdot (\nabla + i\boldsymbol{k})\phi_j(\boldsymbol{x}_i), \tag{6.17}$$

$$\mathbf{B}_{ij} = \varepsilon(\boldsymbol{x})\phi_j(\boldsymbol{x}_i), \tag{6.18}$$

for $j = 1, 2, \ldots, n$ and nodes i with $\boldsymbol{x}_i \in \Omega_\mathrm{s}$.

TM mode, weak form We formulate a local weak form method for TM by applying Galerkin's method to (6.15), giving the following integral:

$$\int (\nabla + i\boldsymbol{k})u \cdot \overline{(\nabla + i\boldsymbol{k})v}\mathrm{d}\boldsymbol{x} = \lambda \int \varepsilon(\boldsymbol{x})u\bar{v}\mathrm{d}\boldsymbol{x}. \tag{6.19}$$

Representing

$$u(\boldsymbol{x}) = \sum_{j=1}^{n} \gamma_j \phi_j(\boldsymbol{x}), \tag{6.20}$$

and

$$v(\boldsymbol{x}) = \phi_i(\boldsymbol{x}), \tag{6.21}$$

we can formulate matrices **C** and **D**, of order $n_\mathrm{w} \times n$, which will also be assembled into the generalised eigenvalue problem. They are given by:

$$\mathbf{C}_{ij} = \int (\nabla + i\boldsymbol{k})\phi_j(\boldsymbol{x}_i) \cdot \overline{(\nabla + i\boldsymbol{k})\phi_i(\boldsymbol{x}_j)}\mathrm{d}\boldsymbol{x}, \tag{6.22}$$

$$\mathbf{D}_{ij} = \int \varepsilon(\boldsymbol{x})\phi_j(\boldsymbol{x}_i) \cdot \phi_i(\boldsymbol{x}_j)\mathrm{d}\boldsymbol{x}. \tag{6.23}$$

for $j = 1, 2, \ldots, n$, and nodes i with $\boldsymbol{x}_i \in \Omega_\mathrm{w}$.

The strong form matrices **A** and **B** can be transformed into global matrices $\mathbf{A}^\mathrm{g}$ and $\mathbf{B}^\mathrm{g}$ of order $n \times n$ by adding rows of zeros to the local matrices for the k^{th} row, *i.e.* for nodes $\boldsymbol{x}_k \in \Omega_\text{w}$. Likewise, the weak form matrices $\mathbf{C}$ and $\mathbf{D}$ may be transformed into global matrices $\mathbf{C}^\text{g}$ and $\mathbf{D}^\text{g}$ by adding rows of zeros to the local matrices for the k^th row, *i.e.* for nodes $\boldsymbol{x}_k \in \Omega_\text{s}$.

TM mode, weak-strong form hybrid Stacking the nodal matrices for the strong and weak forms together row-by-row (for $i = 1, 2, \ldots, n$) to form the global matrices, we obtain the generalised eigenvalue problem:

$$\mathbf{E}(\boldsymbol{k})\boldsymbol{\alpha} = \lambda \mathbf{F}\boldsymbol{\alpha} \tag{6.24}$$

where $\boldsymbol{\alpha}$ are the eigenvectors of order n corresponding to the nodal field values of the modes of propagation that the photonic crystal allows, λ are eigenvalues corresponding to the frequencies of the mode, and $\mathbf{E}(\boldsymbol{k})$ and $\mathbf{F}$ are matrices of order $n \times n$, defined for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$, given by:

$$\mathbf{E}_{ij} = \begin{cases} \mathbf{A}_{ij} & \text{if } \boldsymbol{x}_i \in \Omega_\text{s}, j = 1, 2, \ldots, n, \\ \mathbf{C}_{ij} & \text{if } \boldsymbol{x}_i \in \Omega_\text{w}, j = 1, 2, \ldots, n, \end{cases} \tag{6.25}$$

and

$$\mathbf{F}_{ij} = \begin{cases} \mathbf{B}_{ij} & \text{if } \boldsymbol{x}_i \in \Omega_\text{s}, j = 1, 2, \ldots, n, \\ \mathbf{D}_{ij} & \text{if } \boldsymbol{x}_i \in \Omega_\text{w}, j = 1, 2, \ldots, n. \end{cases} \tag{6.26}$$

The eigenvalues of (6.24) then give the frequencies of the allowed modes at each quasi-momentum vector $\boldsymbol{k}$.

TE mode, strong form We now formulate the local strong form method for the TE polarisation, which is valid only for regions of constant $\varepsilon(\boldsymbol{x})$. Substituting (6.14) into (6.13) gives:

$$-(\nabla + i\boldsymbol{k}) \cdot \frac{1}{\varepsilon(\boldsymbol{x})} \cdot (\nabla + i\boldsymbol{k})u = \lambda u. \tag{6.27}$$

The left-hand side can be expanded to:

$$-\nabla \cdot \left(\frac{1}{\varepsilon(\boldsymbol{x})}\nabla u\right) - i\nabla \cdot \left(\frac{1}{\varepsilon(\boldsymbol{x})}\boldsymbol{k}u\right) - i\boldsymbol{k} \cdot \left(\frac{1}{\varepsilon(\boldsymbol{x})}\nabla u\right) + \boldsymbol{k} \cdot \left(\frac{1}{\varepsilon(\boldsymbol{x})}\boldsymbol{k}u\right) = \lambda u. \tag{6.28}$$

In general the dielectric constant $\varepsilon(\boldsymbol{x})$ is discontinuous in photonic crystals, as they have different dielectric constants either side of the interface between the materials. The differential operator acting on $1/\varepsilon(\boldsymbol{x})$ would lead to a divergent result. In the MLWSFM, we apply this formulation only in regions known to be of constant $\varepsilon(\boldsymbol{x})$.

Substituting

$$u(\boldsymbol{x}) = \sum_{j=1}^{n} \gamma_j \phi_j(\boldsymbol{x}), \tag{6.29}$$

where we write $\phi_j(\boldsymbol{x}) = \phi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)$ and $n = n_\mathrm{s} + n_\mathrm{w}$ is the total number of nodes of the strong and weak forms (with n_s the number of nodes in Ω_s and n_w the number of nodes in Ω_w) into (6.28) we can formulate the following $\mathbf{G}$ and $\mathbf{H}$ matrices, of order $n_\mathrm{s} \times n$, which we will later assemble into a generalised eigenvalue problem:

$$\mathbf{G}_{ij} = -(\nabla + i\boldsymbol{k}) \cdot \frac{1}{\varepsilon(\boldsymbol{x})} \cdot (\nabla + i\boldsymbol{k})\phi_j(\boldsymbol{x}_i), \tag{6.30}$$

$$\mathbf{H}_{ij} = \phi_j(\boldsymbol{x}_i). \tag{6.31}$$

for $j = 1, 2, \ldots, n$; and nodes with $\boldsymbol{x}_i \in \Omega_\mathrm{s}$.

TE mode, weak form The local weak form method for the TE polarisation is also formulated beginning with (6.27). The weak form method will be suitable for regions of continuous or discontinuous $\varepsilon(\boldsymbol{x})$ and in the MLWSFM we apply it in the vicinity of the discontinuity. Applying Galerkin's method to this equation, we get the following integral for TE:

$$\int \frac{1}{\varepsilon(\boldsymbol{x})}(\nabla + i\boldsymbol{k})u \cdot \overline{(\nabla + i\boldsymbol{k})v}\,\mathrm{d}\boldsymbol{x} = \lambda \int u\bar{v}\,\mathrm{d}\boldsymbol{x}. \tag{6.32}$$

Substituting

$$u(\boldsymbol{x}) = \sum_{j=1}^{n} \gamma_j \phi_j(\boldsymbol{x}), \tag{6.33}$$

and

$$v(\boldsymbol{x}) = \phi_i(\boldsymbol{x}) \tag{6.34}$$

into (6.32), we may formulate the following $\mathbf{L}$ and $\mathbf{M}$ matrices, of order $n_{\mathrm{w}} \times n$, which we will later assemble into a generalised eigenvalue problem:

$$\mathbf{L}_{ij} = \int \frac{1}{\varepsilon(\boldsymbol{x})} (\nabla + i\boldsymbol{k})\phi_i(\boldsymbol{x}_j) \cdot \overline{(\nabla + i\boldsymbol{k})\phi_j(\boldsymbol{x}_i)} \mathrm{d}\boldsymbol{x}, \tag{6.35}$$

$$\mathbf{M}_{ij} = \int \phi_i(\boldsymbol{x}_j)\phi_j(\boldsymbol{x}_i) \mathrm{d}\boldsymbol{x}. \tag{6.36}$$

for $i = 1, 2, \ldots, n$; and nodes with $\boldsymbol{x}_i \in \Omega_{\mathrm{w}}$.

The strong form matrices $\mathbf{G}$ and $\mathbf{H}$ can be transformed into global matrices $\mathbf{G}^{\mathrm{g}}$ and $\mathbf{H}^{\mathrm{g}}$ of order $n \times n$ by adding rows of zeros to the local matrices for the k^{th} row, *i.e.* for nodes $\boldsymbol{x}_k \in \Omega_{\mathrm{w}}$. Likewise, the weak form matrices $\mathbf{L}$ and $\mathbf{M}$ may be transformed into global matrices $\mathbf{L}^{\mathrm{g}}$ and $\mathbf{M}^{\mathrm{g}}$ by adding rows of zeros to the local matrices for the k^{th} row, *i.e.* for nodes $\boldsymbol{x}_k \in \Omega_{\mathrm{s}}$.

TE mode, weak-strong form hybrid Stacking the nodal matrices for the strong and weak forms together row-by-row (for $i = 1, 2, \ldots, n$) to form the global matrices, we obtain the generalised eigenvalue problem:

$$\mathbf{P}(\boldsymbol{k})\boldsymbol{\alpha} = \lambda \mathbf{Q}\boldsymbol{\alpha} \tag{6.37}$$

where $\boldsymbol{\alpha}$ are the eigenvectors of order n, λ are eigenvalues, interpreted as in § 6.4.1, and $\mathbf{P}(\boldsymbol{k})$ and $\mathbf{Q}$ are matrices of order $n \times n$, defined for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$, given by:

$$\mathbf{P}_{ij} = \begin{cases} \mathbf{G}_{ij} & \text{if } \boldsymbol{x}_i \in \Omega_{\mathrm{s}}, j = 1, 2, \ldots, n, \\ \mathbf{L}_{ij} & \text{if } \boldsymbol{x}_i \in \Omega_{\mathrm{w}}, j = 1, 2, \ldots, n, \end{cases} \tag{6.38}$$

and

$$\mathbf{Q}_{ij} = \begin{cases} \mathbf{H}_{ij} & \text{if } \boldsymbol{x}_i \in \Omega_{\mathrm{s}}, j = 1, 2, \ldots, n, \\ \mathbf{M}_{ij} & \text{if } \boldsymbol{x}_i \in \Omega_{\mathrm{w}}, j = 1, 2, \ldots, n. \end{cases} \tag{6.39}$$

The eigenvalues of (6.37) then give the frequencies of the allowed modes at each quasi-momentum vector $\boldsymbol{k}$.

Gaussian quadrature

In the weak (and, hence, weak-strong) formulations, integrals (such as (6.19)) arise following the application of Galerkin's method. In order to solve these numerically, Gaussian quadrature was employed. In two dimensions, this provides a method to numerically evaluate the integral of some function $f(x,y)$ over a quadrilateral where $a \leq x \leq b$ and $c \leq y \leq d$. The variables are first changed so that the integration in the new variables runs from -1 to $+1$:

$$\int_a^b \int_c^d f(x,y)\,\mathrm{d}x\mathrm{d}y = \int_{-1}^1 \int_{-1}^1 f\left(x\left(\zeta\right), y\left(\eta\right)\right) |\mathbf{J}|\,\mathrm{d}\zeta\mathrm{d}\eta, \tag{6.40}$$

where $|\mathbf{J}|$ is the Jacobian determinant. Taking the original quadrilateral to be a square with opposite corners given by (a,c) and (b,d), we have:

$$x = \left(\frac{b-a}{2}\right)\zeta + \left(\frac{b+a}{2}\right), \tag{6.41}$$

$$y = \left(\frac{d-c}{2}\right)\eta + \left(\frac{d+c}{2}\right), \tag{6.42}$$

and substituting these into (6.40) we have

$$\int_a^b \int_c^d f(x,y)\,\mathrm{d}x\mathrm{d}y = \\ \left(\frac{b-a}{2}\right)\left(\frac{d-c}{2}\right)\int_{-1}^1 \int_{-1}^1 f\left(\frac{b-a}{2}\zeta + \frac{b+a}{2}, \frac{d-c}{2}\eta + \frac{d+c}{2}\right)\mathrm{d}\zeta\mathrm{d}\eta. \tag{6.43}$$

Gaussian integration then proceeds by approximating the integral as a sum of weighted terms as follows:

$$\left(\frac{b-a}{2}\right)\left(\frac{d-c}{2}\right)\int_{-1}^1 \int_{-1}^1 f\left(\frac{b-a}{2}\zeta + \frac{b+a}{2}, \frac{d-c}{2}\eta + \frac{d+c}{2}\right)\mathrm{d}\zeta\mathrm{d}\eta \approx \\ \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} W_{ix}W_{jy} f\left(\frac{b-a}{2}\zeta_i + \frac{b+a}{2}, \frac{d-c}{2}\eta_j + \frac{d+c}{2}\right), \tag{6.44}$$

where, for the four-point rule that we applied in this work, $n_x = n_y = 2$, $W_{ix} = W_{jy} = 1$, $\zeta_1 = \eta_1 = -\sqrt{3}/3$ and $\zeta_2 = \eta_2 = \sqrt{3}/3$ [63, §25]. The four Gaussian evaluation points are (ζ_i, η_j) for $i = 1, 2$ and $j = 1, 2$. Applying this quadrature rule yields an integral with a remainder of $\mathcal{O}(h^4)$, where in this case, $2h$ is the side-length of the square domain of integration [63, §25.4.62].

In implementing the weak form method developed here, the domain of the system is filled with background cells, each of which contains four evaluation points for the integral; we ensure that there are between three and nine times as many evaluation points as nodes in the system, as recommended in [76, p. 157]. Using too few integration points would decrease the accuracy of the results, which must be balanced against the computational cost of performing quadrature at many points.

Each background cell is associated with a circular support domain, whose radius is equal to the shape parameter of the compactly supported RBF in use. For simplicity, we evaluate the weak form matrices in parts. Taking the example of the weak form matrix **C** in (6.22), we can expand its formulation in terms of "sub-matrices" as follows:

$$\mathbf{C}_{ij} = \mathbf{S}_{ij} + \mathrm{i}\left(\mathbf{P}_{ij} - \mathbf{Q}_{ij}\right) + \boldsymbol{k}^2\mathbf{T}_{ij}, \quad \text{with} \tag{6.45}$$

$$\mathbf{S}_{ij} = \int \nabla\phi_i \cdot \nabla\phi_j \,\mathrm{d}\boldsymbol{x}, \tag{6.46}$$

$$\mathbf{P}_{ij} = \int \boldsymbol{k}\phi_i \cdot \nabla\phi_j \,\mathrm{d}\boldsymbol{x}, \tag{6.47}$$

$$\mathbf{Q}_{ij} = \int \boldsymbol{k}\phi_j \cdot \nabla\phi_i \,\mathrm{d}\boldsymbol{x}, \tag{6.48}$$

$$\mathbf{T}_{ij} = \int \phi_i \cdot \phi_j \,\mathrm{d}\boldsymbol{x}. \tag{6.49}$$

To evaluate an element *e.g.* T_{ij} of the **T** matrix in (6.49) by Gaussian quadrature, we must consider contributions involving the nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ arising from all the background cells whose support domains include the nodes in question. We achieve this as follows:

Algorithm 1 Calculate T_{ij} via Gaussian quadrature

```
1: $\mathbf{T}_{ij} = 0$
2: for all background cells $p$ do
3:     if node $i$ in support domain of $p$ and node $j$ in support domain of $p$ then
4:         set $\varrho_{1,\dots,4}$ to position vectors of each Gaussian evaluation point in $p$
5:         $\mathbf{T}_{ij} = \mathbf{T}_{ij} + \phi(\|\boldsymbol{x}_i - \varrho_1\|) \cdot \phi(\|\boldsymbol{x}_j - \varrho_1\|)$
                   $+ \phi(\|\boldsymbol{x}_i - \varrho_2\|) \cdot \phi(\|\boldsymbol{x}_j - \varrho_2\|)$
                   $+ \phi(\|\boldsymbol{x}_i - \varrho_3\|) \cdot \phi(\|\boldsymbol{x}_j - \varrho_3\|)$
                   $+ \phi(\|\boldsymbol{x}_i - \varrho_4\|) \cdot \phi(\|\boldsymbol{x}_j - \varrho_4\|)$
6:     end if
7: end for
```

Once all the relevant nodes have been considered and the elements of (6.46-6.49) are filled, the **C** matrix may be assembled as per (6.45). Implementing the method in this manner lends perspicacity to the code, making its working clearer and facilitating more straightforward debugging, but at a cost of significantly increased memory footprint due to keeping all the "sub-matrices" in memory until the final matrix is assembled.

6.4.2 An illustrative example

The MLWSFM formulated in §6.4 was implemented with emphasis on convenience of debugging, rather than on optimal speed of execution. This section details two example simulation runs explicitly, illustrating how the nodes, background cells, integration points, *etc.*, come together in the method. We also compare the runtime and resource utilisation of our implementation with that of the established PWEM code (specifically, the MPB package, as detailed in §6.3.1), but we note that the PWEM code has benefited from more time for development and optimisation, and is therefore likely to be considerably more highly optimised than our MLWSFM code. Moreover, as discussed further in §6.4.4, our code uses a LAPACK eigensolver based on the QZ algorithm (specifically, for this example, we used the `zggev` routine from the Sun Performance Library [159] accelerated LAPACK implementation) that attempts to find all the eigenvalues for each generalised eigenproblem, rather than a more suitable iterative eigensolver; the times given include the runtime consumed by this routine.

In this illustrative example, we will furnish absolute runtimes. These were all recorded on the same computer, which features an Intel Core i7 CPU model 920 running at 2.67 GHz, 12 GiB RAM, with an Intel X58/ICH10R chipset.

Simulation parameters We ran the MLWSFM simulation with two sets of parameters, which are detailed fully in table 6.1. We used the smaller numbers of nodes and background cells to produce the graphical illustrations in this section; later in the section, we present run-time and memory allocation metrics related to both parameter sets.

We ran the PWEM with two different configurations, as described in table 6.2. The resolution and mesh size parameters are particular to this implementation, and are described in the MPB manual [160] as follows:

resolution "Specifies the computational grid resolution, in pixels per lattice unit... (The grid size is then the product of the lattice size and the resolution, rounded up to the next positive integer.)"

mesh size "At each grid point, the dielectric constant is averaged over a 'mesh' of points to find an effective dielectric tensor. This mesh is a grid with mesh-size points on a side."

Nodes, strong- and weak-form subdomains In fig. 6.7 we illustrate the placement of the nodes, and the areas in which (in this example) we applied the strong and weak form methods. This corresponds to illustrating which nodes fall into each of the domains Ω_s and Ω_w from fig. 6.6(b). We also show the dielectric constant assigned to each node. No nodes are shown at $x = 1$ or $y = 1$, as these are the same nodes found at $x = 0$ and $y = 0$ respectively, due to the periodic boundary conditions.

description	symbol and value	
	setup 1	setup 2
unit cell edge length	$a = 1$	$\leftarrow$
rod radius	$r = 0.2a$	$\leftarrow$
rod dielectric	$\varepsilon_{\mathrm{rod}} = 8.9$	$\leftarrow$
background dielectric	$\varepsilon_{\mathrm{bg}} = 1.0$	$\leftarrow$
weak form parameter (see fig. 6.6)	$\delta = 0.3$	$\leftarrow$
unit cell dimensions	$a = 1 \quad b = 1$	$\leftarrow$
CSRBF type	Wu's C^4	$\leftarrow$
CSRBF shape parameter	$c = 0.5$	$\leftarrow$
no. of Brillouin zone samples	16	$\leftarrow$
no. of nodes	900	225
no. of background cells	900	289
no. of nodes in Ω_{w}	697	172
no. of nodes in Ω_{s}	203	53

Table 6.1: The parameters chosen for the MLWSFM in "setup 1" and "setup 2" as used in the illustrative example.

description	symbol and value	
	high resolution	lower resolution
unit cell edge length	$a = 1$	$\leftarrow$
rod radius	$r = 0.2a$	$\leftarrow$
rod dielectric	$\varepsilon_{\mathrm{rod}} = 8.9$	$\leftarrow$
background dielectric	$\varepsilon_{\mathrm{bg}} = 1.0$	$\leftarrow$
resolution	2048	256
mesh size	128	32

Table 6.2: The parameters chosen for the PWEM in the high- and lower-resolution cases, as used in the illustrative example.

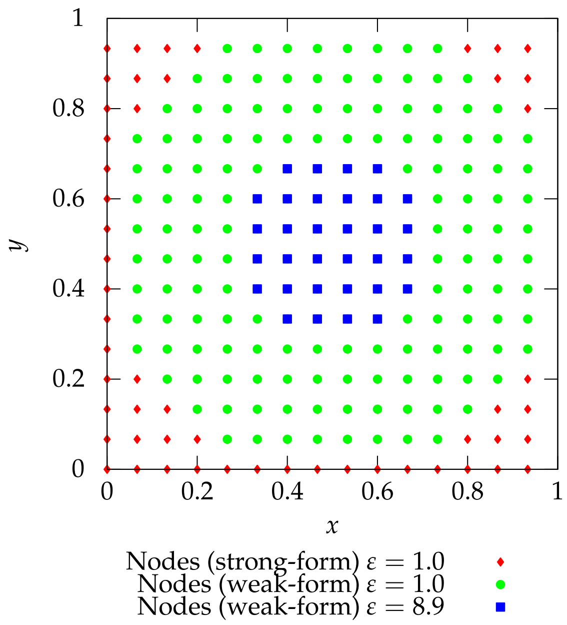


Figure 6.7: Positions of nodes (arranged on a regular grid for visual clarity) used in the example MLWSFM run with 225 nodes in total. Dielectric constants are also shown for the nodes.

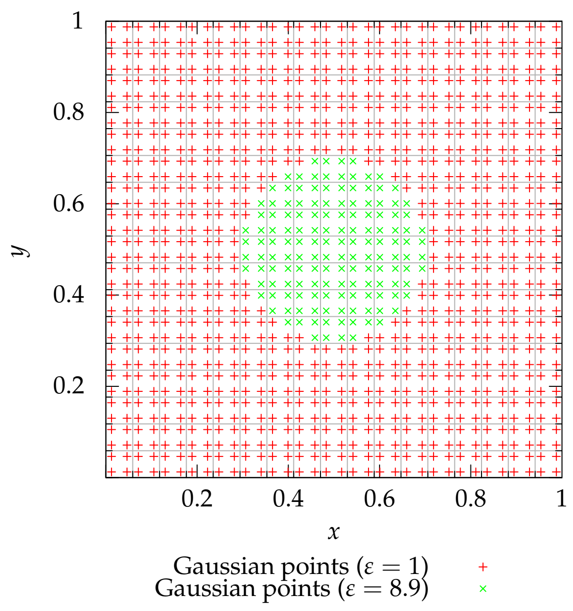


Figure 6.8: Position of Gaussian integration points and background cells (with background cells delineated by thin grey lines).

Gaussian integration points and background cells In fig. 6.8 we illustrate the placement of Gaussian integration points in background cells across the computational domain. The Gaussian points are assigned a dielectric constant based on whether they are located inside or outside of the rod to be simulated.

Resource consumption: runtime and memory requirements We analysed the resource consumptions of the MLWSFM in terms of its wall clock time and its memory requirements, and compared the results to those of the PWEM code. Our MLWSFM implementation was written in C, and (aside from several automatic variables) stores its data in memory assigned via calls to the `malloc` function. We were therefore able to estimate the total memory consumption of the code by recording the number of bytes requested in calls to `malloc`, and the results are summarised in table 6.3. In table 6.3, the entry "housekeeping data" refers to a number of arrays allocated to store, for example, the dielectrics at each node and Gaussian point, records of which of Ω_w and Ω_s each node belongs to, *etc*. The "component" matrices are those from which the eigen-

description	setup 1	setup 2
node coordinates	14.4 kB	3600 B
background cell coordinates	14.4 kB	4624 B
Gaussian point coordinates	57.6 kB	18.5 kB
housekeeping data	16.2 MB	1.1 MB
“component” matrices	252 MB	15.8 MB
eigenproblem matrices	25.9 MB	1.6 MB

Table 6.3: Memory used for various data structures used within our MLWSFM implementation.

problem may be assembled, as well as those that hold evaluations of the CSRBF and its derivatives.

For the purposes of this comparison, we take the PWEM results for the given system, when the PWEM is run at a high resolution (see table 6.2) as our reference values. We define the error of a method to be the mean of the relative errors at the lowest four eigenvalues calculated at each of 16 points around the perimeter of the irreducible Brillouin zone.

In table 6.4, we compare the memory use, runtime and mean error of the MLWSFM for each of our two example configurations, and of the PWEM running at two resolutions. Here we measure not the sizes of the various data structures stored by each algorithm in memory, but rather the maximum resident set size of the process during its lifetime, since this may be straightforwardly measured and compared for both the PWEM and the MLWSFM. Note that these memory consumption figures for the MLWSFM somewhat exceed the sums of those given in table 6.3 because the maximum resident set size is the maximum number of kilobytes of physical memory that the process used (including its data structures, code, and shared libraries). We note that if some of a process’s memory is swapped to disk, the measurement of the maximum resident set size may be unrepresentative of the maximum memory used by the process,

algorithm/setup	run time	error	memory used
MLWSFM, setup 1	456 s	0.01	291 MiB
MLWSFM, setup 2	9 s	0.02	27 MiB
PWEM, high resolution	31 392 s	0	7.3 GiB
PWEM, lower resolution	57 s	0.0002	40 MiB

Table 6.4: Comparison of the resources consumed by the MLWSFM and PWEM codes. Note the zero error for “PWEM, high resolution” is by definition, because we used these results as our reference data; it is not a claim that those results are exact.

so we verified in each case that no swapping occurred.

6.4.3 MLWSFM in the cloud

We applied Microsoft Windows Azure to the development and verification of this method, although the principles that we demonstrate are applicable across other cloud providers. Workers are the building blocks of an Azure-based solution; each consumes messages from a queue, completes the work described in the message, and outputs results to storage or to a different queue, as shown in Figure 6.9. Windows Azure workers provide a Windows operating system and basic libraries such as the .NET runtime to run user applications. The process of provisioning an Azure worker involves building a virtual machine, allocating hardware, booting the operating system and starting the user-defined application code; this is managed by the Azure fabric. In our experience it takes approximately 15-30 minutes to provision an Azure worker. At times when new revisions of the algorithm were ready for testing and characterisation it was not necessary to re-provision the worker. Instead we could halt the code, load the new revision, and start it executing, which typically took under five minutes. Upgrading workers in this manner is also managed by the Azure fabric.

Azure also provides data storage in the form of blob storage. This is highly scalable and designed to be robust. It supports key-based access, so that it is easy to provide

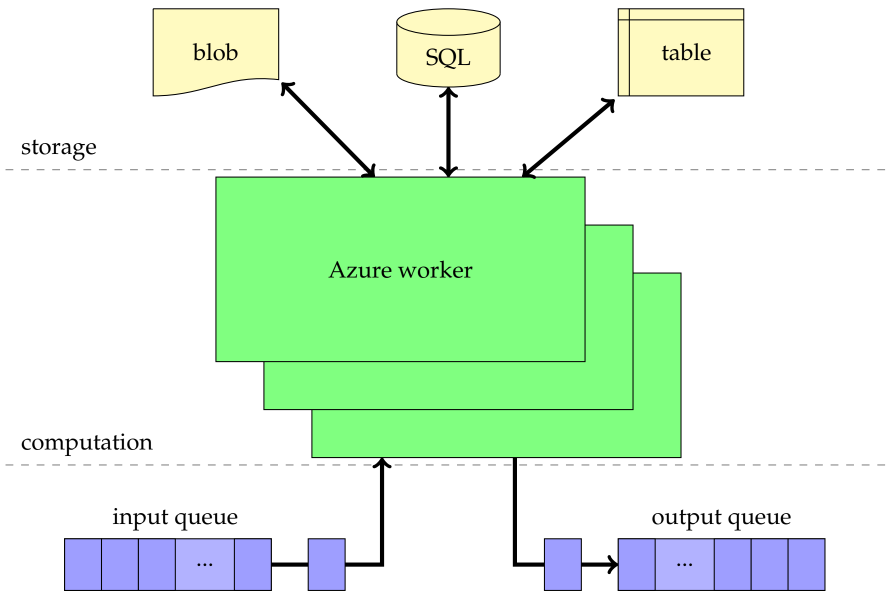


Figure 6.9: Cloud worker architecture pattern: a worker consumes jobs from a queue and writes intermediate output and final results to blob, SQL or table storage, allowing dynamic numbers of workers to process messages. Messages are also written to the output queue if required, facilitating further processing by other workers. Diagram based upon [35, Fig. 2].

multiple users with access to intermediate results stored here during the verification and tuning process, or to final results once the algorithm is known to be working correctly. In cases where the development of the algorithm, or the provision of input data sets, or analysis of output data, is a highly collaborative activity requiring input from teams around the world, this is a clear advantage for cloud-based technologies. Their high bandwidth and availability obviates the need to copy large amounts of data between institutions whilst providing large amounts of compute capability near to the data.

Our cloud-based architecture was utilised in the development cycle whenever we wanted to check that the current revision of the algorithm could produce accurate results, or assess the effect of various parameters on the accuracy and runtime of the algorithm. In order to achieve this, we ran a parameter sweep with Azure.

We set up a scalable cloud-based architecture, in which we placed each combination of parameters of interest in a message which was submitted to the input job queue. We provisioned workers which take messages off the queue and run the algorithm with the specified parameters, storing the output (the results from the algorithm, accompanied by logging and performance data) in blob storage. The software running on the workers consisted of the Windows Server 2008 R2 operating environment that Azure provides, with a custom worker process to access the job queue and call the algorithm via a command-line interface, capturing its output and writing that to blob storage. The algorithm itself was a Windows executable with supporting libraries such as a vendor-optimised LAPACK implementation. This modularity of the worker design facilitated easy updates of the algorithm when these were required. Moreover, treating each worker as an independent computational resource in this way reduces the requirement for inter-worker communication, thus helping to avoid some of the communications latencies encountered in architectures involving significant message passing between nodes [161].

The architecture pattern we used is illustrated in Figure 6.9. As shown, all the Azure workers have access to read job messages from the input queue and can write output to blob storage and, if required, an output queue to facilitate further processing by additional workers. The workers also have access to SQL database storage and could use this for logging success and failure messages as well as performance data. Our workers for all but the very highest resolution simulations were “small” instances, providing a 1.6 GHz CPU and 1.75 GB RAM (our highest resolution simulations required more RAM; to avoid excessive paging we used larger instances with 3.5 GB RAM).

In this work, we applied four workers initially, and at times supplemented those with 20 additional workers, on a separate Windows Azure account. With appropriate permission, the data in cloud storage is globally accessible by any Azure worker. In this way, it would have been possible for collaborators to provide resources financed from different budgets or institutions, all of which could independently consume messages from the same queue and write results to blob storage. The cloud based architecture provides the ability to analyse the effects of changes to the algorithm rapidly, removing the need to either wait for many hours in a cluster's job queue, or to have expensive hardware sitting idle whilst development takes place. Therefore it has the potential to significantly reduce the development cycle time. In this work, we were able to run 24 instances of the MLWSFM simultaneously, achieving significant wall-clock time savings compared to the alternatives. With more workers, the process could have been further accelerated, so that in this architecture, the overall wall-clock time for a parameter sweep is bounded below by the sum of times taken to run the longest individual simulation and to provision the worker on which it runs.

Cloud computing applied to meshless algorithm development and verification

Cloud computing assisted in several ways with the development and verification of this algorithm:

- It was necessary to validate the algorithm against results calculated by the PWEM code [141]. In this respect, Azure provided the ability to rent a large, capable machine with a lot of RAM to run a single instance of the new method, so that the simulation could be run at a high resolution to check that the results match expectations within reasonable tolerance. Initial results were available very soon after they were calculated, so that we could inspect them and verify that the process was proceeding as intended.

- To assess the impact of each of the tunable parameters of the algorithm – such as the resolution, the specific CSRBF used, and the balance of strong form to weak form nodes – we carried out a parameter sweep across a high dimensional space.

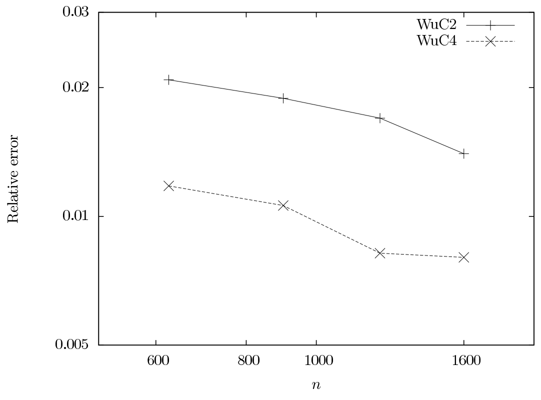
- Once the algorithm's accuracy is established there will be interest in simulating shapes whose characteristics are not already known; cloud-based architectures will facilitate this. If it is desired to simulate a large variety of shapes at once, as might be the case in a bandgap optimisation exercise [162], we could provision many worker nodes for a relatively short period of time. We note that these nodes would still perform independent calculations, so inter-node communications delays should not be a disadvantage in this application. Were it necessary to perform very high resolution calculations to verify the best-case results, it would also be possible to provision a machine with large memory for just the time taken to run the required simulations.

6.4.4 Results

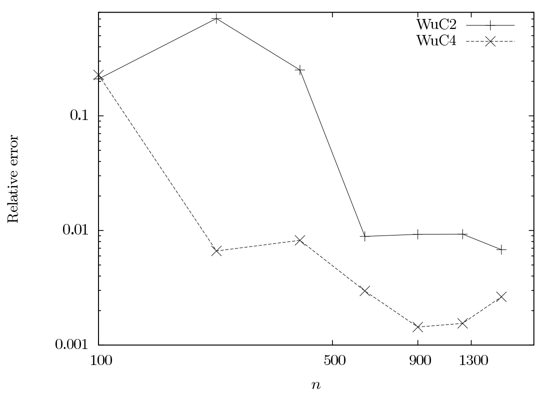
Figure 6.10 illustrates the relative errors of our MLWSFM compared to a high resolution PWEM solution of the same system (see §6.3.1), as a function of the number n of nodes used, for Wu's C^2 and C^4 CSRBFs. We calculated the relative error E for each eigenvalue in the first four bands, where $E = |E_\mathrm{p} - E_\mathrm{m}|/E_\mathrm{p}$ with E_p the high-resolution PWEM eigenvalue and E_m the meshless method eigenvalue. The relative errors we report are the averages of this error over sixteen values of $\boldsymbol{k}$ and the lowest four bands. It can be seen that the new method achieves relative errors better than 1% for several sets of parameters, for both the TE and TM modes, with an error approaching 0.1% for TM mode in some cases.

Each of the subfigures of Fig. 6.10 shows that, for any given number of nodes and background cells, the relative error is better for Wu's C4 than Wu's C2 CSRBF. Subfigures 6.10(a) and 6.10(b) are for the case where the integration proceeds over the same number of background cells as nodes. It is claimed that between 3 to 9 times more Gauss points than nodes [76] should be used; this condition is satisfied here, since we use four Gauss points per background cell. Subfigures 6.10(c) and 6.10(d) show the same data for the case where, for n nodes, there are $(\sqrt{n}+1)^2$ background cells.

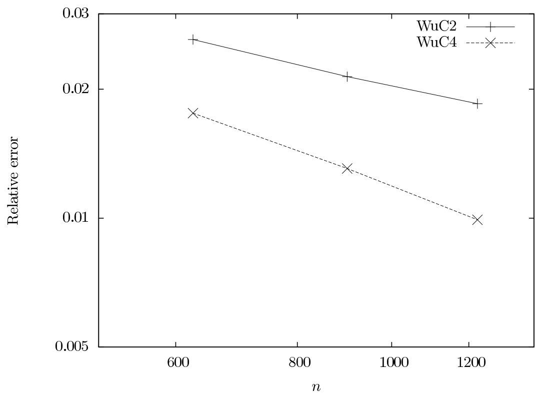
As expected, the general trend is for the error to decrease with increasing numbers of nodes. In some cases though, it reaches a minimum and then increases again. We speculate that this behaviour may be attributable to the increasingly ill-conditioned nature of the eigenvalue problem as the resolution increases [163].



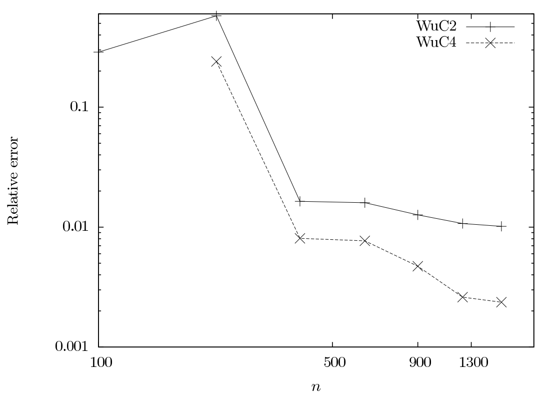
(a) TE, n nodes, n background cells



(b) TM, n nodes, n background cells



(c) TE, n nodes, $(\sqrt{n}+1)^2$ background cells



(d) TM, n nodes, $(\sqrt{n}+1)^2$ background cells

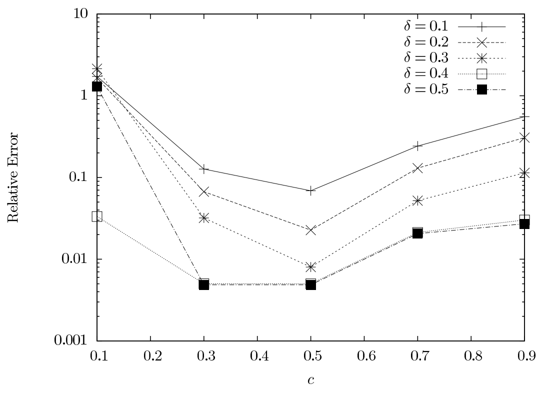
Figure 6.10: Average relative error between MLWSFM and PWEM as a function of the number of nodes n and number of background cells. Subfigures (a) and (c) are for TE and subfigures (b) and (d) are for TM. Radial basis functions used were Wu's C^2 and C^4 functions.

Figure 6.11 illustrates how the relative error of the MLWSFM solution varies with the CSRBF shape parameter c, for several values of the parameter δ, defining the fraction of nodes in the weak and strong form domains (see Fig. 6.6(b) on page 118). It is clear that in virtually all cases, $c = 0.5$ is the optimal value for accuracy, which is in agreement with previous work [112]. Moreover, it can be seen that the method's accuracy increases with increasing δ, which is also as expected, because the MLWFM has been shown to be have a convergence of order $\mathcal{O}(h^4)$ compared to the MLSFM of $\mathcal{O}(h^2)$, where h is the distance between nearest-neighbour nodes [147]: thus, the greater the fraction of nodes using the MLWFM, the better the overall accuracy should be. Values of $\delta > (\sqrt{2} - 0.2)$ were not investigated since these would correspond to the entire domain being solved by the MLWFM. We note that the error resulting with $\delta = 0.4$ is usually comparable to that for $\delta = 0.5$ around the optimal $c = 0.5$.

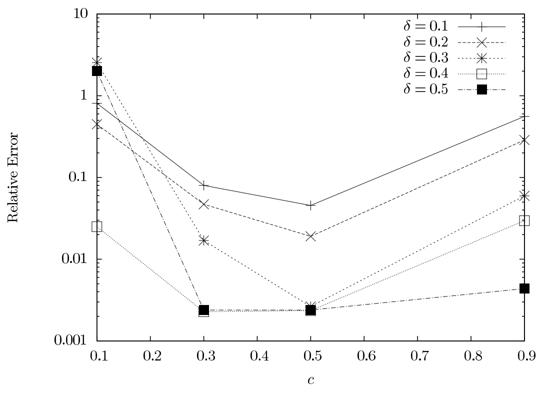
In fig. 6.12(a) we compare TE mode band diagrams generated by the MLWSFM and the PWEM [152]. The band diagram shows the frequencies of the four lowest-frequency modes that may propagate through the crystal. The agreement between the MLWSFM and the PWEM can be seen to be good. In fig. 6.12(b) we compare TM mode band diagrams from the MLWSFM and the PWEM codes. As in the TE case, good agreement can be seen between the two methods. However, previous work has showed that a purely strong-form meshless method may be used to solve the TM problem to a good accuracy [147], so the MLWSFM is comparatively less attractive for the TM case because it requires significantly more time to run, due to the numerical integration that is performed in the weak-form region.

Taking $\delta = 0.3$ and $c = 0.5$, we found the MLWSFM to be faster than the previous meshless method [147] that uses the weak form alone, by around 11% for $n = 400$ nodes, and 8% for $n = 900$ nodes, with one background cell per node in both cases, on account of the reduced amounts of computationally-intensive numerical integration required by the new method.

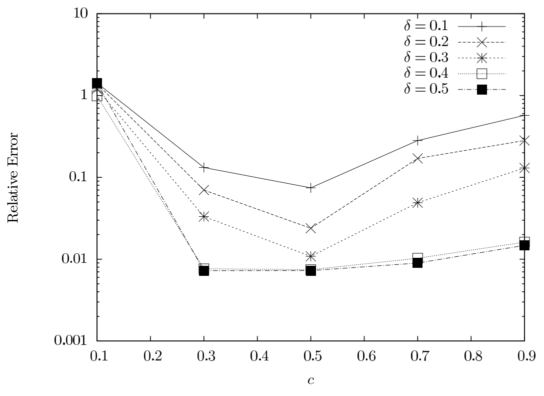
We note that in the aforementioned results, there are some unexpected inconsistencies, in that in some cases additional nodes can be seen to reduce the accuracy of



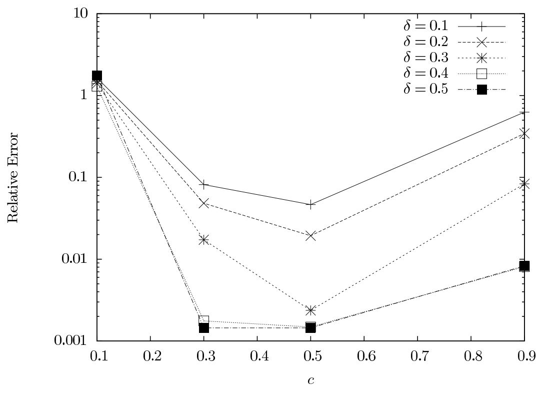
(a) TE, 1600 nodes, 1600 background cells



(b) TM, 1600 nodes, 1600 background cells

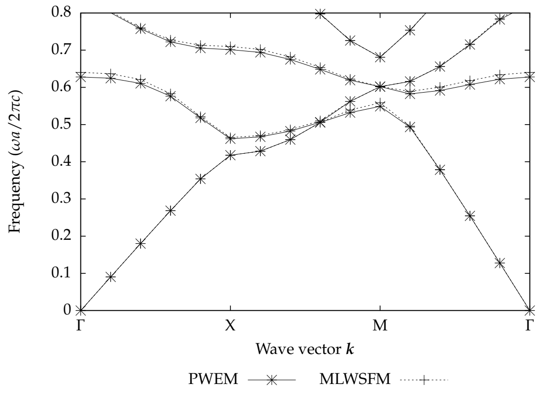


(c) TE, 1600 nodes, 1681 background cells

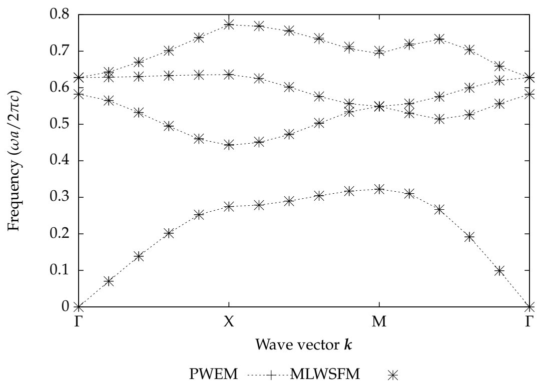


(d) TM, 1600 nodes, 1681 background cells

Figure 6.11: Average relative error between MLWSFM and PWEM for TE and TM using Wu's C4 RBF as a function of c and δ. Subfigures (a) and (c) are for TE; subfigures (b) and (d) are for TM. These figures are for a resolution of $n = 1600$ nodes.



(a) TE



(b) TM

Figure 6.12: Comparison of TE and TM mode band diagrams calculated by the PWEM and MLWSFM. TE mode used Wu's C^4 CSRBF with 1225 nodes and 1225 background cells. TM mode used Wu's C^4 CSRBF with 2500 nodes and 2500 background cells.

the final result. Experience gained after the calculation of these results suggested that these problems may have been related to the poor condition numbers of the matrices involved [163], and to the choice of eigensolver that was employed for these calculations. During the (subsequent) development and testing of the RBF-FD method for an eigenvalue problem on a periodic domain (see §5.4), similar problems arose; changing from the LAPACK `zggev` routine used here (based on the QZ algorithm and attempting to find all eigenvalues) to an iterative Krylov-Schur method [136], which seeks a few of the smallest eigenvalues, brought about much more consistent behaviour and significantly reduced the overall error.

6.5 A meshless RBF-FD method for 2D bandgap calculations

In this section, we expand the meshless radial basis function finite difference (RBF-FD) method developed for an eigenvalue problem with a periodic domain in §5.4.2 to the calculation of band structures of 2D photonic crystals for the TM polarisation mode. We begin by formulating the method in §6.5.1, and we present results from the new method in §6.5.2.

6.5.1 TM mode formulation

For the TM mode problem, we can formulate a strong form radial basis function finite-difference (RBF-FD) method, which is analogous in its formulation to the approach outlined in §6.3.2.

We begin solving (6.12) by substituting the wavefunction (6.14) into (6.12) to give:

$$-\left(\nabla + \mathrm{i}\boldsymbol{k}\right) \cdot \left(\nabla + \mathrm{i}\boldsymbol{k}\right) u = \varepsilon(\boldsymbol{x})\lambda u. \tag{6.50}$$

This expression gives rise to a generalised eigenvalue problem of the form:

$$\mathbf{A}(\boldsymbol{k})\boldsymbol{u} = \lambda \mathbf{B}\boldsymbol{u}, \tag{6.51}$$

where $\boldsymbol{u} = [u_1, u_2, ..., u_N]^{\mathrm{T}}$ are the eigenvectors of each eigensystem corresponding to the spatial patterns of the allowed modes of propagation through the crystal, and the eigenvalues λ correspond to the frequencies of the respective modes, as discussed more fully in §6.2.3.

Representing

$$u(\boldsymbol{x}) = \sum_{j=1}^{M} \psi_j\left(\boldsymbol{x}\right) u_j, \tag{6.52}$$

where M is the number of nodes in the RBF-FD stencil (or influence domain), as discussed in §5.4, with the notation $\psi_j(\boldsymbol{x}) = \psi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)$, and requiring that $u(\boldsymbol{x}_i) = u_i$ at all node locations $\boldsymbol{x}_i$ means that we must have $\psi_j(\boldsymbol{x}_i) = \delta_{ij}$.

The vector $\boldsymbol{\psi} = [\psi_1, \psi_2, \ldots, \psi_M, \mu]$ is found by solving the linear system $\mathbf{V}\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})$, where $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x}), 1]^{\mathrm{T}}$ is a vector formed by evaluating the globally-supported RBF between nodes in the RBF-FD stencil, and $\mathbf{V}$ is the matrix given by:

$$\mathbf{V} = \begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_1) & \cdots & \phi_M(\boldsymbol{x}_1) & 1 \\ \phi_1(\boldsymbol{x}_2) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_M(\boldsymbol{x}_2) & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_1(\boldsymbol{x}_M) & \phi_2(\boldsymbol{x}_M) & \cdots & \phi_M(\boldsymbol{x}_M) & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{6.53}$$

Substituting (6.52) into (6.50) gives the following **A** and **B** matrices for the generalised eigenvalue problem in (6.51):

$$\begin{aligned} \mathbf{A}_{ij} &= -\left(\nabla + \mathrm{i}\boldsymbol{k}\right) \cdot \left(\nabla + \mathrm{i}\boldsymbol{k}\right) \psi_j(\boldsymbol{x}_i), \\ &= -\nabla^2\psi_j\left(\boldsymbol{x}_i\right) - 2\mathrm{i}\boldsymbol{k} \cdot \nabla\psi_j(\boldsymbol{x}_i) + \boldsymbol{k}^2\psi_j(\boldsymbol{x}_i), \qquad (6.54) \\ \mathbf{B}_{ij} &= \varepsilon(\boldsymbol{x}_i)\psi_j(\boldsymbol{x}_i). \qquad (6.55) \end{aligned}$$

The matrix **A** in (6.54) may be expanded as:

$$\mathbf{A}_{ij} = -\mathbf{E}_{ij} - 2\mathrm{i}\boldsymbol{k} \cdot \boldsymbol{F}_{ij} + \boldsymbol{k}^2\mathbf{H}_{ij}, \tag{6.56}$$

where

$$\mathbf{E}_{ij} = \nabla^2\psi_j(\boldsymbol{x}_i), \tag{6.57}$$

$$\boldsymbol{F}_{ij} = \boldsymbol{\nabla}\psi_j(\boldsymbol{x}_i), \tag{6.58}$$

$$\mathbf{H}_{ij} = \psi_j(\boldsymbol{x}_i). \tag{6.59}$$

It follows from (6.52) and the subsequent discussion that **B** is diagonal, so we may re-cast the generalised eigenvalue problem (6.51) as a standard eigenvalue problem:

$$\mathbf{B}^{-1}\mathbf{A}(\boldsymbol{k})\boldsymbol{u} = \lambda\boldsymbol{u}, \tag{6.60}$$

and we have $\mathbf{B}_{i,i}^{-1} = (\mathbf{B}_{i,i})^{-1}$.

We solve the problem on a periodic, square domain, as illustrated in fig. 6.13. By imposing the conditions

$$u(x, 0) = u(x, b), \tag{6.61}$$

$$u(0, y) = u(a, y), \tag{6.62}$$

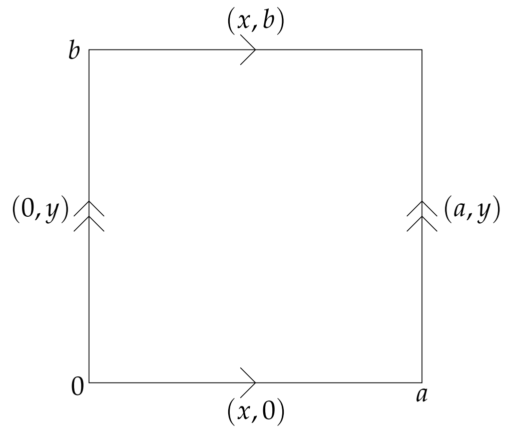


Figure 6.13: The domain of the system is made periodic by enforcing appropriate boundary conditions, see text.

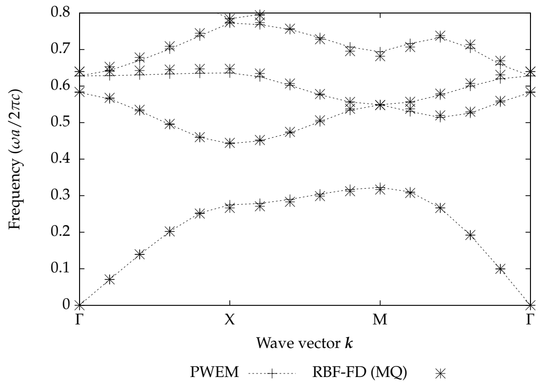
we enforce the periodicity of the domain. In our implementation, we account for this by using the minimum image distance (see *e.g.* [135, §A.1]) between nodes rather than the usual Euclidean distance.

Following a similar procedure to the formulation outlined in §5.4.2, we begin by establishing the influence domain for each node i. Periodic boundary conditions apply, so we consider all nodes whose minimum image distance from the node i is below the threshold radius r_p to be inside the influence domain.

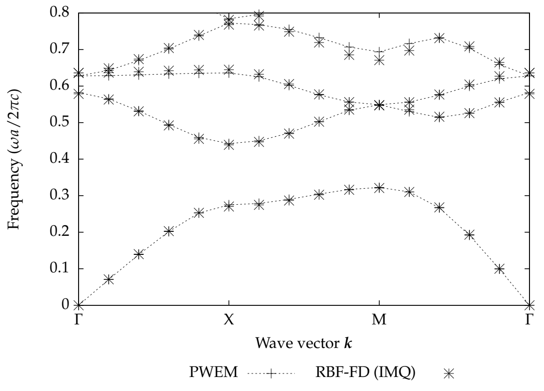
We require representations of the interpolant as well as its first and second derivatives in x and y to populate our eigensystem matrices, so we let the operator $\mathcal{L}$ in (5.50) take each of the values $\partial/\partial x$, $\partial/\partial y$, $\partial^2/\partial x^2$, and $\partial^2/\partial y^2$ in turn, and form each of the resulting linear systems of the form (5.53) using the appropriate influence domain. This yields vectors $\boldsymbol{w}^{(x)}$, $\boldsymbol{w}^{(y)}$, $\boldsymbol{w}^{(xx)}$ and $\boldsymbol{w}^{(yy)}$ respectively.

We populate the eigensystem matrices as follows, considering every node i in the unit cell, and all the nodes j in the influence domain of each node i:

$$\begin{aligned}
\mathbf{H}_{ij} &= \psi_{ij} \\
\mathbf{E}_{ij} &= w_{ij}^{(xx)} + w_{ij}^{(yy)} \\
(\mathbf{F}_x)_{ij} &= w_{ij}^{(x)} \\
(\mathbf{F}_y)_{ij} &= w_{ij}^{(y)}
\end{aligned} \tag{6.63}$$



(a) MQ



(b) IMQ

Figure 6.14: Comparison of TM mode band diagrams calculated by the PWEM and RBF-FD method, using multiquadric (MQ) and inverse multiquadric (IMQ) RBFs.

Finally we form the matrices **A** and **B** in (6.60) and apply an eigensolver to find the eigenvalues corresponding to the frequencies of propagation of the allowed modes at each point around the Brillouin zone.

6.5.2 Results

We ran the RBF-FD solver for the TM mode problem with two different choices of RBF, and the results were in good agreement with the results from the traditional PWEM solver (see §6.3.2 or [141]). We illustrate the results in fig. 6.14(a) for the multiquadric RBF, and in fig. 6.14(b) for the inverse multiquadric RBF. These were calculated for round rods of dielectric $\varepsilon = 8.9$ in air ($\varepsilon = 1$), with radius $r = 0.2a$. In the inverse multiquadric case, we used 400 nodes, with a stencil radius of 0.1, and a shape parameter for the RBF of 0.15. For the multiquadric case, we used the same parameters but changed the shape parameter to 0.12.

6.6 Conclusions and outlook

This section began by introducing photonic crystals as an area of active research and of potentially immense practical application. The physical and mathematical background discussed included essential concepts from crystallography, and the Maxwell equations, which govern the flow of light through photonic crystals, along with the simplifications often employed when these equations are solved. The emergence of band gaps in one-dimensional systems was illustrated, before two-dimensional systems were introduced.

The recently developed meshless methods for two-dimensional photonic crystal modelling were then introduced. The splitting up due to symmetry arguments of the 2D problem into transverse magnetic (TM) and transverse electric (TE) polarisation modes was discussed. Existing meshless local strong- and weak- form methods utilising compactly supported radial basis functions were described, as background for the introduction of a novel meshless local weak-strong form method (MLWSFM).

We gave details of the formulation of the MLWSFM for both TM and TE modes, and provided results illustrating that it is able to accurately calculate the band structures of a sample 2D photonic crystal, yielding results in good agreement with the plane wave expansion method (PWEM) solver. We also discussed how the application of a cloud-based architecture has assisted with accelerating the development and validation of the method; the results we presented were calculated using an implementation of the method running on the Microsoft Windows Azure cloud service.

We also gave the formulation of another novel meshless solver for 2D photonic crystals, which is a radial basis function finite difference (RBF-FD) method, formulated similarly to that in §5.4. We illustrated its performance with results that show for TM mode that it is capable of calculating the band structure of a sample crystal to a good accuracy and achieving good agreement with the PWEM solver.

Natural progress from this work would be to formulate an RBF-FD method for the TE mode, which would necessarily involve a method such as Galerkin's method, to avoid the difficulties associated arising from the requirement to take a derivative of the inverse of the dielectric function – which is discontinuous at the boundary of the rod and its surrounding material. Work has commenced on this method, as discussed in §7.2. It would also be productive to investigate the development of higher-order RBF-FD methods for 2D photonic problems, which would be formulated analogously to the one in §5.4.3, to potentially benefit from better convergence rates and higher accuracy.

Chapter 7

Summary and Outlook

7.1 Summary

We have investigated novel algorithms in computational physics, introducing a new fast multipole method for the energies of systems of pancake vortices in layered high-temperature superconductors which naturally handles periodic boundary conditions, as well as a meshless radial basis function finite difference method for an eigenvalue problem on a periodic domain and two meshless methods for computing the band structures of two dimensional photonic crystals.

After a general introduction to algorithms to set the context of the work, we reviewed in some detail the fast multipole method introduced in 1987 by Greengard and Rokhlin. We noted that since their seminal paper, many different versions of the fast multipole method have been introduced, along with a variety of applications far broader than the scope of the original method for the energy of particles interacting by the Coulomb or gravitational forces.

We introduced a novel fast multipole method for interactions governed by the 2D Yukawa potential – that is, where the inter-particle potential is given by a modified Bessel function of the second kind. This kind of potential arises in simulating the interaction of stacks of pancake vortices in layered high-temperature superconductors. Our algorithm uses multipole-like expansions based upon the rapidly-converging Gegenbauer addition formulae and achieves $\mathcal{O}(N)$ scaling. It also naturally handles periodic boundary conditions, so that it is possible to rapidly simulate infinitely-tiled systems. When truncating the series expansions at 20 terms, typical errors in the energy compared to the naïve method were of order $\mathcal{O}(10^{-7})$, and in our implementation, the multipole method was faster than the naïve method for numbers of particles $N \gtrsim 1100$.

We then turned our attention to the field of meshless methods, which is currently enjoying considerable research interest. We summarised the disadvantages of mesh-based methods that has lead to the development of the meshless methods, before considering the radial basis functions upon which many meshless methods are based.

We introduced the concept of the radial basis function finite difference method, and formulated a new radial basis function finite difference method for solving an eigenvalue problem on a periodic domain. For this example, we took the elliptic Helmholtz equation. We demonstrated that the RBF-FD method is capable of solving the problem to a high level of accuracy compared to the analytical solution. Further, we formulated a higher-order radial basis function finite difference method for the same equation, utilising ideas from Hermite interpolation.

Continuing our exploration of meshless methods, we considered the problem of solving the Maxwell equations on a periodic domain, as applicable to modelling a 2D photonic crystal. We developed a new meshless local weak-strong form hybrid method for calculating the band structure of 2D photonic crystals, which utilises compactly supported radial basis functions in both weak- and strong-forms and handles both the TM and TE mode polarisations. We demonstrated that its results are in good agreement with the leading conventional plane wave expansion method solver. We also formulated a new radial basis function finite difference solver for the TM mode problem in photonic crystal modelling and demonstrated that it achieves good agreement with the plane wave expansion method.

We also reviewed various technological advances that have taken place in scientific computing to enhance the computational performance available to engineers and scientists. We introduced various special-purpose hardware, which offered significant benefits for certain kinds of calculations, but was often very costly and had limited ranges of applications. We then considered how graphics processing units offer a means to run highly parallel numerical codes efficiently and with a favourable price-to-performance ratio. In the development and verification of the meshless local weak-strong form method, we applied another cutting-edge technology which is gaining interest in recent times, cloud computing. In our case, it allowed us to rent hardware to perform parameter sweeps in parallel, so that the time to validate and characterise the algorithm was bounded below by the time required to run the longest single simulation. Cloud based architectures can also bring significant additional benefits to scientific applications where they facilitate rapid, straightforward and secure sharing of data and encourage a modular design that makes it simple to swap different algorithms into use, and compare their results when run on the same input data.

7.2 Outlook and discussion

The methods reported on here have produced results that show they are viable choices for simulations in computational physics. However, there are several areas where further development would either yield improved performance, or where the domain of application of these methods could be expanded, and these are highlighted below.

RBF-FD method for the TE polarisation In §6.5.1 a radial basis function finite difference method was formulated for the TM polarisation of light in two dimensional photonic crystals and the results given in §6.5.2 show that the method is a promising scheme for calculating band structures. The challenge in formulating a similar method for the TE mode arises from a differential operator being applied to the dielectric function, which features a discontinuity. However, Galerkin's method can be used to overcome this challenge, as demonstrated in the case of the meshless local weak-strong form method in §6.4.1. It should therefore be possible to combine the RBF-FD scheme with a Galerkin weak formulation to solve the TE mode problem with an RBF-FD approach.

Work has commenced on implementing this scheme. The structure is very similar to that of the meshless local weak form method (MLWFM) for the TE mode, described in §6.4.1. The equation to be solved is identical and once again we apply Galerkin's method to (6.27), introducing integrals which we solve numerically by Gaussian quadrature:

$$\int \frac{1}{\varepsilon(\boldsymbol{x})}(\nabla + i\boldsymbol{k})u \cdot \overline{(\nabla + i\boldsymbol{k})v}\mathrm{d}\boldsymbol{x} = \lambda \int u\bar{v}\mathrm{d}\boldsymbol{x}. \tag{7.1}$$

The difference is that we switch from using a compactly supported RBF function to an RBF-FD representation. We substitute

$$u(\boldsymbol{x}) = \sum_{i=1}^{M} \psi_i(\boldsymbol{x})u_i, \tag{7.2}$$

$$v(\boldsymbol{x}) = \phi_j(\boldsymbol{x}), \tag{7.3}$$

into (7.1). In this case, as in §6.5.1, the vector $\boldsymbol{\psi} = [\psi_1, \psi_2, \ldots, \psi_M, \mu]$ is found by solving the linear system

$$\mathbf{V}\boldsymbol{\psi}^{\mathrm{T}}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x}),$$

where

$$\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x}), 1]^{\mathrm{T}},$$

is a vector formed by evaluating the globally-supported RBF and $\mathbf{V}$ is the matrix defined in (6.53).

The generalised eigenvalue problem is then

$$\mathbf{C}^1(\boldsymbol{k})\boldsymbol{u} = \lambda\mathbf{D}^1\boldsymbol{u}, \tag{7.4}$$

where

$$\begin{aligned}\mathbf{C}^1_{jl} = \int_\Omega \frac{1}{\varepsilon(\boldsymbol{x})} \big(&\nabla\psi_j(\boldsymbol{x}) \cdot \nabla\phi_l(\boldsymbol{x}) + \mathrm{i}\boldsymbol{k}\psi_j(\boldsymbol{x}) \cdot \nabla\phi_l(\boldsymbol{x}) \\ &-\mathrm{i}\boldsymbol{k}\nabla\psi_j(\boldsymbol{x}) \cdot \phi_l(\boldsymbol{x}) + \boldsymbol{k}^2\psi_j(\boldsymbol{x}) \cdot \phi_l(\boldsymbol{x})\big)\, \mathrm{d}\boldsymbol{x},\end{aligned} \tag{7.5}$$

and

$$\mathbf{D}^1_{jl} = \int_\Omega \psi_j(\boldsymbol{x}) \cdot \phi_l(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \tag{7.6}$$

where the integration is over Ω, the support domain in the Gaussian quadrature for the element in question.

We use the RBF-FD technique to represent $\boldsymbol{\nabla}\psi_j(\boldsymbol{x}) = [w_j^{(x)}(\boldsymbol{x}), w_j^{(y)}(\boldsymbol{x})]^{\mathrm{T}}$ where the weight vectors $\boldsymbol{w}^{(x)}$ and $\boldsymbol{w}^{(y)}$ are found by solving the linear systems

$$\mathbf{V}\boldsymbol{w}^{(x)}(\boldsymbol{x}) = [\phi_x(\|\boldsymbol{x}-\boldsymbol{x}_1\|), \phi_x(\|\boldsymbol{x}-\boldsymbol{x}_2\|), \ldots, \phi_x(\|\boldsymbol{x}-\boldsymbol{x}_M), 0]^{\mathrm{T}},$$

and

$$\mathbf{V}\boldsymbol{w}^{(y)}(\boldsymbol{x}) = [\phi_y(\|\boldsymbol{x}-\boldsymbol{x}_1\|), \phi_y(\|\boldsymbol{x}-\boldsymbol{x}_2\|), \ldots, \phi_y(\|\boldsymbol{x}-\boldsymbol{x}_M), 0]^{\mathrm{T}}.$$

Here, the matrix $\mathbf{V}$ is as defined in (6.53) and we used the notation $\phi_x(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) = \partial\phi(\|\boldsymbol{x}_i - \boldsymbol{x}\|)/\partial x|_{x=x_j}$.

We expand the $\mathbf{C}^1$ matrix into "sub-matrices" as follows:

$$\mathbf{C}^1_{jl} = \mathbf{S}^1_{jl} + \mathrm{i}\left(\mathbf{P}^1_{jl} - \mathbf{Q}^1_{jl}\right) + \boldsymbol{k}^2\mathbf{T}^1_{jl}, \quad \text{with} \tag{7.7}$$

$$\mathbf{S}^1_{jl} = \int_\Omega \frac{1}{\varepsilon(\boldsymbol{x})} \nabla\psi_j \cdot \nabla\phi_l \, \mathrm{d}\boldsymbol{x}, \tag{7.8}$$

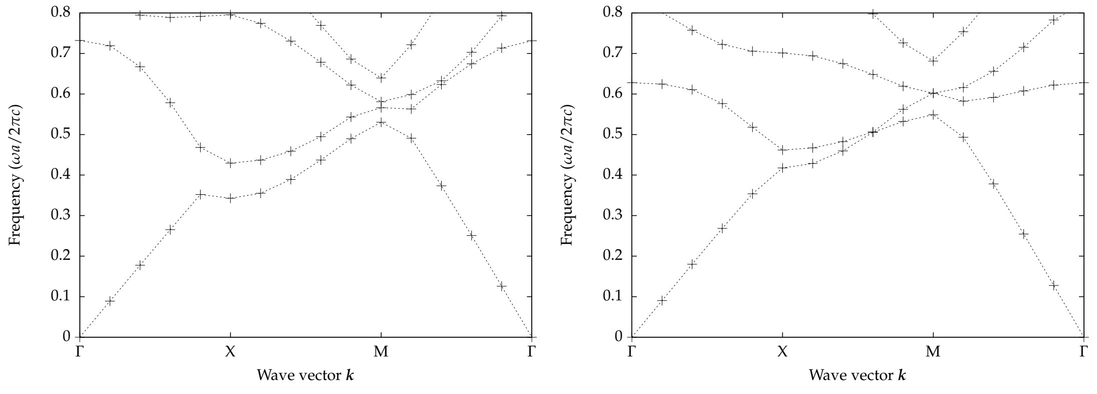
$$\mathbf{P}^1_{jl} = \int_\Omega \frac{1}{\varepsilon(\boldsymbol{x})} \boldsymbol{k}\psi_j \cdot \nabla\phi_l \, \mathrm{d}\boldsymbol{x}, \tag{7.9}$$

$$\mathbf{Q}^1_{jl} = \int_\Omega \frac{1}{\varepsilon(\boldsymbol{x})} \boldsymbol{k}\phi_l \cdot \nabla\psi_j \, \mathrm{d}\boldsymbol{x}, \tag{7.10}$$

$$\mathbf{T}^1_{jl} = \int_\Omega \frac{1}{\varepsilon(\boldsymbol{x})} \psi_j \cdot \phi_l \, \mathrm{d}\boldsymbol{x}. \tag{7.11}$$

The integration is carried out using Gaussian quadrature, as described in §6.4.1 and the matrices of the eigenvalue problem are then assembled. An existing eigensolver is used to find the eigenvalues corresponding to each allowed frequency of propagation at the $\boldsymbol{k}$ values around the irreducible Brillouin zone.

We have tested the current implementation by simulating round rods, of radius $r = 0.2a$, with dielectric $\varepsilon = 8.9$ in air ($\varepsilon = 1$). For certain sets of parameters there is a qualitative agreement between significant features of the results from the TE-mode RBF-FD method and the established solvers. In fig. 7.1 we present a band diagram comparison, which illustrates the qualitative agreement. In the figure, we used 100 nodes with 196 background cells, set the RBF shape parameter $c = 0.10$ and the RBF-



(a) RBF-FD (b) PWEM

Figure 7.1: Comparison of TE mode band diagrams calculated by (a) the PWEM and (b) the RBF-FD method, illustrating qualitative agreement of the main features of the band structure between the two methods. The RBF-FD method employed 100 nodes, 196 background cells, with an inverse multiquadric RBF, shape parameter $c = 0.10$.

FD influence domain radius to be 0.40. We used an inverse multiquadric radial basis function.

The initial results show that the method has some potential, but for many combinations of input parameters (the type of globally supported RBF used, the shape parameter chosen, the radius of the influence domain, and the numbers of nodes and Gaussian points used), the results yielded bear very little resemblance to the expected results. It is hoped that, with further refinement to and possibly debugging of the implementation, and tuning of the parameters, the method will become competitive with the established solvers and the MLWSFM for TE-mode calculations; in the short term, future work will be directed towards achieving this improvement.

Higher order RBF-FD methods for photonic crystals A higher-order RBF-FD scheme was introduced for the elliptic Helmholtz equation in §5.4.3, where information about the derivative as well as the function values is considered at some of the nodes in the stencil. This can lead to enhanced accuracy without increasing the dimensions of the stencil. Introducing higher-order RBF-FD schemes for photonic bandgap calculations would potentially increase the accuracy of these results in a similar way.

Parallelising and optimising the methods There is interest in bandgap optimisation and engineering, which typically involves a geometry optimisation in which the band structures of many different photonic crystal structures need to be calculated. Obviously, if the simulation method can run faster, the optimisation becomes more practicable. The implementations of the methods that gave the results in this work were single-threaded; by parallelising the work the wall-clock time would be reduced on a contemporary multi-core machine. Similarly, if the work were to be split into very many parallel threads (such as, *e.g.* using one thread per node in the MLWSFM), a GPU may be able to deliver significant speedup.

Similar arguments apply to the fast multipole method for the Yukawa potential; this method could foreseeably be called as part of a larger vortex dynamics simulation, which may require the evaluation of the energy of various particle configurations. Here, assigning a thread per box on a chosen refinement level would potentially allow the method to make use of parallel computational resources.

Alternative eigensolvers In the work carried out on photonic crystals, an eigensolver from LAPACK was used in the examples given here. This eigensolver was not ideal for the purpose as it attempts to find all N eigenvalues of the $N \times N$ matrices passed to it, where in photonic applications, it is typically only the few lowest bands – and, hence, the few lowest eigenvalues – that are of interest. Applying a more appropriate eigensolver, such as a subspace iteration algorithm (see, *e.g.* [164]), would foreseeably reduce runtime and wasted calculations significantly.

Additional photonic crystal geometries The geometries simulated in the work here are all based on a square unit cell, which implies a square lattice arrangement. There is interest in simulating photonic crystal structures with alternative geometries such as, for example, triangular lattices. This would involve a rhombic unit cell; it is likely that with some modification, the current algorithms would be suitable. The main areas that would require adjustment would be the definition of the unit cell and the rod shape within it, and the Gaussian integration scheme where the quadrilateral background cells would need to be made rhombic.

3D photonic crystal modelling As RBF based meshless methods are shown to be promising alternative schemes for calculating photonic band gaps in 2D, extending the proposed meshless methods to three-dimensional photonic crystal modelling would be of interest.

Bibliography

[1] Moore, G.E. Cramming more components onto integrated circuits. *Electronics*, 38:114 ff., 1965.

[2] Moore, G.E. Excerpts from A Conversation with Gordon Moore: Moore's Law. Intel video transcript, 2005.

[3] Dongarra, J. and Sullivan, F. Guest editors introduction to the top 10 algorithms. *Computing in Science and Engineering*, 2(1):22 –23, Jan/Feb 2000. doi:10.1109/MCISE.2000.814652.

[4] Greengard, L. and Rokhlin, V. A fast algorithm for particle simulation. *Journal of Computational Physics*, 73:325–348, 1987.

[5] Metropolis, N. and Ulam, S. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[6] Auerbach, D.J., Paul, W., Bakker, A.F., Lutz, C., Rudge, W.E., and Abraham, F.F. A special purpose parallel computer for molecular dynamics: Motivation, design, implementation, and application. *Journal of Physical Chemistry*, 91:4881–4890, 1987.

[7] Boehncke, K., Heller, H., Grubmuller, H., and Schulten, K. Molecular dynamics simulations on a systolic ring of Transputers. In A. Wagner, editor, *Transputer Research and Applications 3*, pages 83–94. IOS Press, Amsterdam, 1990.

[8] Hut, P. and Makino, J. Astrophysics on the GRAPE Family of special-purpose computers. *Science*, 283(5401):501–505, 1999. doi:10.1126/science.283.5401.501.

[9] Athanassoula, E., Bosma, A., Lambert, J.C., and Makino, J. Performance and accuracy of a GRAPE-3 system for collisionless N-body simulations. *Monthly Notices of the Royal Astronomical Society*, 293(4):369–380, 1998. doi:10.1046/j.1365-8711.1998.01102.x.

[10] Makino, J., Taiji, M., Ebisuzaki, T., and Sugimoto, D. GRAPE-4: a one-Tflops special-purpose computer for astrophysical N-body problem. In *Supercomputing '94: Proceedings of the 1994 conference on Supercomputing*, pages 429–438. IEEE Computer Society Press, Los Alamitos, CA, USA, 1994. ISBN 0-8186-6605-6.

[11] Makino, J., Hiraki, K., and Inaba, M. GRAPE-DR: 2 Pflops massively parallel computer with 512 core, 512 Gflops processor chips for scientific computing. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-764-3. doi:10.1145/1362622.1362647.

[12] Makino, J. GRAPE-DR. `http://www.astrogpu.org/talks/Jun%20Makino%20-%20GRAPE-DR.pdf`. Accessed 24 Mar 2010.

[13] Gschwind, M., Erb, D., Manning, S., and Nutter, M. An open source environment for Cell Broadband Engine system software. *Computer*, 40(6):37–47, 2007. doi:10.1109/MC.2007.192.

[14] Shi, G., Kindratenko, V., Pratas, F., Trancoso, P., and Gschwind, M. Application acceleration with the Cell Broadband Engine. *Computing in Science and Engineering*, 12:76–81, 2009. doi:10.1109/MCSE.2010.4.

[15] Williams, S., Shalf, J., Oliker, L., Kamil, S., Husbands, P., and Yelick, K. The potential of the Cell processor for scientific computing. In *CF '06: Proceedings of the 3rd conference on Computing frontiers*, pages 9–20. ACM, New York, NY, USA, 2006. ISBN 1-59593-302-6. doi:10.1145/1128022.1128027.

[16] Cockroft, A. *Sun performance and tuning*. SPARC & Solaris. SunSoft Press/Prentice-Hall, 1995.

[17] Habib, S., Pope, A., Lukić, Z., Daniel, D., Fasel, P., Desai, N., Heitmann, K., Hsu, C.H., Ankeny, L., Mark, G., Bhattacharya, S., and Ahrens, J. Hybrid petacomputing meets cosmology: The Roadrunner Universe project. *Journal of Physics: Conference Series*, 180(1):012019, 2009.

[18] Blagojevic, F., Stamatakis, A., Antonopoulos, C., and Nikolopoulos, D. RAxML-Cell: Parallel phylogenetic tree inference on the Cell Broadband Engine. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. March 2007. doi:10.1109/IPDPS.2007.370267.

[19] NVIDIA Corporation. *NVIDIA CUDA Programming Guide*, v2.3 edition, 2009.

[20] NVIDIA Corporation. *NVIDIA CUDA C Programming Guide*, v5.0 edition, October 2012.

[21] Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E., and Purcell, T.J. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[22] Fernando, R. and Kilgard, M.J. *The Cg Tutorial*. Addison-Wesley, 2003.

[23] Advanced Micro Devices, Inc. *ATI Stream Computing Technical Overview*.

[24] Khronos OpenCL Working Group (Ed.: Aaftab Munshi). *The OpenCL Specification*, version 1.0 revision 48 edition, June 2009.

[25] Khanna, G. and McKennon, J. Numerical modeling of gravitational wave sources accelerated by OpenCL. *Computer Physics Communications*, 181(9):1605 – 1611, 2010. doi:10.1016/j.cpc.2010.05.014.

[26] Schmidt, J., Piret, C., Zhang, N., Kadlec, B.J., Yuen, D.A., Liu, Y., Wright, G.B., and Sevre, E.O.D. Modeling of tsunami waves and atmospheric swirling flows

with graphics processing unit (GPU) and radial basis functions (RBF). *Concurrency and Computation: Practice and Experience*, 22(12):1813–1835, 2010. doi: 10.1002/cpe.1507.

[27] Rosario-Torres, S. and Velez-Reyes, M. Speeding up the MATLAB™ Hyperspectral Image Analysis Toolbox using GPUs and the Jacket Toolbox. In *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, 2009. WHISPERS '09. First Workshop on*. Aug. 2009. doi:10.1109/WHISPERS.2009.5289089.

[28] The Portland Group. *CUDA Fortran Programming Guide and Reference*, v 1.0 edition, November 2009.

[29] Wolfe, M. The PGI Accelerator Programming Model on NVIDIA GPUs. `http://www.pgroup.com/lit/articles/insider/v1n1a1.htm`, June 2009.

[30] Mell, P. and Grance, T. The NIST Definition of Cloud Computing. Special Publication 800-145, National Institution of Standards and Technology, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, September 2011.

[31] Johnston, S.J., Cox, S.J., and Takeda, K. *Grid and Cloud Database Management*, chapter 9: Scientific computation and data management using Microsoft Windows Azure, pages 169-192. Springer, Editors G. Aloisio and S. Fiore, July 2011.

[32] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. Above the clouds: A Berkeley view of cloud computing. Technical report, EECS Department, University of California, Berkeley, February 2009. UCB/EECS-2009-28.

[33] Foster, I. and Kesselman, C., editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufman, 1999.

[34] Viega, J. Cloud Computing and the Common Man. *Computer*, 42(8):106–108, August 2009. doi:10.1109/MC.2009.252.

[35] Johnston, S., O'Brien, N., Lewis, H., Hart, E., White, A., and Cox, S. Clouds in space: Scientific computing using Windows Azure. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1):2, 2013. doi:10.1186/2192-113X-2-2.

[36] Coles, S.J., Frey, J.G., Hursthouse, M.B., Light, M.E., Milsted, A.J., Carr, L.A., DeRoure, D., Gutteridge, C.J., Mills, H.R., Meacham, K.E., Surridge, M., Lyon, E., Heery, R., Duke, M., and Day, M. An E-Science Environment for Service Crystallography – from Submission to Dissemination. *Journal of Chemical Information and Modeling*, 46(3):1006–1016, 2006.

[37] Hazelhurst, S. Scientific computing using virtual high-performance computing: a case study using the Amazon elastic computing cloud. In *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, SAICSIT '08, pages 94–103. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-286-3. doi:10.1145/1456659.1456671.

[38] Jonsson, L.E. and Magro, W.R. Comparative performance of InfiniBand Architecture and Gigabit Ethernet interconnects on Intel Itanium 2 microarchitecture-based clusters. Technical report, Intel Americas, Inc., 2003.

[39] Krishnan, S. *Programming Windows Azure: Programming the Microsoft Cloud*. O'Reilly Media Inc, Sebastopol, CA, May 2010. ISBN 978-0-596-80197-7.

[40] Napper, J. and Bientinesi, P. Can cloud computing reach the TOP500? In *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, UCHPC-MAW '09, pages 17–20. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-557-4. doi:10.1145/1531666.1531671.

[41] Berriman, G.B., Deelman, E., Juve, G., Regelson, M., and Plavchan, P. The application of cloud computing to astronomy: A study of cost and performance. *ArXiv CoRR*, abs/1010.4813, 2010.

[42] Johnston, S.J., Lewis, H., Hart, E.E., White, A., O'Brien, N.S., Takeda, K., and Cox, S.J. Space situational awareness using a cloud based architecture. In *2011 Beijing Space Sustainability Conference*. Secure World Foundation, October 2011.

[43] O'Brien, N.S., Johnston, S.J., Hart, E.E., Djidjeli, K., and Cox, S.J. Exploiting cloud computing for algorithm development. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, pages 336–342. October 2011. doi:10.1109/CyberC.2011.60.

[44] Pfalzner, S. and Gibbon, P. *Many-body Tree Methods in Physics*. Cambridge University Press, 1996.

[45] Allen, M.P. and Tildesley, D.J. *Computer Simulation of Liquids*. Oxford University Press, 1987.

[46] Board, J. and Schulten, L. The fast multipole algorithm. *Computing in Science and Engineering*, 2(1):76 –79, Jan/Feb 2000. doi:10.1109/5992.814662.

[47] Fangohr, H., Price, A.R., Cox, S.J., de Groot, P.A.J., and Daniell, G.J. Efficient methods for handling long-range forces in particle-particle simulations. *Journal of Computational Physics*, 162(2):372–384, August 2000.

[48] McKenney, A., Greengard, L., and Mayo, A. A fast poisson solver for complex geometries. *Journal of Computational Physics*, 118(2):348 – 355, 1995. doi:10.1006/jcph.1995.1104.

[49] Banegas, A. Fast Poisson solvers for problems with sparsity. *Mathematics of Computation*, 32(142):441–446, 1978. doi:10.1090/S0025-5718-1978-0483338-8.

[50] Abdullah, A.R. The four point explicit decoupled group (EDG) Method: a fast Poisson solver. *International Journal of Computer Mathematics*, 38(1-2):61–70, 1991. doi:10.1080/00207169108803958.

[51] Hockley, R.W. and Eastwood, J.W. *Computer simulation using particles*. Taylor and Francis, 1988.

[52] Barnes, J. and Hut, P. A hierarchical $\mathcal{O}(N \log N)$ force-calculation algorithm. *Nature*, 324:446–449, 1986. doi:10.1038/324446a0.

[53] Pincus, M.R. and Scheraga, H.A. An approximate treatment of long-range interactions in proteins. *Journal of Physical Chemistry*, 81(16):1579–1583, 1977. doi: 10.1021/j100531a013.

[54] Carrier, J., Greengard, L., and Rokhlin, V. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on Scientific and Statistical Computing*, 9(4):669–686, July 1988.

[55] Elliott, W.D. and Board, J.A. Fast Fourier transform accelerated fast multipole methods. *SIAM Journal on Scientific Computing*, 17(2):398–415, 1996.

[56] Greengard, L. and Rokhlin, V. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997. doi: 10.1017/S0962492900002725.

[57] Cheng, H., Greengard, L., and Rokhlin, V. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155(2):468–498, 1999.

[58] Darve, E. The fast multipole method i: Error analysis and asymptotic complexity. *SIAM Journal on Numerical Analysis*, 38(1):98–128, 2001.

[59] Pan, Y.C. and Chew, W.C. A hierarchical fast-multipole method for stratified media. *Microwave and Optical Technology Letters*, 27(1):13–17, 2000.

[60] Fong, W. and Darve, E. The black-box fast multipole method. *Journal of Computational Physics*, 228:8712–8725, 2009.

[61] Ying, L., Biros, G., and Zorin, D. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196:591–626, 2004.

[62] Tinkham, M. *Introduction to Superconductivity*. McGraw-Hill International, Inc., New York, 2$^{\text{nd}}$ edition, 1996.

[63] Abramowitz, M. and Stegun, I.A. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1965.

[64] Jensen, H.J., Brass, A., and Berlinsky, A.J. Lattice deformations and plastic flow through deformations in a two-dimensional model for flux pinning in type-II superconductors. *Physical Review Letters*, 60:1676, 1988.

[65] Reichhardt, C., Olson, C.J., and Nori, F. Dynamic phases of vortices in superconductors with periodic pinning. *Physical Review Letters*, 78:2648, 1997.

[66] Ryu, S., Hellerqvist, M., Doniach, S., Kapitulnik, A., and Stroud, D. Dynamical phase transition in a driven disordered vortex lattice. *Physical Review Letters*, 77:5114, 1996.

[67] Greengard, L. and Rokhlin, V. On the efficient implementation of the fast multipole method. Technical Report RR-602, Dept. Comp. Sci., Yale University, New Haven, CT, 1988.

[68] Pringle, G.J. *Numerical Study of Three-Dimensional Flow using Fast Parallel Particle Algorithms*. Ph.D. thesis, Department of Mathematics, Napier University, Edinburgh, 1994.

[69] Engheta, N., Murphy, W.D., Rokhlin, V., and Vassiliou, M.S. The fast multipole method (FMM) for electromagnetic scattering problems. *IEEE Transactions on Antennas and Propagation*, 40(6):634–641, 1992.

[70] Epton, M.A. and Dembart, B. Multipole translation theory for three-dimensional Laplace and Helmholtz equations. *SIAM Journal on Scientific Computing*, 16(4):865–897, 1995.

[71] Rokhlin, V. Sparse diagonal forms for translation operators for the Helmholtz equation in two dimensions. *Applied and Computational Harmonic Analysis*, 5:36–67, 1998.

[72] Watson, G.N. *Theory of Bessel Functions*. Cambridge University Press, 2nd edition, 1944.

[73] Gradshteyn, I.S. and Ryzhig, I.M. *Table of Integrals, Series and Products*. Academic Press, San Diego, 5th edition, 1994.

[74] Press, W.H., Vetterling, W.T., Flannery, B.P., and Teukolsky, S.A. *Numerical Recipes in Fortran*. Cambridge University Press, 2nd edition, 1992.

[75] Li, S. and Liu, W.K. *Meshfree Particle Methods*. Springer, 2004.

[76] Liu, G. and Gu, Y. *An Introduction to Meshfree Methods and Their Programming*. Springer, 2005.

[77] Turner, M.J., Clough, R.W., Martin, R.W., and Topp, L.J. Stiffness and deflection analysis of complex structures. *Journal of the Aeronautical Sciences*, 23:805–823, 1956.

[78] Wilson, E.L. and Nickell, R.E. Application of the finite element method to heat conduction analysis. *Nuclear Engineering and Design*, 4(3):276 – 286, 1966. doi: 10.1016/0029-5493(66)90051-3.

[79] Silvester, P. and Chari, M. Finite element solution of saturable magnetic field problems. *Power Apparatus and Systems, IEEE Transactions on*, PAS-89(7):1642 – 1651, sept. 1970. doi:10.1109/TPAS.1970.292812.

[80] Bathe, K.J. and Khoshgoftaar, M.R. Finite element free surface seepage analysis without mesh iteration. *International Journal for Numerical and Analytical Methods in Geomechanics*, 3(1):13–22, 1979. doi:10.1002/nag.1610030103.

[81] Zienkiewicz, O.C., Taylor, R.L., and Zhu, J.Z. *The Finite Element Method for Fluid Dynamics*. Elsevier Butterworth-Heinemann, 6th edition, 2005.

[82] Rao, S.S. *The Finite Element Method in Engineering*. Butterworth-Heinemann, 5th edition, 2010.

[83] Donald, B.J.M. *Practical Stress Analysis with Finite Elements*. Glasnevin Publishing, 2007.

[84] Dobson, D.C., Gopalakrishnan, J., and Pasciak, J.E. An efficient method for band structure calculations in 3D photonic crystals. *Journal of Computational Physics*, 149:363–376, 1998.

[85] Axmann, W. and Kuchment, P. 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. doi:10.1006/jcph.1999.6188.

[86] Beckett, D.H., Cox, S.J., Generowicz, J.M., Hiett, B.P., Molinari, M., and Thomas, K.S. Application of finite element methods to photonic crystal modelling. In *Computation in Electromagnetics, 2002. CEM 2002. The Fourth International Conference on (Ref. No. 2002/063)*, page 2 pp. April 2002. doi:10.1049/ic:20020148.

[87] Fietz, C., Urzhumov, Y., and Shvets, G. Complex $\boldsymbol{k}$ band diagrams of 3D metamaterial/photonic crystals. *Opt. Express*, 19(20):19027–19041, Sept 2011. doi:10.1364/OE.19.019027.

[88] Kumar, S. Use of unsymmetric finite element method in impact analysis of composite laminates. *Finite Elements in Analysis and Design*, 47(4):373–377, 2011. doi:10.1016/j.finel.2010.12.016.

[89] Han, W. and Meng, X. Some studies of the reproducing kernel particle method. In M. Griebel and M.A. Schweitzer, editors, *Meshfree Methods for Partial Differential Equations*, volume 26 of *Lecture Notes in Computational Science and Engineering*,

pages 193–210. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-43891-5. doi: 10.1007/978-3-642-56103-0_13.

[90] Oskooi, A.F., Roundy, D., Ibanescu, M., Bermel, P., Joannopoulos, J.D., and Johnson, S.G. MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method. *Computer Physics Communications*, 181:687–702, January 2010. doi:10.1016/j.cpc.2009.11.008.

[91] Eymard, R., Gallouët, T., and Herbin, R. Finite volume methods. `http://www.cmi.univ-mrs.fr/~herbin/PUBLI/bookevol.pdf`, October 2006. An update to an article published in Handbook of Numerical Analysis, P.G. Ciarlet, J.L. Lions eds, vol 7, pp 713-1020.

[92] Obayya, S. *Computational Photonics*. Wiley, October 2010.

[93] Ismagilov, T. and Kuzmin, A. Modelling photonic crystal waveguides using finite volume method. In *Days on Diffraction (DD), 2011*, pages 87–91. May-June 2011 2011. doi:10.1109/DD.2011.6094371.

[94] Di Pietro, D.A. and Ern, A. *Mathematical aspects of discontinuous Galerkin methods*. Mathématiques et Applications. Springer, 2012.

[95] Peskin, C.S. The immersed boundary method. *Acta Numerica*, 11:479–517, 0 2002. doi:10.1017/S0962492902000077.

[96] Peskin, C.S. *Flow patterns around heart valves: A digital computer method for solving the equations of motion*. Ph.D. thesis, Albert Einstein College of Medicine, 1977.

[97] Liu, G.R. *Mesh Free Methods: Moving Beyond the Finite Element Method*. CRC Press, 2002.

[98] Slater, J.C. Electronic energy bands in metals. *Physical Review*, 45:794–801, Jun 1934. doi:10.1103/PhysRev.45.794.

[99] Brunner, H. *Collocation Methods for Volterra Integral and Related Functional Differential Equations*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, November 2004.

[100] Nayroles, B., Touzot, G., and Villon, P. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, 10:307–318, 1992. 10.1007/BF00364252.

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

[102] Kansa, E. Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics – i surface approximations and partial derivative estimates. *Computers & Mathematics with Applications*, 19(8-9):127–145, 1990. doi:10.1016/0898-1221(90)90270-T.

[103] Wu, Z. Hermite-Birkhoff interpolation of scattered data by radial basis functions. *Approximation Theory and its Applications*, 8:1–10, 1992. 10.1007/BF02836101.

[104] Liu, G.R. and Gu, Y.T. A meshfree method: meshfree weak-strong (MWS) form method, for 2D solids. *Computational Mechanics*, 33:2–14, 2003. 10.1007/s00466-003-0477-5.

[105] Liu, G., Gu, Y., and Wu, Y. A Meshfree Weak-Strong form (MWS) method for fluid mechanics. In *Proceedings of the International Workshop on MeshFree Methods*, pages 73–78. Lisbon, Portugal, 2003.

[106] Mirzaei, D. and Schaback, R. Direct meshless local Petrov-Galerkin (DMLPG) method: A generalized MLS approximation. *Applied Numerical Mathematics*, 68(0):73 – 82, 2013. doi:10.1016/j.apnum.2013.01.002.

[107] Belytschko, T., Lu, Y., Gu, L., and Tabbara, M. Element-free Galerkin methods for static and dynamic fracture. *International Journal of Solids and Structures*, 32(17-18):2547–2570, 1995. doi:10.1016/0020-7683(94)00282-2. A Symposium on

the Dynamic Failure Mechanics of Modern Materials In Memory of Professor J. Duffy.

[108] Li, S., Hao, W., and Liu, W.K. Numerical simulations of large deformation of thin shell structures using meshfree methods. *Computational Mechanics*, 25:102–116, 2000. doi:10.1007/s004660050463.

[109] Liu, W.K., Jun, S., Sihling, D.T., Chen, Y., and Hao, W. Multiresolution reproducing kernel particle method for computational fluid dynamics. *International Journal for Numerical Methods in Fluids*, 24(12):1391–1415, 1997. doi:10.1002/(SICI)1097-0363(199706)24:12<1391::AID-FLD566>3.0.CO;2-2.

[110] Chinchapatnam, P.P., Djidjeli, K., Nair, P.B., and Tan, M. A compact RBF-FD based meshless method for the incompressible Navier-Stokes equations. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 223(3):275–290, 2009. doi:10.1243/14750902JEME151.

[111] Cleary, P.W. Modelling confined multi-material heat and mass flows using SPH. *Applied Mathematical Modelling*, 22(12):981–993, 1998. doi:10.1016/S0307-904X(98)10031-8.

[112] Hart, E.E., Cox, S.J., Djidjeli, K., and Kubytskyi, V.O. Solving an eigenvalue problem with a periodic domain using radial basis functions. *Engineering Analysis with Boundary Elements*, 33(2):258–262, 2009. doi:10.1016/j.enganabound.2008.04.009.

[113] Hart, E.E. *Algorithms and Technologies for Photonic Crystal Modelling*. Ph.D. thesis, School of Engineering Sciences, University of Southampton, 2010.

[114] Nicomedes, L., Mesquita, C., and Moreira, J. Calculating the band structure of photonic crystals through the meshless local Petrov-Galerkin (MLPG) method and periodic shape functions. *Magnetics, IEEE Transactions on*, 48(2):551–554, February 2012. doi:10.1109/TMAG.2011.2175206.

[115] Aluru, N.R. and Li, G. Finite cloud method: a true meshless technique based on a fixed reproducing kernel approximation. *International Journal for Numerical Methods in Engineering*, 50(10):2373–2410, 2001. doi:10.1002/nme.124.

[116] Burke, D.R. and Smy, T.J. Optical mode solving for complex waveguides using a finite cloud method. *Optics Express*, 20(16):17783–17796, Jul 2012. doi:10.1364/OE.20.017783.

[117] Burke, D.R. and Smy, T.J. A meshless based solution to vectorial mode fields in optical microstructured waveguides. In *Proc. SPIE 8255, Physics and Simulation of Optoelectronic Devices XX, 82550L (February 6, 2012)*, pages 82550L–82550L–10. 2012. doi:10.1117/12.907355.

[118] Wu, Z. Compactly supported positive definite radial functions. *Advances in Computational Mathematics*, 4(1):283–292, 1995.

[119] Wendland, H. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396, 1995.

[120] Buhmann, M.D. A new class of radial basis functions with compact support. *Mathematics of Computation*, 70(233):307–318, 2001.

[121] Micchelli, C.A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986. 10.1007/BF01893414.

[122] Zhang, X., Song, K.Z., Lu, M.W., and Liu, X. Meshless methods based on collocation with radial basis functions. *Computational Mechanics*, 26:333–343, 2000.

[123] Oñate, E., Idelsohn, S., Zienkiewicz, O.C., and Taylor, R.L. A finite point method in computational mechanics. Applications to convective transport and fluid flow. *International Journal for Numerical Methods in Engineering*, 39(22):3839–3866, 1996. doi:10.1002/(SICI)1097-0207(19961130)39:22<3839::AID-NME27>3.0.CO;2-R.

[124] Girault, V. Theory of a finite difference method on irregular networks. *SIAM Journal on Numerical Analysis*, 11(2):260–282, 1974. doi:10.1137/0711026.

[125] Atluri, S.N. and Zhu, T. A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics. *Computational Mechanics*, 22:117–127, 1998. doi: 10.1007/s004660050346.

[126] Lucy, L.B. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, Dec. 1977. doi:10.1086/112164.

[127] Benz, W. Smooth Particle Hydrodynamics – a review. In J. R. Buchler, editor, *Numerical Modelling of Nonlinear Stellar Pulsations Problems and Prospects*, page 269. 1990.

[128] Bollig, E.F., Flyer, N., and Erlebacher, G. Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs. *Journal of Computational Physics*, 231(21):7133 – 7151, 2012. doi:10.1016/j.jcp.2012.06.030.

[129] Li, J. and Hon, Y.C. Domain decomposition for radial basis meshless methods. *Numerical Methods for Partial Differential Equations*, 20(3):450–462, 2004. doi:10.1002/num.10096.

[130] Duan, Y., Hon, Y., and Zhao, W. Stability estimate on meshless unsymmetric collocation method for solving boundary value problems. *Engineering Analysis with Boundary Elements*, 37(4):666 – 672, 2013. doi:10.1016/j.enganabound.2013.02.003.

[131] Wendland, H. On the stability of meshless symmetric collocation for boundary value problems. *BIT Numerical Mathematics*, 47(2):455–468, 2007. doi:10.1007/s10543-007-0121-4.

[132] Wright, G.B. and Fornberg, B. Scattered node compact finite difference-type formulas generated from radial basis functions. *Journal of Computational Physics*, 212:99–123, February 2006. doi:10.1016/j.jcp.2005.05.030.

[133] Perrone, N. and Kao, R. A general finite difference method for arbitrary meshes. *Computers & Structures*, 5(1):45–57, 1975. doi:10.1016/0045-7949(75)90018-8.

[134] Tolstykh, A.I. and Shirobokov, D.A. On using radial basis functions in a "finite difference mode" with applications to elasticity problems. *Computational Mechanics*, 33:68–79, 2003. 10.1007/s00466-003-0501-9.

[135] Balbuena, P. and Seminario, J. *Molecular Dynamics: From Classical to Quantum Methods*. Elsevier Science, 1999.

[136] Hernandez, V., Roman, J.E., and Vidal, V. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(3):351–362, 2005.

[137] Collatz, L. *The Numerical Treatment of Differential Equations*. Springer, Berlin, 1960.

[138] Lele, S.K. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992. doi:10.1016/0021-9991(92)90324-R.

[139] Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[140] Marquardt, D. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. doi:10.1137/0111030.

[141] Johnson, S.G. and Joannopoulos, J.D. Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis. *Optics Express*, 8(3):173–190, 2001.

[142] John, S. Strong localization of photons in certain disordered dielectric superlattices. *Physical Review Letters*, 58(23):2486–2489, Jun 1987. doi:10.1103/PhysRevLett.58.2486.

[143] Yablonovitch, E. Inhibited spontaneous emission in solid-state physics and electronics. *Physical Review Letters*, 58(20):2059–2062, May 1987. doi:10.1103/PhysRevLett.58.2059.

[144] Rayleigh, L. 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.

[145] Subramaniam, G. Fabrication of photonic crystals. In *Emerging Technologies - Nanoelectronics, 2006 IEEE Conference on*, pages 357–359. January 2006. doi:10.1109/NANOEL.2006.1609747.

[146] Kao, C.Y., Osher, S., and Yablonovitch, E. Maximizing band gaps in two-dimensional photonic crystals by using level set methods. *Applied Physics B: Lasers and Optics*, 81:235–244, 2005. 10.1007/s00340-005-1877-3.

[147] Hart, E.E., Cox, S.J., and Djidjeli, K. Compact RBF meshless methods for photonic crystal modelling. *Journal of Computational Physics*, 230(12):4910–4921, 2011. doi:10.1016/j.jcp.2011.03.010.

[148] Myers, H.P. *Introductory Solid State Physics*. CRC Press, 1997.

[149] Kittel, C. *Introduction to Solid State Physics*. Wiley, 8th edition, 2004.

[150] Bloch, F. Über die Quantenmechanik der Elektronen in Kristallgittern. *Zeitschrift für Physik A Hadrons and Nuclei*, 52:555–600, 1929. 10.1007/BF01339455.

[151] Floquet, G. Sur les équations différentielles linéaires à coefficients périodiques. *Annales scientifiques de l'École Normale Supérieure*, 12:47–88, 1883.

[152] Joannopoulos, J.D., Johnson, S.G., Winn, J.N., and Meade, R.D. *Photonic Crystals: Molding the Flow of Light*. Princeton University Press, 2nd edition, 2008.

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

[154] Hecht, E. *Optics*. Addison Wesley, San Francisco, 4th edition, 2002.

[155] Pedrotti, F.L. and Pedrotti, L.S. *Introduction to Optics*. Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 1987.

[156] Rayleigh, L. On the reflection of light from a regularly stratified medium. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 93(655):pp. 565–577, 1917.

[157] Busch, K., Lölkes, S., Wehrspohn, R.B., and Föll, H., editors. *Photonic Crystals: Advances in design, fabrication and characterization*. Wiley-VCH, Weinheim, 2004.

[158] Jun, S., Cho, Y.S., and Im, S. Moving least-square method for the band-structure calculation of 2D photonic crystals. *Optics Express*, 11(6):541–551, March 2003. doi:10.1364/OE.11.000541.

[159] Sun Microsystems, Inc. *Sun Performance Library User's Guide*, Sun Studio 12 Update 1 edition, July 2009.

[160] Johnson, S.G. and the Joannopoulos Ab Initio Physics Group. *MPB User Reference*, 2009.

[161] Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., and Good, J. On the use of cloud computing for scientific workflows. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 640–645. December 2008. doi:10.1109/eScience.2008.167.

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

[163] Fasshauer, G.E. *Meshfree approximation methods with MATLAB*. World Scientific Publishing, 2007.

[164] Saad, Y. *Numerical methods for large eigenvalue problems*. SIAM, 2nd edition, 2011.