University of Reading

School of Mathematics, Meteorology & Physics

A COLLOCATION METHOD FOR HIGH FREQUENCY SCATTERING BY CONVEX POLYGONS.

Stephen Arden

August 2005

This dissertation is submitted to the Department of Mathematics in partial fulfilment of the requirements for the degree of Master of Science

ABSTRACT

We consider the problem of high frequency acoustic scattering by sound soft convex polygons. Standard numerical schemes have a computational cost that grows in direct proportion with the frequency of the incident wave. Recently Chandler-Wilde and Langdon presented a Galerkin boundary element method whereby the products of plane wave basis functions with piecewise polynomials supported on a graded mesh were incorporated into the approximation space. For this scheme the degrees of freedom grows only logarithmically with the frequency. We propose and implement a collocation method which is more computationally efficient when using a similar approximation space and we give a thorough account of the theory used in the implementation of this method. Although the method does not lend itself well to the same rigorous error analysis as the Galerkin method, we are able to present the numerical results obtained and conduct a comparison analysis of our method with the existing numerical results obtained by the Galerkin method. These results indicate that we have obtained a substantial reduction in the computational cost whilst sustaining a similar order of convergence.

Acknowledgments

I would like to acknowledge Dr. Stephen Langdon who supervised this project to whom I am very grateful for all the help and knowledge he has given me to aid the completion of this dissertation. I would also like to acknowledge the support and encouragement of all in the mathematics department, especially those on the MSc courses and my friends who had to put up with me during the stressful times over the last year.

Finally, I would like to recognise the financial assistance from the EPSRC studentship.

Declaration

I confirm this report is all my own work and any material taken from other sources has been fully and properly acknowledged.

Stephen Arden

CONTENTS

1. Introduction 7

2. Background Information 9

2.1 Wave Scattering 9

2.2 Direct Scattering Problem 9

2.2.1 The Helmholtz Equation 10

2.2.2 Boundary Condition on the Scatterer 10

2.2.3 Sommerfeld Radiation Condition 11

2.3 Greens Representation and Reformulation 11

2.4 High Frequency 13

2.5 The Problem On One Side 15

2.6 The Full Parametrised Problem 19

2.7 The Mesh 22

3. Boundary Element Methods 25

3.1 Galerkin Method 25

3.2 Collocation Method 27

3.2.1 Choosing the Basis Functions (Collocation Method) 28

3.2.2 Choosing the Collocation Points 30

4. Collocation Method Implementation 32

4.1 LHS1 33

4.2 LHS2 34

4.2.1 Case (i) 34

4.2.2 Case (ii) 36

4.3 RHS 36

4.3.1 RHS - Shadow Sides 36

4.3.2 RHS - Illuminated Sides 37

4.4 Gaussian Quadrature 38

4.4.1 Non-Oscillatory Functions 40
4.4.2 Oscillatory Functions 40
4.5 Gaussian Quadrature Error 42
4.6 Machine Accuracy 43

5. Numerical Results and Discussion 47

6. Conclusions and Further Work 57

LIST OF FIGURES

2.1 Approximation to one complete wavelength 14
2.2 Scattering on a half plane by a convex polygon, consisting of incident, reflected and diffracted waves 16
2.3 Speculative view on what the diffracted wave may look like across one side of a polygon (of length one) with equal diffraction at each of the corners 19
2.4 Mesh points for one side of square of length 2π, wavenumber $k = 10$ for $N = 2, 4, 8, 16, 32$. *Left*: mesh for υ_+. *Right*: mesh for υ_- 24

4.1 Example of what the oscillating function may look like across a basis function many wavelengths long 41

5.1 Total Acoustic Field for k = 10, scattering by a square with incident wave $\pi/4$ measured from the downward vertical (top left corner to bottom right) 48
5.2 Degrees of freedom for increasing wavenumbers for $N = 64$ and $N = 128$ 51
5.3 Square, coordinates $(0,0)$ $(2\pi,0)$ $(2\pi,2\pi)$ $(0,2\pi)$. $\theta = \pi/4$, $k = 10$. 55
5.4 Square, coordinates $(0,0)$ $(2\pi,0)$ $(2\pi,2\pi)$ $(0,2\pi)$. $\theta = \pi/8$, $k = 5$ . . 55
5.5 Hexagon, coordinates $(0,0)$ $(2,1)$ $(2,2)$ $(0,3)$ $(-2,2)$ $(-2,1)$. $\theta = \pi/2$, $k = 20$ 56
5.6 Polygon, coordinates $(0,0)$ $(\frac{2}{3}\pi,-\pi)$ $(\frac{4}{3}\pi,-\pi)$ $(2\pi,\pi)$ $(\frac{2}{3}\pi,\pi)$. $\theta = \pi/8$, $k = 10$ 56

LIST OF TABLES

4.1 Rounding errors in collocation points when falling close to boundary 45

5.1 Relative L_2 errors, $k = 5, 10, 20, 40, 80$ and 160. $N = 2, 4, 8, 16, 32, 64,$ and 128 50

5.2 Timings, $k = 10, 20,$ and 40. $N = 2, 4, 8, 16, 32, 64,$ and 128 Collocation Vs Galerkin 52

5.3 Timings for increasing k, $N = 4$ and $N = 16$ 53

1. INTRODUCTION

Wave scattering spans a large range of physical scenarios which we can describe using *Partial Differential Equations* (PDE's), complemented with both boundary and initial data. These PDE's cover scenarios such as acoustics, electromagnetics, transmission lines (which store and convey energy), vibrations over some elastic membrane to name but a few, and have many practical applications in the modern world. Whether we require to know the acoustics in a music hall, and how the architect needs to design for optimality, or how we find oil fields and assess the size and quantity to see whether it is economically feasible for the oil to be removed, wave scattering has many implications everywhere. Therefore the numerical simulations of such challenges and the understanding of these simulations is a very important area of research.

This dissertation is to focus on time-harmonic acoustic scattering by sound soft convex polygons in an infinite plane. Using the boundary element method we can reformulate the problem as a problem on the boundary of the obstacle whose solution can be approximated using various methods. We shall also be focusing on high frequency waves where traditionally we require a larger number of degrees of freedom in order to obtain a good representation. Recent research has led to a new graded mesh being developed and used on the boundary of the obstacle instead of a more traditional uniform mesh where we discretise at regular intervals. The number of degrees of freedom required to achieve a given level of accuracy on this new graded mesh is logarithmically proportional to the wavenumber instead of directly proportional as we would see with the more traditional mesh. This means that simulation of high wavenumbers can be calculated a lot more efficiently than before. Recently, a Galerkin boundary element method (see [4]) with non-standard basis functions has been implemented successfully. Although it lends itself well to rigorous analysis it suffers from large computational costs for the problem.

The aim of this dissertation is develop a *Collocation Method* so that we can compute the entire acoustic wave field when given some known incident wave posed with a convex polygon scattering obstacle in the two dimensional plane. The vertices

of the obstacle in the isotropic two dimensional plane will be given and we will then numerically calculate the total wave field along the boundary of the object allowing us to construct an image of the total field. This method is more computationally efficient than Galerkin but does not lend itself to the same rigorous analysis. The method will be implemented using the *MATLAB* environment.

In chapter 2, where we discuss much of the theory, we give a brief introduction to the problem and discuss how we can reformulate it as an integral equation on the boundary, parametrise the problem and define the new graded mesh as in [4].

Chapter 3 is concerned with two of the methods we can use to solve the problem that currently exist, here, we give a brief overview of the Galerkin and collocation methods. We will then highlight their differences and discuss how we can choose the appropriate mathematical constraints in order to optimise our method, and eliminate some of the inaccuracies in our solution that may occur if a poor choice was made.

In chapter 4 we discuss the implementation of the collocation method in building the linear system we need to solve and how we can evaluate some of the entries containing oscillatory integrals analytically. We also discuss how to evaluate some of the oscillatory integrals which we cannot evaluate analytically and the reasons why we cannot do this. This section is primarily concerned with building the linear system defined by the method, but is also concerned with implementing the method in an efficient manner when numerically integrating and treating problems caused by the finite accuracy of machine hardware.

Numerical results produced by the method can be found in chapter 5, in this section we discuss these results and whether or not the method has proved to be successful for the problem and the reasons for this.

We conclude the report in chapter 6 and suggest avenues of investigation that have been brought to our attention throughout.

2. BACKGROUND INFORMATION

In this chapter, we present some of the background information and motivation for the problem at hand. This background review is not comprehensive but provides us with building blocks for the problem and so the theory here is pertinent to our specific application. Much of this background theory is based around the paper [4]. In this paper the same problem was tackled but instead of the collocation method we use here, a Galerkin method was implemented.

2.1 Wave Scattering

When investigating *scattering theory* we are concerned with what effect an obstacle has on some known incident wave (or waves). Scattering theory falls into two types of problem area, these are

- **Direct Scattering Problem** - Concerned with determining the scattered field in the plane where we have knowledge of the incident field and the scatterer. Here we refer to the scatterer as the scattering obstacle.

- **Inverse Scattering Problem** - Concerned with determining properties of the scatterer, for example, shape and physical attributes, from the measurement at specified positions of the scattered field, usually using a number of different incident fields. Ultrasounds are an example of an inverse scattering problem.

Our problem falls into the category of direct scattering, we now introduce the ideas behind developing the problem, as well as the problem itself.

2.2 Direct Scattering Problem

Along with defining the incident wave, the direct scattering problem consists of three requirements in order to solve for the total field when scattered by our convex polygon Ω. By total field we mean the sum of the incident wave and the waves

reflected and diffracted by the scatterer. These requirements consist of the following three subsections.

2.2.1 The Helmholtz Equation

If we consider the isotropic scalar field $V(\mathbf{x}, t) \in C^2(\mathbb{R}^m \backslash \Omega) \times C^2(0, T)$ for $m = 2, 3, \ldots$, where m is the dimension, then for a wave propagating with some speed $c > 0$ then V satisfies the wave equation

$$V_{tt} = c^2 \Delta V. \tag{2.1}$$

Since we are dealing with acoustic waves we look for a *time harmonic solution* to (2.1), that is, a solution which takes the form

$$V(\mathbf{x}, t) = u(\mathbf{x}) e^{-i\omega t},$$

where ω is the angular frequency. If we substitute this into (2.1) and perform some manipulation we obtain

$$\Delta u(\mathbf{x}) + k^2 u(\mathbf{x}) = 0, \tag{2.2}$$

for $k = \omega/c$, where k is known as the *wavenumber* and equation (2.2) is known as the *Helmholtz equation*. Note that the the wavenumber is directly proportional to the frequency of the wave and as the problem in question is specifically looking at high frequency waves, we are interested in the case when the value of k is large, or equivalently when the scattering obstacle is large. So we are specifically concerned with problems where the obstacle is many wavelengths long.

2.2.2 Boundary Condition on the Scatterer

We also require a boundary condition on the surface of the scatterer. This boundary condition can consist of Dirichlet, Neumann or Robin conditions. For acoustic waves these are known as sound soft, sound hard or impedance boundary conditions respectively. For the problem we are to consider the scatterer is to have a zero, sound soft (dirichlet) boundary condition and so we have

$$u = 0, \qquad \text{on } \Gamma, \tag{2.3}$$

where Γ represents the entire boundary of the polygon.

2.2.3 Sommerfeld Radiation Condition

The final requirement is that we are to supplement (2.2) with a *Sommerfeld Radiation Condition* (SRC). This ensures uniqueness of the solution for the total field. For example, one solution to (2.2) would clearly be $u = 0$, this of course is incorrect and so without the SRC we would not necessarily have a unique solution. The SRC takes the form

$$\lim_{r\to\infty} r^{\frac{m-1}{2}} \left(\frac{\partial u^s(\mathbf{x})}{\partial r} - iku^s(\mathbf{x}) \right) = 0, \qquad r = |x|, \tag{2.4}$$

where m is the dimension in which we are working ($m = 2$ in our problem) and $u^s(\mathbf{x})$ is defined to be the scattered field and is the difference between the total field $u(\mathbf{x})$ and the incident field $u^i(\mathbf{x})$ i.e. $u^s = u - u^i$. The SRC ensures the scattered field decays in all directions at ∞, and ensures the problem (2.2) together with the boundary condition (2.3) has a unique solution. Another way to think of this is that the scattered wave has a zero boundary condition at $r = \infty$.

We are able to reformulate the problem (2.2),(2.3) and (2.4) to be defined on the boundary of the object, this is known as a *Boundary Integral Equation* (BIE), and is what we are looking to solve. We describe this reformulation process in the following section.

2.3 Greens Representation and Reformulation

Equation (2.2) is an elliptic PDE and so we are able to represent the total acoustic field u, at any point in our domain explicitly in terms of the u and the normal derivative on the boundary of the polygon, Γ. So using Green's representation theorem, together with the zero Dirichlet boundary conditions on the polygon we have [5, Theorem 3.12]

$$u(\mathbf{x}) = u^i(\mathbf{x}) - \int_\Gamma \Phi(\mathbf{x},\mathbf{y}) \frac{\partial u}{\partial \mathbf{n}}(\mathbf{y})\, ds(\mathbf{y}), \qquad \mathbf{x} \in D, \tag{2.5}$$

where $\mathbf{n}$ is the normal to the boundary, $D := \mathbb{R}^2 \backslash \Omega$ and $\Phi(\mathbf{x},\mathbf{y}) := (i/4) H_0^{(1)}(k|\mathbf{x}-\mathbf{y}|)$ is the standard fundamental solution for the Helmholtz equation and $u^i(\mathbf{x})$ is the incident wave. We define the fundamental solution as follows (see for example [7, p.267] [11, pp.14-15]

Definition 1. For a linear PDE $Lu = 0$ in $\bar{\Omega} \subset \mathbb{R}^d$ the *fundamental solution* s (singularity function) is a function Φ of two variables $\mathbf{x}$ and $\mathbf{y}$, defined for all the

domain $\bar{\Omega}$, $\mathbf{x} \neq \mathbf{y}$, which for every fixed $y \in \Omega$ satisfies as a function of $\mathbf{x}$

$$L_x \Phi(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y})$$

where L_x denotes the operator differentiated with respect to $\mathbf{x}$, and δ is the dirac delta function.

We therefore have $u(\mathbf{x}) = \Phi(\mathbf{x}, \mathbf{y})$ satisfying the helmholtz equation and SRC everywhere except at the points $\mathbf{x} = \mathbf{y}$. The fundamental solution, Φ, consists of the function $H_0^{(1)}$, this is known as the Hankel function of the first kind, of order zero. Hankel functions are a linear combination of the Bessel and Neumann functions, more specifically

$$H_p^{(1,2)} = J_p(x) \pm i Y_p(x), \qquad p, x \in \mathbb{R}, \tag{2.6}$$

where $J_p(x)$ is the bessel function and $Y_p(x)$ the Neumann function. We will not investigate the Hankel function further here, the main features we are to take from it is that as the argument approaches zero from above then the absolute value of the Hankel function tends to infinity, and so it is not defined at the origin. Also, as we move away from the origin (in the positive direction) the magnitude of this function decays. Further information on these functions can be found in [16, pp.6-10] and [1, ch. 9].

We recall that we are looking to reformulate the problem (2.2),(2.3) and (2.4) as an integral equation on the boundary of the polygon, we let $\mathbf{x} \to \Gamma$ in (2.5) and using the zero dirichlet boundary conditions on Γ we obtain

$$u^i(\mathbf{x}) = \int_\Gamma \Phi(\mathbf{x}, \mathbf{y}) \frac{\partial u}{\partial \mathbf{n}}(\mathbf{y})\, ds(\mathbf{y}), \qquad \mathbf{x} \in \Gamma, \tag{2.7}$$

this is an integral equation of type *first kind*, but for some values of k no solution exists. Instead we compute the normal derivative of (2.5) and let $\mathbf{x} \to \Gamma$ once more to obtain

$$\frac{1}{2}\frac{\partial u}{\partial \mathbf{n}}(\mathbf{x}) + \int_\Gamma \frac{\partial \Phi(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}(\mathbf{x})} \frac{\partial u}{\partial \mathbf{n}}(\mathbf{y})\, ds(\mathbf{y}) = \frac{\partial u^i}{\partial \mathbf{n}}(\mathbf{x}), \qquad \mathbf{x} \in \Gamma, \tag{2.8}$$

where the half comes from the *jump relations* as we approach the boundary (see [5]). Equation (2.8) is an integral equation of type *second kind*, which although being easier to solve also suffers again as for some values of k no solution exists. To ensure that we obtain a solution for all k we take a hybrid of these two integral equations and add $i\eta\times$ (2.7) to (2.8) to obtain the *Burton/Miller equation* [3].

$$\frac{1}{2}\frac{\partial u}{\partial \mathbf{n}}(\mathbf{x}) + \int_\Gamma \left(\frac{\partial \Phi(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}(\mathbf{x})} + i\eta \Phi(\mathbf{x}, \mathbf{y}) \right) \frac{\partial u}{\partial \mathbf{n}}(\mathbf{y})\, ds(\mathbf{y}) = \tilde{f}(\mathbf{x}), \qquad \mathbf{x} \in \Gamma, \tag{2.9}$$

where we define the $\tilde{f}(\mathbf{x})$ to be

$$\tilde{f}(\mathbf{x}) := \frac{\partial u^i}{\partial \mathbf{n}} + i\eta u^i(\mathbf{x}) \tag{2.10}$$

with the coupling parameter $\eta \in \mathbb{R}\backslash\{0\}$. The choice of η can be dependent on many things, to balance (2.9) we choose $\eta = k$. By balance we mean that each of the terms in the Burton/Miller equation will be multiplied by the wavenumber k so it seems a logical choice and may also help to minimise the condition number in the system we solve. Note that this is not essential and not even necessarily the best choice, there are however many reports on what the best choice may be (see for example [2]) and it is a large area of research, we will not go into this further here.

Equation (2.9) is a uniquely solvable integral equation of type second kind which has a solution for all k. We are looking to solve (2.9) for the unknown $\partial u/\partial \mathbf{n}$ using some appropriate method and substitute this into (2.5) allowing us to compute the total acoustic field anywhere in our infinite domain. From here on the total acoustic field is to be known as $u^t(\mathbf{x})$, and as with the scattered and incident fields is for any point $\mathbf{x} \in D$. There are many numerical methods we can use to solve (2.9), two such methods are the *Galerkin method* and the *collocation method*, these are the two methods we shall focus our attention on throughout and we will return to the details of each in chapter 3.

The process of reformulating the Helmholtz equation on the boundary of the polygon and solving is known as the *Boundary Element Method* (BEM). Note that we have successfully taken the original two-dimensional problem (2.2), (2.3) and (2.4) and reduced to a one-dimensional problem on the boundary in (2.9), an obvious advantage, which we can then solve and hence find the total field. Before discussing the methods for solving further, we will talk about the effects of high frequency and how we can improve the efficiency of the standard BEM's to be computationally cheaper. This can be done by choosing an optimal mesh and using non-standard basis functions. We will also discuss why we need to do this as opposed to using a traditional mesh with nodes set at uniform intervals.

2.4 High Frequency

When numerically approximating waves using piecewise polynomials, typically as we see in figure 2.1, we require at least ten nodes per wavelength for a good estimation. It is clear to see that if we were solving for a wavelength of one unit length, then

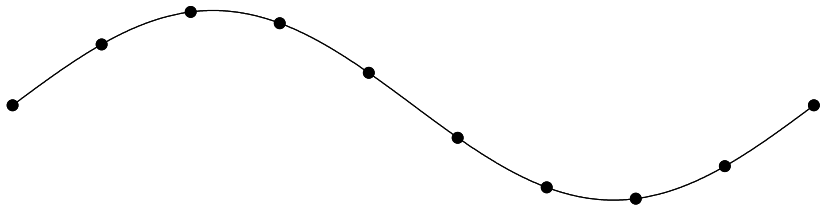

Fig. 2.1: Approximation to one complete wavelength

we would need, say j nodes to numerically represent the wave. However, if we were to consider solving a problem where the wave had length $1/100^{th}$ of a unit then we would need $100 \times j$ nodes to represent the wave over the same interval. This is of course not feasible in computational terms as $k \to \infty$, where we recall k is the wavenumber, and so we are looking to find a method to reduce the computational time considerably. We will preferably have a method where the degrees of freedom required over some interval are not in direct proportion to kL, where L is the length of the boundary.

If we consider for the moment some convex scatterer with a smooth boundary (i.e. not a polygon) one way we can solve for the entire acoustic field is remove some of the oscillation by factoring out the oscillation of the incident wave from the total field, thus

$$\frac{\partial u}{\partial \mathbf{n}}(\mathbf{y}) = \frac{\partial u^i}{\partial \mathbf{n}}(\mathbf{y}) \times F(y), \tag{2.11}$$

we are then able to approximate $F(y)$ using some conventional BEM (such as Galerkin) by piecewise polynomials of an appropriate degree. What we find is that (2.11) holds relatively well for $F(y) \approx 2$ on the illuminated sides of the obstacle and $F(y) \approx 0$ on the shadow sides. This makes sense if we consider physical optics, on the illuminated side we will get a reflection of the incident wave roughly equal to the incident wave itself. Likewise, on the shadow side we will get no incident wave and hence no reflection so the total wave will be roughly zero. As mentioned previously this is for a smooth object and helps reduce the degrees of freedom required per wavelength, this does not however work so well for polygons but we can use this idea of factoring out the oscillating wave to optimise our method. We shall return to this idea shortly.

2.5 The Problem On One Side

We can show by geometrical optics [8] that

$$u^t(\mathbf{x}) = u^i(\mathbf{x}) + u^r(\mathbf{x}) + u^d(\mathbf{x}), \tag{2.12}$$

that is, the total acoustic field is the sum of the incident, reflected and diffracted fields respectively. Here, $u^d(\mathbf{x})$ represents the diffracted field in the plane and $u^r(\mathbf{x})$ is the reflected field, the sum of these gives us the total scattered field, i.e. $u^s = u^d + u^r$. The same can also be said for their normal derivatives. Clearly the incident wave is known (of the form $u^i = e^{ik\mathbf{x}.\mathbf{d}} = e^{ik(x_1 d_1 + x_2 d_2)}$, where all terms are known) and when dealing with convex polygons it is a straightforward task to calculate the reflected wave. We give a brief overview of how we can do this here. In the absence of diffraction we have $u = u^i + u^r$ and so using the zero boundary condition (2.3) it follows that $u^r = -u^i$ on the boundary, and so assuming we have an infinite side of a polygon we seek $\tilde{d}$ in $u^r = e^{ik\mathbf{x}.\tilde{\mathbf{d}}}$ such that the reflected field equals the negative incident field subject to the zero boundary condition (i.e. $u^t = 0$ on x_2, where x_2 is the axis parallel to the polygon edge). This means that the only term we need focus our attention on for now is the term $u^d(\mathbf{x})$ concerning the diffracted field, since everything else is either known or easy to calculate.

If we consider one side of a polygon with corners P and Q (for now let this be an illuminated side) and let us extend the side of the polygon infinitely in our domain. We can define D_L to be the half-plane with the boundary $\Gamma_+ \cup \gamma \cup \Gamma_-$, we see this in figure 2.2 (see also [4, fig.2]).

We define the Dirichlet Green function in the half-plane D_L as

$$G(\mathbf{x}, \mathbf{y}) := \Phi(\mathbf{x}, \mathbf{y}) - \Phi\left(\mathbf{x}, \mathbf{y}'\right) \tag{2.13}$$

for $\mathbf{x}, \mathbf{y} \in D_L$. Here $\mathbf{y}'$ is the reflection of $\mathbf{y}$ along the boundary of our half-plane $\Gamma_+ \cup \gamma \cup \Gamma_-$ and (2.13) satisfies the helmholtz equation in D_L. If we consider (2.12) then in this case (when the side of the polygon is infinitely long) $u^i(\mathbf{x}) = e^{ik\mathbf{x}.\mathbf{d}}$ defines the incident field, whereas $u^r(\mathbf{x}) = -e^{ik\mathbf{x}.\mathbf{d'}}$ denotes the reflected field, where $\mathbf{d} = (d_1, d_2)$ is the direction of the wave. We define precisely what d_1 and d_2 are later on. Using Greens Representation Theorem [5] in the plane D_L we can represent the diffracted wave $u^d(\mathbf{x})$ as

$$u^d(\mathbf{x}) = \int_{\Gamma_+ \cup \gamma \cup \Gamma_-} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{n}(\mathbf{y})} u(\mathbf{y})\, ds(y), \qquad \mathbf{x} \in D, \tag{2.14}$$

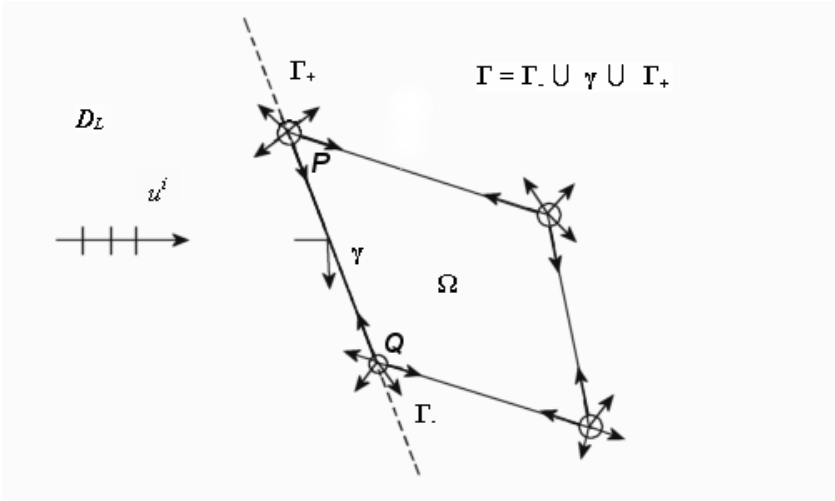

Fig. 2.2: Scattering on a half plane by a convex polygon, consisting of incident, reflected and diffracted waves

and since $u(\mathbf{y}) = 0$ on γ, using symmetry we can write

$$u^d(\mathbf{x}) = 2\int_{\Gamma_+\cup\Gamma_-} \frac{\partial\Phi(\mathbf{x},\mathbf{y})}{\partial\mathbf{n}(\mathbf{y})} u(\mathbf{y})\, ds(\mathbf{y}), \qquad \mathbf{x}\in\gamma, \tag{2.15}$$

we now take the normal derivative of (2.12) along the boundary and have for each side of the polygon

$$\frac{\partial u}{\partial\mathbf{n}}(\mathbf{x}) = 2\frac{\partial u^i}{\partial\mathbf{n}}(\mathbf{x}) + 2\int_{\Gamma_+\cup\Gamma_-} \frac{\partial^2\Phi(\mathbf{x},\mathbf{y})}{\partial\mathbf{n}(\mathbf{x})\,\partial\mathbf{n}(\mathbf{y})} u(\mathbf{y})\, ds(\mathbf{y}), \qquad \mathbf{x}\in\gamma. \tag{2.16}$$

Equation (2.16) is not *hypersingular* (not very singular) since at no point can $\mathbf{x} = \mathbf{y}$. We also note that the term $2\partial u^i/\partial\mathbf{n}(\mathbf{x})$ is absent on the shadow sides of the polygon.

We are looking to parametrise along γ and so represent the side as $P + s(Q - P)/|Q-P|, s \in [0, |P-Q|]$ where $L := |P-Q|$ is the length of γ and s is the distance along γ. Standard properties of bessel functions (see [1, ch.9]) allows us to show

$$\frac{\partial^2\Phi(\mathbf{x},\mathbf{y})}{\partial\mathbf{n}(\mathbf{x})\,\partial\mathbf{n}(\mathbf{x})} = \frac{ikH_1^{(1)}(k|\mathbf{x}-\mathbf{y}|)}{4|\mathbf{x}-\mathbf{y}|}, \tag{2.17}$$

so we can form an explicit parametric representation of each side for $s \in [0, L]$ as

$$\frac{\partial u}{\partial\mathbf{n}}(\mathbf{x}) = 2\frac{\partial u^i}{\partial\mathbf{n}} + \frac{ik^2}{2}\left[\int_{-\infty}^{0} H(k|s-s_0|)\,u(s_0)\,ds_0 + \int_{L}^{\infty} H(k|s-s_0|)\,u(s_0)\,ds_0\right] \tag{2.18}$$

where $H(z) := H_1^{(1)}/z, z > 0$. For simplicity and ease of read we make the following substitutions in terms of our parameter s.

$$\phi(s) := \frac{1}{k}\frac{\partial u}{\partial \mathbf{n}}, \qquad \phi^i(s) := \frac{1}{k}\frac{\partial u^i}{\partial \mathbf{n}}, \qquad \psi(s) := u, \tag{2.19}$$

where we have yet to explicitly identify what values $\phi(s)$ and $\phi^i(s)$ take, thus

$$\phi(s) = 2\phi^i(s) + \frac{i}{2}\left[e^{iks}\upsilon_+(s) + e^{-iks}\upsilon_-(s)\right], \tag{2.20}$$

where in (2.20) the normal derivative of the diffracted field $\partial u^d/\partial \mathbf{n}$ is represented by the last term with

$$\upsilon_+(s) = k\int_{-\infty}^{0} e^{-ik(s-s_0)}H(k(s-s_0))\,e^{-iks_0}\psi(s_0)\,ds_0,$$

and

$$\upsilon_-(s) = k\int_{L}^{\infty} e^{ik(s-s_0)}H(k(s-s_0))\,e^{iks_0}\psi(s_0)\,ds_0.$$

Putting

$$\mu(z) := e^{-iz}H(z) = e^{-iz}\frac{H_1^{(1)}(z)}{z}, \qquad z > 0,$$

as done in [4, p.5] we have

$$\upsilon_+(s) = k\int_{-\infty}^{0} \mu(k|s-t|)\,e^{-ikt}\psi(t)\,dt,$$

$$\upsilon_-(s) = k\int_{L}^{\infty} \mu(k|s-t|)\,e^{ikt}\psi(t)\,dt.$$

These integrals are over the Γ_+ and Γ_- and so can be thought as of the diffracted waves traveling along the side γ, which of course is what we are trying to numerically solve for. We remind ourselves that we wish to solve (2.9) and we have (2.20) as an explicit representation for its solution. By doing this in such a manner, it means we have now separated out the oscillatory and non-oscillatory parts of the solution in the sense that the μ terms are non-oscillatory, in that as $z \to \infty$, we have $\mu^{(m)} = O\left(z^{-3/2-m}\right)$ [4, p.5]. This gives us two important results about the diffracted wave for each υ.

1_+. $k^{-m}\left|\upsilon_+^{(m)}(s)\right| \le C_m(ks)^{-1/2-m}, \quad ks \ge 1,$

2_+. $k^{-m}\left|\upsilon_+^{(m)}(s)\right| \le C_m(ks)^{-\alpha-m}, \quad ks \le 1, \quad \alpha \in (0,1/2),$

1_-. $k^{-m}\left|\upsilon_+^{(m)}(L-s)\right| \le C_m(k(L-s))^{-1/2-m}, \quad k(L-s) \ge 1,$

$$2_-. \quad k^{-m}\left|v_+^{(m)}(L-s)\right| \leq C_m\left(k(L-s)\right)^{-\alpha-m}, \quad k(L-s)\leq 1, \quad \alpha\in(0,1/2),$$

where the constants C_m are independent of the wavenumber k. The first result for each tells us that if we are away from the corner, i.e. $ks > 1$ for v_+ and $k(L-s) > 1$ for v_-, then the magnitude of the $v_\pm$ is relatively small, and decreases the further from the corners we are, note that in the definition these correspond to opposite corners for the polygon edge. The second result tells us that as we get close to the corners, i.e. $ks < 1$ for v_+ and $k(L-s) < 1$ for v_-, then the diffracted wave becomes highly peaked and almost singular at the corner. This means that the $v_\pm$ are peaked at opposite corners along the edge of the polygon and decay as we tend away. This is useful to keep mind when we consider the mesh design, we shall return to this when we introduce the mesh to be used in section 2.7. Not only do the results above hold, but it is also dependent on the external angle of the corner ϖ. The value for α is defined as

$$\alpha := 1-\frac{\varpi}{\pi}, \qquad \text{where } \varpi\in(\pi,2\pi).$$

This definition in conjunction with our results about the $v_\pm$ suggests that we have more diffraction (so the wave is more highly peaked) where we have acute corners, this makes sense since if the external angle is equal to π, meaning no corner, then no diffraction will exist. If however we have just a point as the corner (angle equal to 2π) then we would expect more diffraction caused, which indeed it does and the results suggest this to be the case. Proofs of these results can be found in [4, Theorem 2.1, Theorem 2.2].

The terms $e^{\pm iks}$ in equation (2.20) provides much of the oscillation in the diffracted wave, this means we can approximate the $v_\pm$ using piecewise polynomials and multiply these by the oscillatory functions to get a good approximation, we incorporate this into what is known as the basis functions in the collocation method, we shall return to this choice in the chapter 3.

By using this information we can impose a graded mesh along the boundary of our polygon where the grading is dependent on the wavenumber and internal angle of the corner, in this mesh we define smaller elements near to the corner. If we consider figure 2.3 which denotes a speculative view on what we would expect the diffracted wave to look like where each end of the graph denotes a corner. At these corners due to the highly peaked (almost singular) nature dependent on wavenumber it is harder to represent what is going on using a few elements. We can however represent the central part (for instance from 0.2 to 0.8) using fewer elements (perhaps even one or two in this example) and get a fairly good representation of the

solution. By using this idea and deriving some rigorous regularity results it means a mesh can be designed with fewer degrees of freedom to obtain a more accurate result. The number of elements required will only increase logarithmically with kL as opposed to directly proportional as we saw before, a big improvement. In this report we will give a brief overview of the mesh points used (see section 2.7), but all major details have been left out.

We have considered one side of the polygon, it now remains to discuss how we

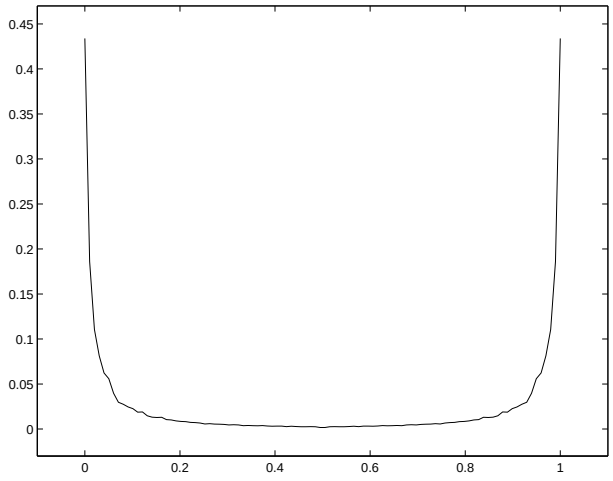

Fig. 2.3: Speculative view on what the diffracted wave may look like across one side of a polygon (of length one) with equal diffraction at each of the corners

fully parametrise the problem by carrying out similar observations over the whole boundary.

2.6 The Full Parametrised Problem

We wish to consider the boundary as a one dimensional problem instead of two dimensional as it stands now.

We can write the entire boundary of the polygon as the sum of each of the sides and order the sides so that we have the first n_s sides in shadow and the remaining sides illuminated. We call the entire boundary Γ and define it as $\Gamma = \bigcup_{j=1}^{n_s} \Gamma_j + \bigcup_{j=n_s+1}^{n} \Gamma_j$ for the n sides of the polygon, where Γ_j is the j^{th} side. We move around the polygon in an anticlockwise direction, and for our program define the first node to be the point $(0,0)$ in the plane. The j^{th} vertex is defined as $P_j = (p_j, q_j)$ and Γ_j connects P_j and P_{j+1}, as a result, for our parametrised boundary we have $P_1 = P_{n+1}$. The length of the j^{th} side is defined to be $L_j = |P_{j+1} - P_j|$, we can

then represent each edge of the polygon by

$$\Gamma_j = P_j + \tilde{s}\frac{P_{j+1} - P_j}{L_j}, \qquad \tilde{s} \in [0, L_j]\,.$$

Using this we can represent the whole boundary, Γ, parametrically as

$$\mathbf{x}(s) = P_j + \left(s - \sum_{l=1}^{j-1} L_l\right)\left(\frac{P_{j+1} - P_j}{L_j}\right), \qquad s \in \left(\sum_{l=1}^{j-1} L_l, \sum_{l=1}^{j} L_l\right) \qquad j = 1, \cdots, n$$

Let us briefly turn our attention to the incident wave u^i. We take the angle θ of the incident wave to be measured anticlockwise from the downward vertical and have $\theta \in [0, \pi/2]$, and write

$$u^i(\mathbf{x}) = e^{ik\mathbf{x}.\mathbf{d}} = e^{ik\left(x_1 \sin\theta - x_2 \cos\theta\right)}, \tag{2.21}$$

where $\mathbf{d}$ is the direction of the incident wave defined to be $(\sin\theta, -\cos\theta)$

Recalling the equation we wish to solve (2.9), our aim here is to represent the BIE in terms of some parameter s. We define the normal derivative to the edge Γ_j by $\mathbf{n}_j := (n_{j1}, n_{j2})$ then for $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$ and using properties of bessel functions [1] we have

$$\begin{aligned} G(\mathbf{x}, \mathbf{y}) &:= \left(\tfrac{\partial \Phi(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}(\mathbf{x})} + i\eta\Phi(\mathbf{x}, \mathbf{y})\right) \\ &= -\tfrac{1}{4}\left[\eta H_0^{(1)}\left(k\left|\mathbf{x} - \mathbf{y}\right|\right) + ik\tfrac{H_1^{(1)}(k|\mathbf{x}-\mathbf{y}|)}{|\mathbf{x}-\mathbf{y}|}\left[n_{j1}\left(x_1 - y_1\right) + n_{j2}\left(x_2 - y_2\right)\right]\right] \end{aligned} \tag{2.22}$$

If we now consider (2.10) (the right hand side of (2.9)), we can use (2.21) to obtain a new representation. We have for $\mathrm{x} \in \Gamma_j$

$$\begin{aligned} \tilde{f}(\mathbf{x}) &:= \tfrac{\partial u^i}{\partial \mathbf{n}}(\mathbf{x}) + i\eta u^i(\mathbf{x}), \\ &= \left[n_{j1} ik \sin\theta - n_{j2} ik \cos\theta + i\eta\right] e^{ik\left(x_1 \sin\theta - x_2 \cos\theta\right)}. \end{aligned} \tag{2.23}$$

We are able to use these previous results for fully parametrising (2.9) which is to be used when coding the method in *MATLAB*. By making the following substitutions

$$\mathbf{x} := \mathbf{x}(s), \qquad \mathbf{y} := \mathbf{y}(t), \qquad \phi(s) := \frac{1}{k}\frac{\partial u}{\partial n(\mathbf{x}(s))}$$

and using results discussed above we can write (2.9) as

$$\frac{1}{2}\phi(s) + \int_0^L G(\mathbf{x}(s), \mathbf{y}(t))\,\phi(t)\,dt = \frac{1}{k}\tilde{f}(\mathbf{x}(s)), \qquad s \in (0, L)$$

where $L = \sum_{l=1}^{n} L_l$ is the total length of the boundary. This equation can be broken down into a sum of integrals over each of the sides of our polygon, hence

$$\frac{1}{2}\phi(s) + \sum_{j=1}^{n} \int_{\sum_{l=1}^{j-1} L_l}^{\sum_{l=1}^{j} L_l} G(\mathbf{x}(s), \mathbf{y}(t)) \phi(t)\, dt = \frac{1}{k}\tilde{f}(\mathbf{x}(s)), \qquad s \in (0, L).$$

We now define for each of the j sides the following variables

$$a_j := \frac{p_{j+1} - p_j}{L_j}, \qquad b_j := \frac{q_{j+1} - q_j}{L_j},$$

$$c_j := p_j - a_j \sum_{l=1}^{j-1} L_l, \qquad d_j := q_j - b_j \sum_{l=1}^{j-1} L_l,$$

so that we have

$$x_1 - y_1 = a_m s - a_j t + c_m - c_j,$$
$$x_2 - y_2 = b_m s - b_j t + d_m - d_j,$$

for $\mathbf{x}(s) \in \Gamma_m$ and $\mathbf{y}(t) \in \Gamma_j$, where $m, j = 1, \ldots, n$.
We also note that

$$n_{j1} = b_j, \qquad n_{j2} = -a_j,$$

and can therefore define the absolute value of $|\mathbf{x} - \mathbf{y}|$ to be

$$|\mathbf{x} - \mathbf{y}| = \sqrt{(a_m s - a_j t + c_m - c_j)^2 + (b_m s - b_j t + d_m - d_j)^2} =: R(s,t) = R, \tag{2.24}$$

thus allowing us to write (2.22) in the parametrised form

$$\begin{aligned} G(\mathbf{x}, \mathbf{y}) &= -\tfrac{1}{4}\left[\eta H_0^{(1)}(kR) + \frac{ik((a_m b_j - b_m a_j)t + b_m(c_m - c_j) - a_m(d_m - d_j))H_1^{(1)}(kR)}{R}\right] \\ &=: K(s,t). \end{aligned} \tag{2.25}$$

Similarly, the right hand side (2.23) can be represented parametrically as

$$\frac{1}{k}\tilde{f}(\mathbf{x}(s)) = i\left[b_m \sin\theta + a_m \cos\theta + \left(\frac{\eta}{k}\right)\right] e^{ik\left((a_m s + c_m)\sin\theta - (b_m s + d_m)\cos\theta\right)} =: f(s). \tag{2.26}$$

This gives us the full parametrised equation for the *Burton-Miller* formulation (2.9) which we are looking to solve as

$$\phi(s) + 2\int_0^L K(s,t)\phi(t)\, dt = 2f(s). \tag{2.27}$$

If we recall (2.20) we were able to separate off the leading order behavior of the wave field consisting of the incident and reflected waves leaving us with just the

diffracted wave field. We do the same procedure here, so we define the following function to represent this

$$\Upsilon(s) := \begin{cases} 2\phi^i(s), & \text{if } m > n_s, \\ 0 & \text{if } m \leq n_s, \end{cases} \tag{2.28}$$

for $s \in \left(\sum_{l=1}^{i-1} L_l, \sum_{l=1}^{i} L_l\right)$, where m runs from 1 through n, n_s represents the sides in shadow (remembering we order the sides) and $\phi^i(s)$ is the leading order behavior as in (2.20). The $\phi^i(s)$ also has a parametric representation and is equal to

$$\phi^i(s) := \frac{1}{k}\frac{\partial u^i}{\partial \mathbf{n}}(\mathbf{x}(s)) = i(b_m \sin\theta + a_m \cos\theta) e^{ik\left[(a_m s + c_m)\sin\theta - (b_m s + d_m)\cos\theta\right]}.$$

We are interested in the diffracted wave and so wish to set up a parametrised integral equation to represent only this, we wish to do this by first defining

$$\varphi(s) := \phi(s) - \Upsilon(s),$$

so that we can substitute this into (2.27) to obtain

$$\varphi(s) + 2\int_0^L K(s,t)\varphi(t)\,dt = 2f(s) - \Upsilon(s) - 2\int_0^L K(s,t)\Upsilon(t)\,dt := F(s), \tag{2.29}$$

and so obtain our parametrised integral equation which we require to solve for as

$$\varphi(s) + \kappa\varphi(s) = F(s), \qquad s \in [0, L],$$

where

$$\kappa\varphi(s) := 2\int_0^L K(s,t)\varphi(t)\,dt.$$

We have our integral equation in which we must approximate for the diffracted field. The rest of the total field consisting of the incident and reflected waves can be calculated easily since we have the explicit form for u^i, and u^r can be found easily as described previously. It now remains to introduce the mesh we use on each side of the polygon to optimise the method by reducing the degrees of freedom needed.

2.7 The Mesh

We are looking to approximate $e^{iks}v_+ + e^{-iks}v_-$. We do this by incorporating the functions $e^{\pm iks}$ into our basis functions (see section 3.2.1), which allows us to approximate the non-oscillatory $v_\pm$ by piecewise polynomials of some degree $\nu \geq 0$ to be chosen. This is possible since what we are approximating is smooth,

and so we should be able to get a good numerical approximation to the $\upsilon_{\pm}$. We approximate on a graded mesh with a higher grading near the corners as stated already, this mesh grading is dependent on the wavenumber and the internal angles of each of the corners. We can expect the diffraction to be highly peaked on more acute corners and so will require a higher mesh grading for this instance, likewise the mesh grading is also to be dependent on the wavenumber since the diffraction wave will be larger for higher values of k and so need a higher grading. We use a composite mesh on $[0, A]$ where $A > \lambda := 1/k$ with a polynomial grading on $[0, \lambda]$ and a geometric grading on $[\lambda, A]$. The mesh used here is the same as in [4, p.15],[12, p.4] and similar to that defined in [14, pp.16-22] and defined as

Definition 2. For N = 2,3,... and for $A > \lambda$ the mesh $\Lambda_{N,A,\lambda,\upsilon,\alpha} := \{y_0, \ldots, y_{N+N_A}\}$ consists of the points

$$y_i = \lambda \left(\frac{i}{N}\right)^q, \qquad i = 0, \ldots, N,$$

where $q := (2\nu + 3) / (1 - 2\alpha)$, with $\alpha := 1 - \frac{\varpi}{\pi}$, along with the points

$$y_{N+j} := \lambda \left(\frac{A}{\lambda}\right)^{j/N_A}, j = 1, \ldots, N_A,$$

where $N_A = \lceil N^* \rceil$ is the smallest integer greater than or equal to N^*, where

$$N^* = \frac{-\log(A/\lambda)}{q \log(1 - 1/N)}.$$

This choice of N^* ensures a smooth transition between the polynomial and geometric meshes.

This definition describes one of our grids we need to set up. We recall that we have the υ_+ and υ_- from section 2.5, each is peaked over the sides of each edge of the polygon towards opposite corners. The definition above will denote the grid on the side of the polygon $[0, A]$ for υ_+ since we have a higher grading towards the zero corner for that edge. For υ_- we need another grid, this time it will be graded towards the A corner of the side due to the diffraction being highly peaked at the opposite end. The process of computing is exactly the same for this second grid where we can either compute the nodes directly or in the same way as above (i.e. assuming the corner is at the zero end) and reflecting the points. Figure 2.4 represents what each mesh calculated will look like. We have two meshes for each side and although they are both used in calculating the behavior of the wave, in practice we consider them separately. This process of course needs to be done for

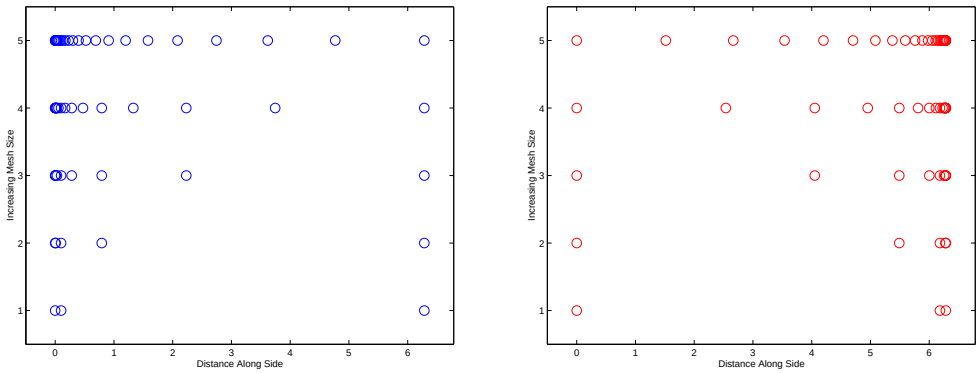

Fig. 2.4: Mesh points for one side of square of length 2π, wavenumber $k = 10$ for $N = 2, 4, 8, 16, 32$. *Left*: mesh for v_+. *Right*: mesh for v_-

each side of the polygon.

This mesh is only logarithmically proportional to kL and so doubling k give us a logarithmic increase in the degrees of freedom, hence it is especially useful for high frequency problems. In this chapter we have discussed some of the theory applicable to the high frequency scattering problems, but have yet to discuss any of the methods we use. In the next chapter we describe two such methods for solving the problem.

3. BOUNDARY ELEMENT METHODS

We now give a description of the *Galerkin* and *collocation* methods and a brief comparison between the two. We highlight their advantages and disadvantages and the reasons why this further work into the implementation of the collocation method is a logical step forwards from the Galerkin method.

If we take the general form of the integral equation of the second kind we are required to solve, we have

$$\phi + K\phi = f, \tag{3.1}$$

where ϕ is the unknown to solve for, K is some known integral operator and f some known right hand side function. We look to approximate ϕ by a sum of unknown constants and known basis functions. We define our approximation to be $\phi_N(s)$ and this takes the form

$$\phi_N(s) = \sum_{j=1}^{N} c_j \rho_j(s), \tag{3.2}$$

where we look to solve for the unknowns c_j and choose the basis functions $\rho_j(s)$ in some appropriate manner. The above information is generic to both methods we are to discuss, from here they differ and so we will tackle them individually.

3.1 Galerkin Method

Before we formally define the Galerkin method let us first introduce the notation of inner products

$$(v, w) := \int_{\Gamma} v\bar{w} d\Gamma,$$

which has the corresponding L_2-norm defined as $\|v\|_{L_2} := (v, v)^{\frac{1}{2}}$.

We proceed in the Galerkin method by multiplying (3.1) by some test function (call this χ) and integrating. (3.1) becomes what is known as *The Weak Formulation*, defined to be

$$(\phi, \chi) + (K\phi, \chi) = (f, \chi). \tag{3.3}$$

The Galerkin method says that we require for this to hold $\forall \chi \in \{V_{N_G}\}$ where V_{N_G} is our approximation space. Let $\phi_{N_G} = \phi_N$ be the Galerkin approximation to ϕ. We

seek to approximate $\phi_{N_G} \in V_{N_G}$ and we choose $V_{N_G} = \text{span}\ \{\rho_j\}$, $j = 1, \ldots, N_G$ i.e. we choose the χ to be equal to the ρ_j, the set of basis functions we use to approximate ϕ by. If we substitute this into (3.3) along with the approximation to ϕ, we have

$$\sum_{j=1}^{N_G} c_j \left[(\rho_j, \rho_m) + (K\rho_j, \rho_m)\right] = (f, \rho_m), \qquad \text{for } m = 1, 2, \ldots, N_G.$$

Using this we are able to form a linear system in order to solve and find the unknowns in question. The values of m and j each run up to N_G, where N_G is the number of elements (basis functions) over the interval we wish solve for. As a result, the system we solve is of the form

$$\begin{bmatrix} (\rho_1,\rho_1)+(K\rho_1,\rho_1) & (\rho_2,\rho_1)+(K\rho_2,\rho_1) & \cdots & \cdots & \cdots \\ (\rho_1,\rho_2)+(K\rho_1,\rho_2) & (\rho_2,\rho_2)+(K\rho_2,\rho_2) & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & & \\ \vdots & \vdots & & \ddots & \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} (f_1,\rho_1) \\ (f_2,\rho_2) \\ \vdots \\ \vdots \end{bmatrix} \tag{3.4}$$

We are required to find the unknowns in the linear system (3.4) and using (3.2) we obtain a representation for the approximation to ϕ. Solving this system can be done very easily in *MATLAB* using the simple matrix division function. However, the Galerkin method does suffer due to the initial computational cost of setting up the matrix. The general entry in the matrix contains a single integral summed with a term of the form

$$(K\rho_j, \rho_m) = \int_{\text{supp}\rho_m} \int_{\text{supp}\rho_j} \left(\frac{\partial \Phi}{\partial \mathbf{n}} + i\eta\Phi\right) \rho_j(s)\, \rho_m(t)\, ds dt.$$

This is a double integral over the support of each of the basis functions (the ρ's) of an oscillatory function since the term $(\partial\Phi/\partial\mathbf{n} + i\eta\Phi)$ is oscillatory as are the basis functions ρ_j and ρ_m. The major disadvantage here is because we have a double integral then the computational cost is very high, we recall that for each wavelength we need a minimum of ten nodes, as a result we need a lot of nodes in two directions for a good representation and this proves to be very slow computationally. For these integrals if we wish to improve the accuracy by doubling the amount of nodes then the time to evaluate will increase four times and so something faster is desirable. For example, a method using at most one dimensional integrals, where doubling nodes over each integral will only double the time taken since the nodes will only need doubling in one direction.

The next question is, if it is less computationally efficient than other methods, then why do we use the Galerkin method? The answer is that the method is a stable method and it converges to the true solution so we know we will obtain a good result. Not only is this true, but also we can prove that this is the case, we do not see this type of analysis for some other methods, for example, the collocation method. We will now discuss how the collocation method works, and why we are attempting to use it in this dissertation to tackle the problem.

3.2 Collocation Method

In the collocation method we do not multiply (3.1) by a test function, and simply replace ϕ by our approximation (3.2) in (3.1). For the collocation method our approximation will be known as $\phi_{N_c} = \phi_N$, thus

$$\sum_{j=1}^{N_c} c_j \left[\rho_j + K\rho_j\right] = f, \qquad \text{for } m = 1, 2, \ldots, N_c. \tag{3.5}$$

The method works by forcing (3.5) to be true for N_c predefined collocation points $x_1, x_2, \cdots, x_{N_c}$, i.e.

$$\sum_{j=1}^{N_c} c_j \left[\rho_j (x_m) + K\rho_j (x_m)\right] = f (x_m), \qquad \text{for } m = 1, 2, \ldots, N_c.$$

As in the Galerkin method we form a linear system in which we require to solve for our unknowns. The system takes the general form

$$\begin{bmatrix} \rho_1 (x_1) + K\rho_1 (x_1) & \rho_2 (x_1) + K\rho_2 (x_1) & \cdots & \cdots & \cdots \\ \rho_1 (x_2) + K\rho_1 (x_2) & \rho_2 (x_2) + K\rho_2 (x_2) & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & & \\ \vdots & \vdots & & \ddots & \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1 (x_1) \\ f_2 (x_2) \\ \vdots \\ \vdots \end{bmatrix} \tag{3.6}$$

Clearly this system is more computationally efficient than we had before. The terms in the matrix have now been reduced to a simple function evaluation and a single integral. Although in the current form for our problem the integrals are still oscillatory, it is clear that this system will be calculated considerably faster than what we had with the Galerkin method and is the key advantage in using this method as no double integrals are involved. One might ask why do we not start with this method and forget about Galerkin. The reason for this is that the collocation method cannot be analysed in the same way as the Galerkin, by this,

we mean we cannot prove the stability or the convergence of the method and so it is not a good starting point when investigating a new idea. What we can and will do is obtain results using Galerkin and use this as a comparison (for example using the L_2-error) to investigate whether or not the collocation method converges in the same way. If we do have convergence that is not dependent on the wavenumber then we have a successful method. It now remains to decide on the basis functions we must use and the implications of the implementation phase.

In this chapter, so far, we have looked at two different boundary element methods, both of which require the use of basis functions to be determined. To conclude this section we apply this to our problem and choose the basis functions in order to ensure we have an accurate solution, i.e. we do not have an ill-conditioned system. The optimal choice for the basis functions will depend on the method we use, for that reason, we focus on the collocation method basis functions and how we determine them. We will also discuss how we choose our collocation points for the problem.

3.2.1 Choosing the Basis Functions (Collocation Method)

First, let us recall what we are trying to approximate in this problem. That is, the approximation to the diffracted field in (2.20), which takes the form

$$\left[e^{iks}v_+(s) + e^{-iks}v_-(s)\right].$$

We mentioned in the chapter 2 that we remove lots of the oscillations of the diffraction and are left with the $v_\pm$ functions which are non-oscillatory. We also discussed that we approximate using piecewise polynomials, in our case we shall use degree 0 (piecewise constants).

Due to the nature of the diffracted wave it seems logical to let the basis functions be some multiple of $e^{\pm iks}$. We can then multiply these basis functions by our piecewise polynomial approximation to $v_\pm$ to get the diffracted field, these are known as plane wave basis functions (since they take the form of a plane wave, see [13, ch. 2]) and are the non-standard basis functions we use. This makes sense since we are approximating the non-oscillatory functions $v_\pm$ and can get a lot better representation with piecewise constants using fewer degrees of freedom than if the functions were heavily oscillating. We mentioned how we approximate the unknown using (3.2) and because we are approximating $v_\pm$ which is multiplied by $e^{\pm iks}$ we

are effectively trying to find an approximation to ϕ in the form

$$\phi_{N_c}(s) = \sum_{j=1}^{N_1} c_j^+ e^{iks} + \sum_{j=1}^{N_2} c_j^- e^{-iks}.$$

So we have N_1+N_2 $(= N_c)$ unknowns to solve for. This also means that we will have a combination of two meshes over the boundary of the polygon (as we discussed in section 2.7). One will represent the basis functions concerning the e^{iks} with the mesh graded towards one of the corners and few nodes elsewhere, the other will be for e^{-iks} basis functions which will be graded towards the opposite corner. From here we will refer to these as *gridx* and *gridy* respectively and are as we saw in figure 2.4.

We mentioned the condition number earlier and how wish to minimise this, the condition number describes how sensitive the system is to errors in the vector when solving the system and so the smaller the better. Theory of integral equations means that if we have a second kind integral equation that takes the form $(I + K)\phi = f$, where I is the identity, then provided the integral operator K satisfies $\|K\| < 1$ then we have a well conditioned system, we can also show that $(I + K)^{-1}$ exists and is bounded [14, Theorem. 4.1] and so a solution exists. For this problem it is unclear whether or not this condition on K is satisfied, however, generally this is still better than the standard $L\phi = f$ form. The integral equation we are looking to solve is similar to this $(I + K)\phi = f$ form and looks like

$$[\rho_j(s) + K\rho_j(s)]\, c_m = f(s), \tag{3.7}$$

for the unknowns c_m. This means that if we make the term $\rho(s)$ as close to the identity as possible then we should be able to obtain a better conditioned system. The easiest way to do this, due to the sparsity of the matrix, is to have the leading diagonal entries equal to one and attempt to minimise any other entries we might have, therefore we are to choose the basis functions to be

$$\rho_j(s) = e^{\sigma_j ik(s-z)} \chi[y_j, y_{j+1}] \tag{3.8}$$

where

$\sigma_j = \pm 1$, depending if we are on gridx or gridy

y_j are the points of the graded mesh

$\chi[y_j, y_{j+1}]$ is the characteristic function $= \begin{cases} 1 & s \in (y_j, y_{j+1}) \\ 0 & \text{otherwise} \end{cases}$

and

$$z = \frac{y_j + y_{j+1}}{2}. \tag{3.9}$$

This choice of basis function covers both cases for approximating $v_{\pm}$, one of which is covered by $\sigma_j = 1$ and the other $\sigma_j = -1$. The z values are the mid-points of our intervals and the s are the collocation points. We have not yet formally identified what we choose to be the collocation points, there are many different ways we can choose and so shall discuss this now.

3.2.2 Choosing the Collocation Points

Recall the method works by collocating the selected points $x_1, \ldots, x_{N_c}$ so that the integral equation (3.7) holds (i.e. forcing the equation to hold at these points). We need to select the points so that we have one collocation point for each interval in our mesh. [15] discusses many ways we are able to make this choice on a uniform mesh and also talks about a general rule we follow. The rule of thumb states that if we are approximating using piecewise linears or cubic splines then the collocation points should be the break points of the mesh i.e. at $x_m = k/m$ intervals. If however, we are using piecewise constants, then we should use the mid-points of the intervals, there is also the option to use ϵ-collocation giving greater freedom of choice where the ϵ allows a small perturbation along each interval. ϵ-collocation is defined as

$$x_m = \frac{k + \epsilon}{m}, \qquad m = 1, \ldots, (N_1 + N_2).$$

If we were follow the rule above for the piecewise constants we choose the mid-points to be our collocation points, i.e. the value of $\epsilon = 1/2$. For uniform meshes approximating by piecewise constants, this choice of ϵ will give an order of convergence of at least one better than using some other point in the interval. Our problem differs since the mesh we are to use is non-uniform and so this rule does not directly apply. Currently, no known rule exists for a non-uniform mesh in choosing these points, however, we are still to choose the collocation points to be the mid-points of the intervals. Although we have no analysis to support our decision, this seems the most logical choice from what is known, and so seems the most obvious starting point when implementing the method.

The z values (3.9) in the basis functions are the mid-points of our intervals, the collocation points are also contained within the intervals (as one would expect), this can however cause one problem. If, in the unlikely event we have a collocation point and a mid-point (one on gridx and one on gridy) that exactly coincide

with each other, then two rows of the matrix in the linear system will be the same, this will mean that the system is singular which is clearly bad news as no solution will exist. This situation is unlikely, and in the short term we need not to worry about this. However, if the software was required for commercial use this would be an issue and there are many ways around it. For example, it is possible to have two separate collocation points on each interval and tackle the problem that way, although possible, it is not ideal. We could also formulate the mesh differently, by this we mean divide the side of the polygon into two. In the first half where the v_+ is peaked we could have a mesh graded towards the corner and no elements crossing the central point, and in the other half where the v_- approximation is peaked we could have another graded mesh. This will mean that the meshes will not overlap and we would guarantee that a mid-point on one mesh will never equal the collocation point from the other, hence the singular matrix scenario will be avoided and a solution will always exist. We will not pursue this any further in this report, but it is something to think about when developing the software further.

4. COLLOCATION METHOD IMPLEMENTATION

In this chapter we give an overview of the implementation implications, and how for different basis functions and collocation points we approach the problem differently. We recall the integral equation we wish to solve takes the form

$$\varphi(s) + \kappa\varphi(s) = F(s), \qquad s \in [0, L]. \tag{4.1}$$

where

$$\kappa\varphi(s) := 2\int_0^L K(s,t)\,\varphi(t)\,dt,$$

and s is some parameter along the boundary.
We also recall that we look to approximate our unknown φ using the collocation method and the series approximation

$$\varphi_{N_c}(s) = \sum_{j=1}^{N_c} c_j \rho_j(s) \tag{4.2}$$

where we define the ρ_j to be the basis functions (3.8).

It is also worth reiterating here that the collocation points we are to use are to be the central points of each of the intervals within the mesh, that is, they are the central points of each of our basis function intervals. This means we shall have a set of collocation points for gridx and also a set for gridy. We define the entire set of collocation points to be s_m for $m = 1, \ldots, N_c$. The method states that we take (4.2) and substitute it into (4.1) and so

$$\sum_{j=1}^{N_c} c_j \rho_j(s) + 2\int_0^L K(s,t) \sum_{j=1}^{N_c} c_j \rho_j(t)\,dt = F(s),$$

thus

$$\sum_{j=1}^{N_c} \left[\rho_j(s) + 2\int_0^L K(s,t)\,\rho_j(t)\,dt\right] c_j = F(s), \tag{4.3}$$

where we wish to solve for the unknowns c_j and (4.3) holds for all $s = s_m$.

When implementing this collocation method within the *MATLAB* environment

we can break (4.3) into three separate problems when building the system, these take the form

$$\text{LHS1} = \rho_j(s_m), \tag{4.4}$$

$$\text{LHS2} = \int_0^L K(s_m, t)\,\rho_j(t)\,dt, \tag{4.5}$$

$$\text{RHS} = F(s_m), \tag{4.6}$$

where all functions have been defined previously and $j, m = 1, \ldots, N_c$. We will now look at each case separately and discuss how we tackle the problem of evaluating each term in *MATLAB*.

4.1 LHS1

Firstly, we shall look at (4.4). This of course is a matrix of size $N_c \times N_c$ where N_c is the amount of basis functions/collocation points we have. The entries of LHS1 are a simple function evaluation for each function (3.8) with the appropriate collocation point as the argument. If we let ρ_j represent the j^{th} basis function and s_m represent the m^{th} collocation point, then if ρ_j and s_m are on different sides of the polygon then (3.8) is zero immediately by definition. Similarly, if the collocation point falls outside the support of the basis function when ρ_j and s_m are on the same side then we have zero immediately.

If ρ_j and s_m are on the same side of the polygon we have two cases to consider. Firstly, when $\sigma_j = \sigma_m$, that is, when both the basis function and the collocation point are supported on the same grid (either gridx or gridy) and the collocation point coincides with the basis function. Clearly, in this case the entry we are calculating shall fall on the diagonal of the matrix. By taking the collocation point to be the mid-point of the basis function means that we have $s_m = z$ and the basis function evaluation is equal to one since

$$\rho_j(s_m) = e^{\sigma_j ik(s_m - z)} = e^{\sigma_j ik \times 0} = e^0 = 1$$

The second case we look at is when ρ_j and s_m are on supported on different grids, this time we have $\sigma_j \neq \sigma_m$ (one is supported on gridx and the other on gridy). In this case we have a simple evaluation of the basis function with the appropriate collocation point as the argument.

LHS1 is the simplest case we shall come across when setting up the system (3.6) as no integration is required and so few entries are needed to be evaluated. In fact, the matrix will only have $2 \times N_c$ non-zero entries and so as a result for

memory reasons, when we implement LHS1 using *MATLAB* we shall use *sparse matrix storage* format. We will now look at the more complicated LHS2 and the implementation ramifications we need to take into account.

4.2 LHS2

If we recall our parametrised integral operator (2.25) for the problem, bringing the $-\frac{1}{4}$ constant outside this definition for K and using the basis functions (3.8), we can write (4.5) as

$$-\frac{1}{4}\frac{1}{e^{ikz}}\int_{y_j}^{y_{j+1}} K\left(s_m,t\right)\times e^{\sigma_j ikt}dt \tag{4.7}$$

where

$$K\left(s_m,t\right)=\left[\eta H_0^{(1)}\left(kR\right)+\frac{ik((a_mb_j-b_ma_j)t+b_m(c_m-c_j)-a_m(d_m-d_j))H_1^{(1)}(kR)}{R}\right]$$

with $R=R\left(s_m,t\right)$. Let us also note that the m-subscripted values represent those defined by the side in which the collocation point lies and the j-subscripted values represent those corresponding to the side where the basis function sits. The matrix formed by this is of size $N_c\times N_c$ as we saw for LHS1. In computing the entries of this matrix we have two possibilities, these are as follows

4.2.1 Case (i)

The first case is when the collocation point lies on the same side as the support of the basis function. We find that when the collocation point falls inside or close to the basis function then the larger the entry in the matrix is, almost singular in the first instance. As a result, and for a more efficient and accurate result we need to make a number of substitutions and calculations on the K function. In this case the entries represented fall in the leading block diagonals of the matrix. It is easy to show that when they both lie on the same side that

$$K\left(s_m,t\right)=\eta H_0^{(1)}\left(k\left|s_m-t\right|\right), \tag{4.8}$$

and using the identity [4, p.20]

$$H_0^{(1)}\left(s\right)=-\frac{2i}{\pi}\int_0^{\infty}\frac{e^{(i-t)s}}{t^{\frac{1}{2}}\left(t-2i\right)^{\frac{1}{2}}}dt,\qquad s>0, \tag{4.9}$$

we can write (4.7) as

$$-\frac{1}{4}\frac{1}{e^{ikz}}\int_{y_j}^{y_{j+1}}\eta H_0^{(1)}\left(k\left|s_m-t\right|\right)e^{\sigma_j ikt}dt,$$

substituting in (4.9), we switch the order of integration and obtain a new form for the entries of the matrix for this particular case, hence, we wish to evaluate

$$\frac{i\eta}{2\pi e^{ikz}}\int_0^\infty\left[\int_{y_j}^{y_{j+1}}\frac{e^{(i-r)k|s_m-t|}e^{\sigma_j ikt}}{r^{\frac{1}{2}}\left(r-2i\right)^{\frac{1}{2}}}dt\right]dr. \tag{4.10}$$

We are able to calculate the inner integral analytically and so using some numerical integration rule we need to evaluate the outer integral. (4.10) takes the general form (ignoring the constant)

$$\int_0^\infty\frac{I\left(r\right)}{r^{\frac{1}{2}}\left(r-2i\right)^{\frac{1}{2}}}dr,$$

for some function $I\left(r\right)$. In order to solve this using quadrature we make the substitution $r=s^2/\left(1-s^2\right)$, this reduces the interval of integration to $[0,1]$ and eliminates the singularity at $r=0$. It now remains to consider the procedure for calculating the analytic integrals $I\left(r\right)$. This is broken into three further categories depending on the positioning of the s_m in relation to ρ_j, all of which have a slightly different result. These are, when the collocation point is smaller than the basis function, the collocation point is larger than the basis function and finally when the collocation point falls inside the support of the basis function. These three cases will be known from here as $I1$, $I2$ and $I3$ respectively. We define these as the following where the y_j and y_{j+1} are the end points of the j^{th} basis function,

$I1$: $s_m<y_j$,

$$\int_{y_j}^{y_{j+1}}e^{(i-r)k(t-s_m)}e^{\sigma_j ikt}dt$$
$$=\frac{e^{k(r-i)s_m}\left(e^{-ky_j(r-i(1+\sigma_j))}-e^{-ky_{j+1}(r-i(1+\sigma_j))}\right)}{k\left(r-i\left(1+\sigma_j\right)\right)}, \tag{4.11}$$

$I2$: $s_m>y_{j+1}$,

$$\int_{y_j}^{y_{j+1}}e^{(i-r)k(s_m-t)}e^{\sigma_j ikt}dt$$
$$=\frac{e^{-k(r-i)s_m}\left(-e^{ky_j(r+i(\sigma_j-1))}+e^{ky_{j+1}(r+i(\sigma_j-1))}\right)}{k\left(r+i\left(\sigma_j-1\right)\right)}, \tag{4.12}$$

and finally $I3$: $y_j<s_m<y_{j+1}$,

$$\int_{y_j}^{s_m}e^{(i-r)k(s_m-t)}e^{\sigma_j ikt}dt+\int_{s_m}^{y_{j+1}}e^{(i-r)k(t-s_m)}e^{\sigma_j ikt}dt$$

$$= \frac{e^{kis_m\sigma_j} - e^{rk(y_j - s_m) + ik(s_m + y_j(\sigma_j - 1))}}{ik\left((\sigma_j - 1)\right)} + \frac{e^{iks_m\sigma_j} - e^{rk(s_m - y_{j+1}) + ik(y_{j+1}(1+\sigma_j) - s_m)}}{ik\left(r - (1 + \sigma_j)\right)}. \tag{4.13}$$

By doing this, we have successfully reduced the original integral to be solved into an integral which does not oscillate by solving the oscillatory part analytically, this allows us to use a smaller amount of quadrature points to get a more efficient and accurate approximation to the function. It also means we avoid the function being singular (or almost very highly peaked) when the collocation point is close to or sits in the support of a basis function.

4.2.2 Case (ii)

The second case we consider is when the basis function is supported on a different side from the collocation point. These represent the off block diagonals in the matrix LHS2 i.e. the rest of the entries. In this situation we can calculate nothing analytically and so have to use quadrature to approximate $K\left(s_m, t\right)$, where the form of $K\left(s_m, t\right)$ is demonstrated by (4.7). We also note that once again if the collocation point and the basis function are close to each other we would expect a large peak in the evaluation of (4.7) and special allowances need to be made. This will occur when we are dealing with adjacent corners of the polygon (since for *case (ii)* s_m and ρ_j are on different sides). We return to what allowances we need to make in section 4.4.

4.3 RHS

The third separate problem we need to consider is the N_c-column vector on the right hand side of our linear system. The evaluation of the vector comes from the terms

$$F\left(s_m\right) := 2f\left(s_m\right) - \Upsilon\left(s_m\right) - 2\int_0^L K\left(s_m, t\right)\Upsilon\left(t\right)dt \tag{4.14}$$

as defined by (2.29). As we did for LHS2, we consider (4.14) in separate cases (two in this case). One case when we are on a shadow side and the second case when we are on an illuminated side of the polygon. The first case we consider are the shadow sides, this represents the first n_s sides of the polygon.

4.3.1 RHS - Shadow Sides

When we are considering a collocation point on a shadow side then the $\Upsilon\left(s_m\right)$ term in (4.14) vanishes as does much of the integration since not all of $[0, L]$ is

illuminated. This leaves us with a simple evaluation for $f(s_m)$, defined by (2.26), (where s_m will be the collocation point in question) plus some small contribution from the integral which we can evaluate using quadrature.

Similar to LHS2 we find that when a collocation point is close to the support of the integral in the last term of (4.14), then the function is highly peaked (almost singular when the collocation point is contained within the support). We note that when this integral is on a shadow side, it is clearly zero, since $\Upsilon(t)$ is zero. As a result we need not make any special allowances for the shadow collocation points, however, when considering an illuminated side we need to take this into account since s_m will fall within the boundaries of the basis function and hence the integral is highly peaked (as we saw with LHS2). We will now discuss the entries of the vector when s_m is on an illuminated side, the remaining $(n - n_s)$ sides.

4.3.2 RHS - Illuminated Sides

We now have the additional (central) term in (4.14), this requires no special allowances since it is just a simple function evaluation of known values. The same is also true for the evaluation of the first term in (4.14) and so the only term we need worry about is the integral term. As mentioned before when the collocation point is in the interval of integration we would expect to have a highly peaked function. As a result, we need to take similar action to what we did in LHS2 on the leading block diagonals and solve analytically first, we must also divide the integral up into n separate integrals where each one represents a side of the polygon. Where we have a collocation point falling outside any of the n integrals we proceed by using numerical quadrature as done for the shadow sides and also in LHS2 where we also used numerical integration. In the instance where the collocation point falls within one of these integrals, in order to evaluate the oscillatory part of the integral analytically, we can use the same code as we did for $I3$ except replace σ_j with $a_m \sin\theta - b_m \cos\theta$ and multiplying the whole integral by a different predetermined constant. By proceeding as we did previously, using identity (4.8) along with the definition of Υ (see (2.28)) and substituting into the (integral) term in (4.14) we have

$$\int_{L_j}^{L_{j+1}} K(s_m, t)\, \Upsilon(t)\, dt,$$

$$= \int_{L_j}^{L_{j+1}} -\frac{\eta}{4} H_0^{(1)}\left(k\left|s_m - t\right|\right) 2i \left(b_m \sin\theta + a_m \cos\theta\right) e^{ik\left[(a_m t + c_m)\sin\theta - (b_m t + d_m)\cos\theta\right]},$$

$$= -\frac{i\eta}{2}\left(b_m \sin\theta + a_m \cos\theta\right) e^{ik\left(c_m \sin\theta - d_m \cos\theta\right)} \int_{L_j}^{L_{j+1}} H_0^{(1)}\left(k\left|s_m - t\right|\right) e^{ik\left(a_m \sin\theta - b_m \cos\right)t} dt,$$

and using identity (4.9), switching the order of integration gives us

$$C \int_0^{\infty} \left[\int_{L_j}^{L_{j+1}} \frac{e^{(i-r)k|s_m - t| + ikat}}{r^{\frac{1}{2}}\left(r - 2i\right)^{\frac{1}{2}}} dt \right] dr, \tag{4.15}$$

with

$$s_m \in (L_j, L_{j+1})$$

where

$$C = -\frac{\eta}{\pi}\left(b_m \sin\theta + a_m \cos\theta\right) e^{ik\left(c_m \sin\theta - d_m \cos\theta\right)},$$

and

$$a = a_m \sin\theta - b_m \cos\theta. \tag{4.16}$$

So when we solve the inner integral of (4.15) analytically, we can think of it as (4.13) except with a different value for σ (or a as seen above), and with the constant multiplication, C. We approach the second integral calculation in exactly the same way as we did before by making the substitution $r = s^2/\left(1 - s^2\right)$ to reduce the range of integration and eliminate the singularity at $r = 0$.

So far in this chapter we have looked at the methods for setting up the linear system and the different considerations we require. For this *MATLAB* program being written we require it to be as efficient as possible and as a result it is important to consider what numerical integration technique we shall be using. We also need to decide how many points to choose over each interval of integration and how we can make this as small as possible to obtain a numerically accurate result. Before deciding on the amount of nodes we require over the interval we shall give a brief introduction to the numerical technique we will be using.

4.4 Gaussian Quadrature

There are many different techniques in which we can numerically approximate an integral, here we give a summary of the method we shall use along with how we determine the amount of nodes required.

There are two main types of method in which we can numerically integrate functions. Firstly, we have *Newton-Cotes Formulae*, these are a family of techniques that use evenly spaced functional values across the range of integration. For example, Trapezium Rule, Simpson's Rule, Duran's Rule, Weddle's Rule to name but a

few. The second type is the *Gaussian Quadrature* (GQ) which selects functional values at non-uniform intervals and corresponding weights across the range of integration to achieve a higher accuracy. We shall be using the GQ type.

The method we shall use is to be a composite GQ, by this we mean that we will split our interval into N_Q subintervals determined by some means, within each of these subintervals we determine the M_Q nodes and weights for the GQ method. When calculating integrals at the implementation phase we compute the required weights and nodes on the interval $[-1, 1]$ and transform them onto our interval of integration. We give a brief overview here as to how the GQ method works before moving on to specifically applying it to our problem and how we calculate the optimal values for N_Q and M_Q. Consider the integral

$$I(f) = \int_{-1}^{1} f(x)\, dx$$

we seek to approximate $I(f)$ by our quadrature of the form

$$Q(f) = \sum_{m=1}^{M_Q} w_m f(x_m) = w_1 f(x_1) + w_2 f(x_2) + \ldots + w_{M_Q} f\left(x_{M_Q}\right)$$

where the w_m are the weights and $f(x_m)$ is the function evaluation at the point x_m. The values of the weights and the position of the nodes are determined so that the method yields exact integration for $f(x) = x^0, x^1, \ldots, x^{2M_Q-1}$, this gives us $2M_Q$ degrees of freedom and so we are able to solve exactly if f was a polynomial of degree $2M_Q$. For our problem, the kernels in the integration when computing the linear system are of course not polynomials, but we can use this method to approximate them to a high order polynomial and get a very good approximation. All of the integrals we shall deal with will not fall in this range unfortunately and so we are required to transform the weights and nodes onto a general interval $[a, b]$ (an alternative approach is to transform the integral range $[a, b]$ onto $[-1, 1]$).
We make the following transformations in order to achieve this. For the weights we have

$$w_m \mapsto \frac{(b-a)}{2} \times w_m,$$

for the nodes we use the transformation

$$x_m \mapsto a + (b-a)\,\frac{(x_m+1)}{2}.$$

We know that we shall be using a composite GQ for our numerical integration for calculating entries in our system, so it now remains to determine the number of

subintervals required and Gaussian nodes on these intervals for each integral. The integrals we have are a combination of non-oscillatory and oscillatory functions, the latter of course needing more degrees of freedom over the same interval to get similar accuracy. We will now look at the implications of this and discuss how we can optimise our program to give the best results in the quickest time. Ideally, we would like to get an accurate result using as few nodes and weights as possible so that the *MATLAB* code completes the calculations quickly and efficiently.

There are many different considerations we need to take into account in determining the amount of subintervals and Gaussian nodes we require. We remind ourselves that the amount of subintervals over our integral will be N_Q and the number of Gaussian nodes and weights required in each of these intervals will be M_Q. We shall proceed by keeping the value of M_Q fixed and varying how many subintervals we shall divide the integral into, i.e. the N_Q value. We shall consider the non-oscillatory and oscillatory functions separately.

4.4.1 Non-Oscillatory Functions

In the case where we have a non-oscillatory function we are able to use fewer points to represent the integrand. It is acceptable in this case to choose $N_Q = 1$ and an appropriate amount of Gaussian nodes, i.e. $M_Q = 30$. The non-oscillatory functions we have are the instances where the collocation point falls on the same side of the polygon as the basis function, that is, the leading block diagonals in LHS2 which takes the general form

$$\int_0^\infty \frac{I(r)}{r^{\frac{1}{2}}(r-2i)^{\frac{1}{2}}}dr$$

and the evaluations in the right hand side vector on the illuminated sides, see (4.15) through (4.16).

4.4.2 Oscillatory Functions

For the integrals that oscillate we need to determine the value of N_Q, preferably dependent on the amount of oscillations and size of the integral support. The oscillatory integrals are the rest of the evaluations in our system (all those not mentioned in the case above) i.e. where we have not solved anything analytically. We recall that for each mesh (gridx and gridy) the basis functions near one of the corners of the polygon are smaller than those away from the other corner on that side (remembering of course that each mesh is graded towards opposite corners).

This suggests that we have a lower number of wavelengths in the smaller intervals, as a result we should be able to obtain an accurate result using fewer points. Consider

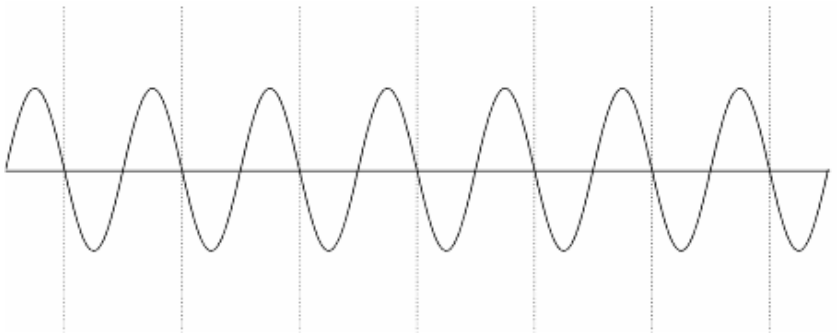

Fig. 4.1: Example of what the oscillating function may look like across a basis function many wavelengths long

figure 4.1, we wish to find a value of N_Q so that it is equal to the number of wavelengths across the interval, i.e. we are looking for N_Q intervals of length $\lambda = 1/k$ (one wavelength long), so we wish to divide as shown. We can then use M_Q nodes in each of these subintervals to approximate. We are looking for a value of N_Q such that

$$b - a = N_Q \lambda,$$

where b and a are the boundaries of each integral, this implies

$$N_Q = \left\lceil \frac{(b-a)}{\lambda} \right\rceil = \lceil k\,(b-a) \rceil\,, \tag{4.17}$$

where the $\lceil * \rceil$ denotes the smallest integer greater than or equal to $*$. In practice, for a more accurate result, we may increase this value of N_Q by dividing $(b-a)/\lambda$ by some value less than one in order to get more subintervals, this should however be sufficient for the problem. By choosing N_Q to be this value we will have more subintervals across the larger basis functions, this makes sense as we will clearly have a larger amount of wavelengths traveling across the larger elements at any one time and so more nodes will be needed. In section 2.4 we said typically we need ten nodes to represent one wavelength, we choose the value of M_Q to be larger than this, typically $M_Q = 30$ should be more than sufficient.

We now know how to choose how many Gaussian nodes and weights we require, we do however have an exception to this rule. We know that when a collocation point falls close to the edge of a basis function we have more contribution from that element in our linear system meaning that it is highly peaked. This happens at the

corners of the polygon. This choice of N_Q cannot model this well and so we will require a lot more points to get a true representation, hence we need a clause in this rule. That is, an "if" statement in our program stating that when a collocation point is less than some distance from the support of a basis function, then we need to increase the number of points in that interval, ideally using a graded mesh. The value we choose to be this distance is one wavelength, as a result as k increases, fewer elements fall into this category.

Although we will not implement a graded mesh for this specific program we shall mention how we would set one up. [12, pp.9-10] describes how this can be done over a two-dimensional interval, a similar approach can be applied over a one dimensional domain as we have here. Consider the interval $[a, a+\lambda]$, where a is the corner of our polygon and λ is close to a. We could impose a graded mesh and place nodes at for example the points

$$a + \lambda 0.15^{N_Q},$$

where N_Q is the number of nodes we choose to place on the interval determined by some means relative to how diffracted the wave will be, this will give us more dense points towards the corner where the diffraction will be highest as more nodes will be required. Doing this would allow maximum efficiency from our code and is another point to consider in developing the software. We will not develop this further here.

We will follow this discussion on GQ by discussing the error involved in approximating our functions.

4.5 Gaussian Quadrature Error

Numerical integration by GQ is designed to interpolate polynomials up to a specified degree where we have discrete data given. We mentioned in section 4.4, that given some function, we are able to integrate up to degree $2M_Q$ exactly, this is useful when we wish to solve polynomials, and can solve non-oscillatory functions relatively well, but GQ is not designed for oscillating functions. In this section we are going to assume that the function we wish to integrate behaves like the oscillatory function e^{ikt} and wish to integrate with respect to t. In actual fact, the oscillatory functions we are integrating in the set up of the system are a lot more complicated than this, but for the purposes of this error discussion, this will demonstrate the ideas involved. The GQ error for evaluating a function f, with $2n$ continuous derivatives

is defined as

$$\frac{f^{(2n)}(\xi)}{(2n)!}\|p_n\|^2$$

for some $\xi \in (a,b)$, and p_n is the orthogonal polynomial of degree n.
If we consider the n^{th} derivative of our assumed function then the general form is $(ik)^n e^{ikt}$, this function behaves like $(ik)^n$ i.e. it blows up as $n \to \infty$ (due to being oscillatory). This means for small n that the error estimate blows up rapidly before converging, as a result, it means we need a large amount of Gaussian points to get an accurate representation. As we are using a composite quadrature rule, we divide the interval into many smaller problems dependent on the wavenumber k. As a result, if we double k, for the same level of accuracy, we need twice the number of subintervals and so twice the computation capacity. This is not ideal when talking about very high wavenumbers.

The evaluation of high oscillatory integrals is a large area of research in optimising the efficiency in which a solution can be found, this is done by taking advantages of certain properties of the oscillating function. [9] and [10] discuss such ways we can do this, it is however unclear whether or not the properties on $K(s,t)$ are satisfied and so we have no guarantee of these methods converging. These methods will not be implemented here and so we will not consider them further. The final section in this chapter discusses how machine accuracy plays a part in our problem.

4.6 Machine Accuracy

We already know that as we get close to the corners of our polygon then the size of the basis functions are considerably smaller than if we are elsewhere on the grid, this of course is where we gain our computational advantage (by having more elements where we need them) but this can cause problems in the computation. Computer hardware can only carry out floating point arithmetic to some finite accuracy, this has three consequences in terms of what we must consider when implementing such a mesh.

1. There is a limited number of significant digits stored,
2. There is a maximum floating point number,
3. There is a smallest positive floating point number.

For our program, point 3 is most applicable. At first hand this problem of machine accuracy may seem negligible but it is an important consideration. In recent history

many disasters have been caused by numerical errors in computations resulting in either loss of life or huge financial catastrophes that could have been avoided. These include patriot missile failure, explosion of an unmanned rocket and the sinking of an offshore platform to name but a few, all of which could have been avoided if the errors were treated correctly. Any failure in this program because of this will not have such devastating effects, but may prove to give inaccuracies in our results and so we must treat them carefully.

Due to inaccuracies in computer hardware, it means that as we approach a larger number from a floating point number that is a very small distance away, then at some point the value will be rounded to that large number and so an error will form. Another way to put, is that because of the finite accuracy, there is a floating point number, $\delta > 0.0$, such that $1.0 - \delta$ equals 1.0 as far as the computer hardware can tell. This may not seem too much of a problem, but consider the following two equations and the results obtained.

1. large - (large' - ϵ)
2. (large - large') + ϵ

for $\epsilon \leq \delta$. In these cases we may obtain inaccuracies in the first calculation although the results should be the same. For some large value, the bracketed part in 1. will round to large', and so what actually is calculate is large - large'. This is clearly not correct, and this problem does not arise in case 2. If we consider the more extreme case when large = large' then the first calculation gives zero, and the second gives the correct answer of ϵ so we need to consider this in the implementation phrase.

The value of δ varies from machine to machine and so it is hard to put an actual figure on what it is, but it is very easy to write a piece of software to do calculate this. Typically, using *MATLAB* we have $\delta = 1.11e-16$ when using double precision (this value is considerably less for single precision). For the machine the results in this dissertation are being calculated on we have $\delta = 1.110223024955037e-016$. All this has major implications when writing the code. If we consider what sort of mesh sizes we will be dealing with, for a square with each side of length 2π, wavenumber $k = 160$ and $N = 32$ then the first collocation point on each side is of a distance $8.8818e-017$ (half the interval of the basis function) away from the corner, let us refer to this value as t. The easiest way to calculate t is simply calculate half the support for the ρ_j and add this to each of the lower boundaries of the basis functions, but we see the problem with this demonstrated by table 4.1. If programming in the

Collocation Point	Actual Calculated Col Pt	Error
t	t	0
2π+t	2π	t
4π+t	4π	t
6π+t	6π	t

Tab. 4.1: Rounding errors in collocation points when falling close to boundary

way described by 1. then it is clear that the collocation points are stored as the wrong values. This does make a small difference to the values calculated and means results can become more inaccurate as we keep carrying these small rounding errors forward, and so we look at ways to avoid them so that we do not cause too many inaccuracies in the solution to our problem. One of the main problems arises in the calculation of (2.24) for use in (2.22). In (2.22) a Hankel function is evaluated with the argument (2.24)×constant. Now, recall (2.24) takes the form

$$R = R\left(s_m, t\right) := \sqrt{\left(a_m s_m - a_j t + c_m - c_j\right)^2 + \left(b_m s - b_j t + d_m - d_j\right)^2},$$

where s_m represents the collocation point and t is the integration variable (when evaluating the integrals in our system), all the other variables are known constants. We tackle this slightly differently than how it is written, rather than storing the collocation points as values we have two storage vectors, one stores the length of the side, the other the distance the collocation point is from the nearest corner in the positive direction. This is slightly different depending on if we are on gridx or gridy but the basic principle is the same. i.e. all of the "distance from" points for gridx are positive, and gridy negative. If we consider the above calculation (first term only) then to avoid these rounding errors we break the collocation point into the nearest corner distance from zero and the distance of the collocation point from the corner and as a result program the code as

$$\left(\left(a_l \times \text{nearestside} - a_j t + c_l - c_j\right) + al * \text{distfromside}\right)^2.$$

This is needed as in some instances the value of R will be almost zero, if we do not do this then the evaluation of R will be zero due to the rounding error, and as mentioned before this needs to be passed through the Hankel function which is undefined at zero and hence the program cannot run. We also divide by this R value so we end up dividing through by zero which again is undefined so it is clear to see that these modifications to get results are necessary and should not be

treated lightly. The same also needs to be done for the second lot of terms in R, as in other areas of the program, this section has only considered the one example.

We have now discussed all theory behind the problem and how we implement the method. In the next chapter we discuss the results obtained from the method and assess whether or not the method has been successful.

5. NUMERICAL RESULTS AND DISCUSSION

We have mentioned throughout that one of the reasons we are implementing the collocation method is because it is more computationally efficient then the Galerkin method, this is an important advantage of our method and is something we shall focus on shortly, but first we are to focus on how successful the method has been. When judging if the collocation method implemented is successful for this problem, we investigate two main features of the results.

1. Firstly, as we increase the degrees of freedom over the boundary, we would require the error in the numerical solution to decrease with an estimated order of convergence (EOC) of one. We would expect this value for the EOC as we are approximating by piecewise constants, if we were to use higher order polynomials to approximate we would expect a better EOC. Using constants means that if we double the degrees of freedom we would expect the error to halve, although this is unlikely to be exact for reasons we discuss shortly, but we should see a trend.

2. Secondly, as we increase the value of the wavenumber, k, then the relative error stays roughly the same for constant N, (where N is the number of nodes between $[0, \lambda]$ as in definition 2). This means that the error is not dependent on the wavenumber of the incident wave.

If both these points hold it will suggest that this is a good method to use when considering high frequency problems. We can show that for the Galerkin method in the piecewise polynomial case that [12, Theorem 1.1.2]

$$k^{1/2} \left\| \varphi - \varphi_N \right\|_2 \leq C \sup_{\mathbf{x} \in D} \left| u\left(\mathbf{x}\right) \right| \frac{\left[n \left(1 + \log\left(kL/n \right) \right) \right]^{3/2}}{N}.$$

Where C is a constant independent of k and N and $\left\| \varphi - \varphi_N \right\|_2$ is the L_2-error. Although we cannot provide similar analysis for the collocation method, we would hope that our results hold close to the Galerkin error estimate above. To investigate this, we are to use a model problem of a square. Each side is to have length 2π,

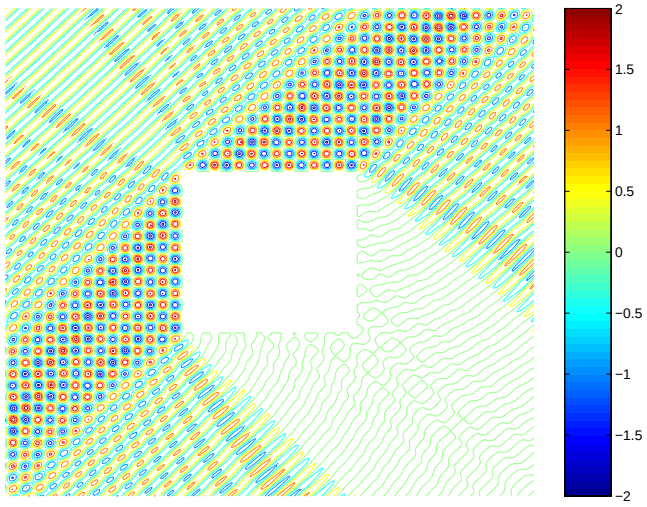

Fig. 5.1: Total Acoustic Field for k = 10, scattering by a square with incident wave $\pi/4$ measured from the downward vertical (top left corner to bottom right)

with each of the corners having an internal angle of $\pi/2$ and the angle of incident is to be $\pi/4$ measured anticlockwise from the downward vertical. The problem is as shown in figure 5.1

In calculating the errors we need an exact solution, unfortunately no such analytic solution exists for the problem and so we are going to assume that the best estimate we have for the problem is to be the exact solution and so will use the Galerkin estimate with $N = 256$. This model problem is the same as the one considered in [4].

We are to investigate varying values of k starting with $k = 5$ and doubling up for each test. The speed in which the linear system (3.6) is set up primarily depends on the choices of the number of subintervals and the Gaussian nodes/weights over each integral, clearly, the more we choose the more accurate the result obtained will be, but the computational time required will be higher. We could expect that doubling the nodes over each integral will double the time taken and so a balance needs to be struck between the accuracy of these calculations and the computational time taken. Where we have non-oscillatory integrals we take $N_Q = 1$ and $M_Q = 30$, in the case of oscillatory integrals we are to take $M_Q = 30$ for the Gaussian nodes and weights over each interval and N_Q to be as previously defined in section 4.4, and we recall it to be of the form

$$N_Q = \lceil k\,(b-a) \rceil\,. \tag{5.1}$$

For the purpose of this accuracy test, when the collocation point is less than one wavelength away from the basis function (when on different sides) we choose the number of subintervals, N_Q, to be dependent on the wavenumber. This is because for higher values of k we notice that each entry evaluation is more peaked than lower values of k, i.e. as k grows large, the functions we evaluate need more mesh points to represent their behavior. A brief comparison test suggests that $N_Q = 3k$ is sufficient for obtaining an accurate result in the evaluation of our integrals for this problem and so we take this to be the amount of subintervals in this exception case. The numerical evaluation of the integrals will still not be exact and we would still expect inaccuracies in the solution but these choices should reduce any errors that may arise. Table 5.1 demonstrates the results obtained for increasing values of k and N. For each k, we show the N values, the degrees of freedom (selected by definition 2 for setting up our mesh), the relative error (rel. $= \|\varphi - \varphi_{N_c}\|_2 \,/\, \|\varphi\|_2$) and the estimated order of convergence (EOC). The error calculated is the L_2 error where we take the exact solution to be the Galerkin solution for $N = 256$.

The results we have obtained look very promising for each case as it is clear to see the solution is converging. In a small number of cases it appears that the increase in N does not help improve our numerical solution. This is not unexpected since often results can take a while to settle down and so it is the latter results we find of most interest when analysing. It also appears that the EOC is not always close to one and in some cases very far away. This is because this calculation of the EOC assumes that we have evaluated everything exactly, this is clearly not the case as we are numerically integrating many functions and so will not obtain exact answers, so small errors will carry over to our solution. The EOC does however, in general, stay close to one for increasing N and so the results are what we might expect.

We wish to now discuss if we can call the method successful by definition of the two points we mentioned previously. Point 1 above is certainly satisfied for each wavenumber as it clearly converges as N grows larger with an EOC roughly equal to one. Point 2 requires us comparing the relative errors for each wavenumber, we notice that the errors for each N value are almost always of the same order as k increases. Although this is not perfect, because they are of the same order of magnitude, it is a reasonable statement to say that the method is not dependent on the wavenumber and so the method is successful for solving high frequency problems in the sense of the two points mentioned above. It is worth noting that

k	N	dof	Rel.	EOC	k	N	dof	Rel.	EOC
5	2	24	7.1425×10^{-1}	0.6	40	2	24	7.3945×10^{-1}	0.4
	4	48	4.7393×10^{-1}	0.6		4	56	5.4597×10^{-1}	0.7
	8	88	3.2269×10^{-1}	1.0		8	104	3.4089×10^{-1}	-0.1
	16	176	1.5496×10^{-1}	1.0		16	208	3.6095×10^{-1}	0.4
	32	360	7.7404×10^{-2}	1.0		32	416	2.8317×10^{-1}	2.9
	64	712	3.9425×10^{-2}	0.4		64	824	3.7158×10^{-2}	1.0
	128	1416	2.9569×10^{-2}	-		128	1656	1.8542×10^{-2}	-
10	2	24	7.5453×10^{-1}	0.7	80	2	24	7.6504×10^{-1}	0.7
	4	48	4.7335×10^{-1}	0.8		4	56	4.6096×10^{-1}	1.0
	8	96	2.6980×10^{-1}	1.1		8	112	2.3333×10^{-1}	0.6
	16	192	1.2670×10^{-1}	0.9		16	216	1.5975×10^{-1}	0.2
	32	376	6.8440×10^{-2}	1.2		32	432	1.4203×10^{-1}	1.7
	64	752	3.3034×10^{-2}	0.4		64	864	4.4374×10^{-2}	1.1
	128	1496	2.3141×10^{-2}	-		128	1736	2.1325×10^{-2}	.
20	2	24	7.5254×10^{-1}	0.1	160	2	32	6.8901×10^{-1}	0.6
	4	48	7.1085×10^{-1}	1.2		4	56	4.4455×10^{-1}	-0.1
	8	104	3.0762×10^{-1}	0.8		8	112	4.6445×10^{-1}	1.0
	16	200	1.7872×10^{-1}	1.7		16	224	2.3456×10^{-1}	1.3
	32	392	5.5728×10^{-2}	0.4		32	456	9.3327×10^{-2}	1.0
	64	792	4.1295×10^{-2}	1.0		64	904	4.8153×10^{-2}	0.7
	128	1576	2.0186×10^{-2}	-		128	1808	2.9822×10^{-2}	.

Tab. 5.1: Relative L_2 errors, $k = 5, 10, 20, 40, 80$ and 160. $N = 2, 4, 8, 16, 32, 64$, and 128

if we did eliminate these inaccuracies in the numerical integration we could expect these requirements to hold more strongly, unfortunately for obvious reasons this is not possible.

The final thing that we are take from tables 5.1 is looking at the degrees of freedom (dof) required in order to obtain a fixed level of accuracy. Throughout we have said that the amount of mesh points was no longer directly proportional to kL and we now have (dof) $\propto N \log k$. Figure 5.2 demonstrates the degrees of freedom required as k grows for the higher values of N (that is $N = 64$ and $N = 128$). It is clear to see the advantages of using this mesh for high frequency problems, the graphs clearly demonstrate the logarithmic increase in mesh points as the value of k increases. So doubling k has little effect on the degrees of freedom for fixed N as one would hope.

So far, we have established that the method converges for increasing values of

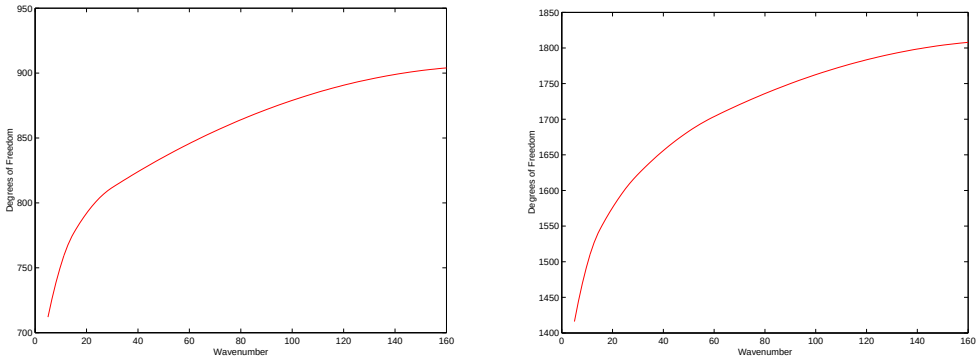

Fig. 5.2: Degrees of freedom for increasing wavenumbers for $N = 64$ and $N = 128$

N and is not dependent on the wavenumber k and so the method does indeed work for solving ((2.2),(2.3) and (2.4)).

We are now to look briefly into the computational cost of the collocation method in comparison to the Galerkin method. This is hard to compare directly since for a fair comparison we would require the computational times for each method in order to get the same level of accuracy. This is a near impossible task, but we can demonstrate the advantage of using the collocation method for increasing wavenumbers over the Galerkin method. It is well known that the collocation method is faster in setting up the systems in comparison to the Galerkin method and many comparisons have been done, [14] is one such example of this. In this comparison, a different problem was tackled but cost of setting up the system was similar. It was found that the amount of floating point operations (known as flops which gives one way to measure numerical work), required by the collocation method is typically 56-58% to that of the Galerkin method for a varying amount of mesh points. We may obtain a similar result to this figure but it ultimately depends on the quadrature points we use and because the quadrature points in the Galerkin method are calculated differently due to the double integrals, it is unlikely we will be able to make a direct comparison in the *cpu* times. For this test, in the collocation method we have taken the values of N_Q and M_Q to be of that we saw in the accuracy test, for the Galerkin method we have taken the quadrature points over each interval to be 6. It is not possible to get an accurate result from the Galerkin method by choosing the quadrature to be so small, but we choose this small value because of time constraints. Therefore, we cannot make a direct comparison and so for this test we are to investigate what increase in computational time is obtained

when increasing both N value and the wavenumber. Table 5.2 demonstrates the computational time for each of the methods for different wavenumbers and the ratio increase, the results were computed on an *AMD Athlon XP 3000+*, with *cpu* times presented in seconds.

k	N	Col Time	Ratio
10	2	$2.4530 \times 10^{+0}$	2.2
	4	$5.4380 \times 10^{+0}$	2.7
	8	$1.4578 \times 10^{+1}$	3.1
	16	$4.5094 \times 10^{+1}$	3.2
	32	$1.4324 \times 10^{+2}$	3.7
	64	$5.2354 \times 10^{+2}$	3.7
	128	$1.9266 \times 10^{+3}$	-

k	N	Gal Time	Ratio
10	2	$1.1797 \times 10^{+1}$	1.0
	4	$1.2016 \times 10^{+1}$	1.4
	8	$1.7093 \times 10^{+1}$	2.1
	16	$3.5671 \times 10^{+1}$	2.5
	32	$9.0281 \times 10^{+1}$	3.3
	64	$3.0217 \times 10^{+2}$	3.9
	128	$1.1689 \times 10^{+3}$	-

k	N	Col Time	Ratio
20	2	$4.2650 \times 10^{+0}$	2.3
	4	$9.5930 \times 10^{+0}$	2.8
	8	$2.6860 \times 10^{+1}$	2.9
	16	$7.7985 \times 10^{+1}$	3.1
	32	$2.4350 \times 10^{+2}$	3.6
	64	$8.8328 \times 10^{+2}$	3.6
	128	$3.1686 \times 10^{+3}$	-

k	N	Gal Time	Ratio
20	2	$4.0531 \times 10^{+1}$	0.8
	4	$3.4531 \times 10^{+1}$	1.2
	8	$4.1171 \times 10^{+1}$	1.5
	16	$6.2891 \times 10^{+1}$	2.1
	32	$1.3123 \times 10^{+2}$	2.8
	64	$3.7363 \times 10^{+2}$	3.7
	128	$1.3775 \times 10^{+3}$	–

k	N	Col Time	Ratio
40	2	$8.0630 \times 10^{+0}$	2.7
	4	$2.1735 \times 10^{+1}$	2.3
	8	$4.9640 \times 10^{+1}$	3.0
	16	$1.4686 \times 10^{+2}$	3.0
	32	$4.4695 \times 10^{+2}$	3.5
	64	$1.5575 \times 10^{+3}$	3.7
	128	$5.6956 \times 10^{+3}$	-

k	N	Gal Time	Ratio
40	2	$1.5377 \times 10^{+2}$	0.8
	4	$1.1722 \times 10^{+2}$	1.1
	8	$1.2319 \times 10^{+2}$	1.2
	16	$1.5203 \times 10^{+2}$	1.6
	32	$2.3975 \times 10^{+2}$	2.2
	64	$5.2584 \times 10^{+2}$	3.2
	128	$1.6912 \times 10^{+3}$	–

Tab. 5.2: Timings, $k = 10, 20,$ and 40. $N = 2, 4, 8, 16, 32, 64,$ and 128 Collocation Vs Galerkin

As mentioned previously we cannot compare the *cpu* times directly, but we can look at the increases for each method. For the collocation method we see that asymptotically (i.e. as $N \to \infty$) that for fixed k, if we double N, then the computational cost increases approximately four times. This is what one would expect as if we were to double N then the degrees of freedom roughly doubles and so the main matrix in the system will be double the size in both directions, i.e. it is

four times the size. There is of course the right hand side of the system to consider, this will double in size and is why we would expect the four-fold increase to be most noticeable in the limit. So as $N \to \infty$ the percentage of the overall calculations contributed from the right hand side vector will tend to zero. If we compare this to the Galerkin, similar results are obtained. This is for exactly the same reason, when we double N the degrees of freedom roughly doubles and so the system will be four times the size, and hence four times as many calculations required, again we see this best represented in the limit. So neither method gives a computational advantage in terms of the ratio when we increase the value for N.

$N = 4$	10	20	40	80	160
Col Time	$5.4380 \times 10^{+0}$	$9.5930 \times 10^{+0}$	$2.1735 \times 10^{+1}$	$3.9734 \times 10^{+1}$	$7.8625 \times 10^{+1}$
Ratio Inc.	1.8	2.3	1.8	2.0	–
Gal Time	$1.2016 \times 10^{+1}$	$3.4531 \times 10^{+1}$	$1.1722 \times 10^{+2}$	$4.3524 \times 10^{+2}$	$2.3641 \times 10^{+3}$
Ratio Inc.	2.9	3.4	3.7	5.4	–
$N = 16$	10	20	40	80	160
Col Time	$4.5094 \times 10^{+1}$	$7.7985 \times 10^{+1}$	$1.4686 \times 10^{+2}$	$2.8827 \times 10^{+2}$	$5.6218 \times 10^{+2}$
Ratio Inc.	1.7	1.8	1.9	2.0	–
Gal Time	$3.5671 \times 10^{+1}$	$6.2891 \times 10^{+1}$	$1.5203 \times 10^{+2}$	$4.7111 \times 10^{+2}$	$1.5081 \times 10^{+3}$
Ratio Inc.	1.8	2.4	3.1	3.2	–

Tab. 5.3: Timings for increasing k, $N = 4$ and $N = 16$

Next, we wish to establish the computational cost increase when we double the value of the wavenumber k. First, let us consider the collocation method, if we recall equation (4.17) (see also 5.1) where we define the number of Gaussian nodes and weights over each interval we recall it was proportional to the wavenumber. This demonstrates that for N constant, by doubling the wavenumber then the quadrature points over each interval will also double and so the computational time should double with it. For the Galerkin method a similar process is used in determining the quadrature. We recall back to section 3.1 where we commented that when setting up the linear system we have double integrals to numerically solve. As we have these double integrals the quadrature is doubled in two dimensions as oppose to one. This means that when we double k we expect the computational time to quadruple. Table 5.3 demonstrates the cases where $N = 4$ and $N = 16$ for varying k for each of the methods. We have chosen smaller values for N to represent this because as N increases, to get a good result we need to take higher values for the wavenumber and so choosing N to be low means we do not have to run k up to

extremely high numbers. As expected the results suggest that doubling k whilst increasing computational time four times for Galerkin, only doubles the *cpu* time for the collocation method, demonstrating a speed advantage as k gets large. Even if for lower k the Galerkin method is quicker then at some point when k gets to a large enough value then the collocation method will always be quicker to implement for solving the problem.

To finish off this section we are to present some more images from some of the results to further demonstrate the program produces correct results for a general problem. Throughout the demonstration we have limited k to be 20 or less in order to allow easier viewing of what the behavior is for each example. For each demonstration we show the full picture (which includes the sum of the scattered and incident waves) and also show the "diffracted" wave field (i.e. the total field minus the leading order terms) since we have been mainly concerned with the diffracted wave throughout. We calculate the value in this field by using (2.5) and our approximation to $\partial u/\partial \mathbf{n} = \phi = \varphi + \Upsilon$ and so calculate and plot

$$\begin{aligned} u &= u^i - \int \Phi\phi \\ &= u^i - \int \Phi(\Upsilon + \varphi) \\ &= u^i - \int \Phi\Upsilon - \int \Phi\varphi \end{aligned}$$

where we take the last term in the third equation to be the diffracted wave.

Figures (5.3) through (5.6) represent a series of demonstrations with varying values for k and θ (angle of incident where $\theta \in [0, \pi/2]$) as well as a number of different polygons we use to scatter the waves. For each figure we have four separate images, the top left represents the total acoustic pressure in the field (for this we have used the real values), the top right represents the absolute value of the total field and helps us clarify the overall behavior of the problem, this allows us to establish where we have waves with higher magnitude and where we do not. The bottom images represent the diffracted field, again we have the acoustic pressure and the absolute values.

In each example it is clear to see the wave field formed, the magnitude of the wave can be seen more clearly in the absolute value graphs for both the total field and the diffracted field. If we focus on the absolute values of the diffracted field it is clear to see that we have a higher diffraction in the places expected. These consist of the acute corners and where more of the corner is seen by the incident wave as we would expect.

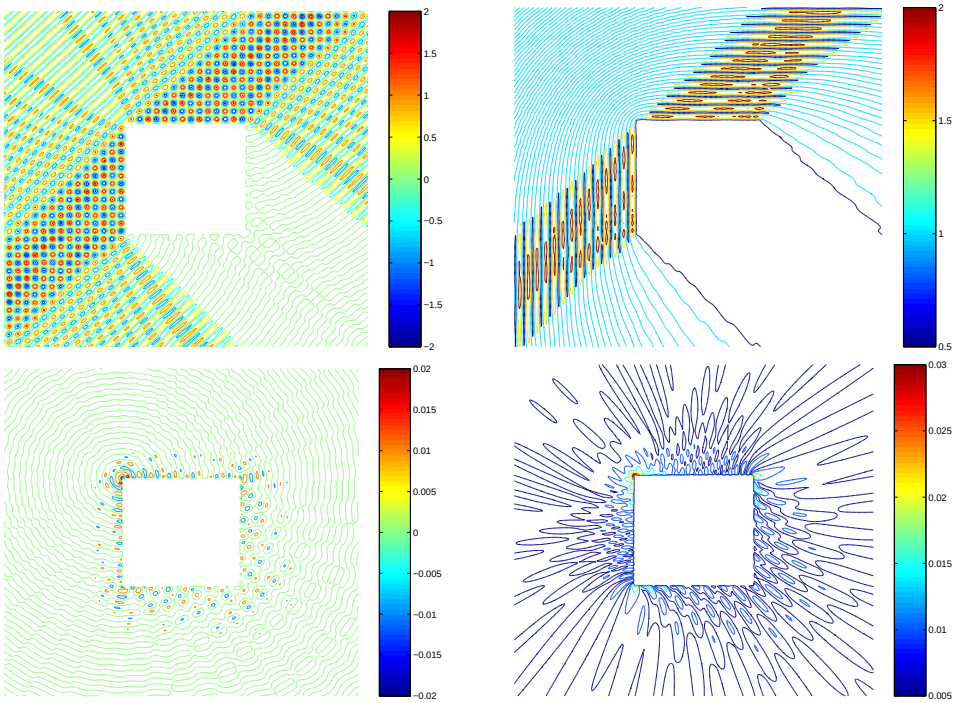

Fig. 5.3: Square, coordinates $(0,0)$ $(2\pi,0)$ $(2\pi,2\pi)$ $(0,2\pi)$. $\theta = \pi/4$, $k = 10$

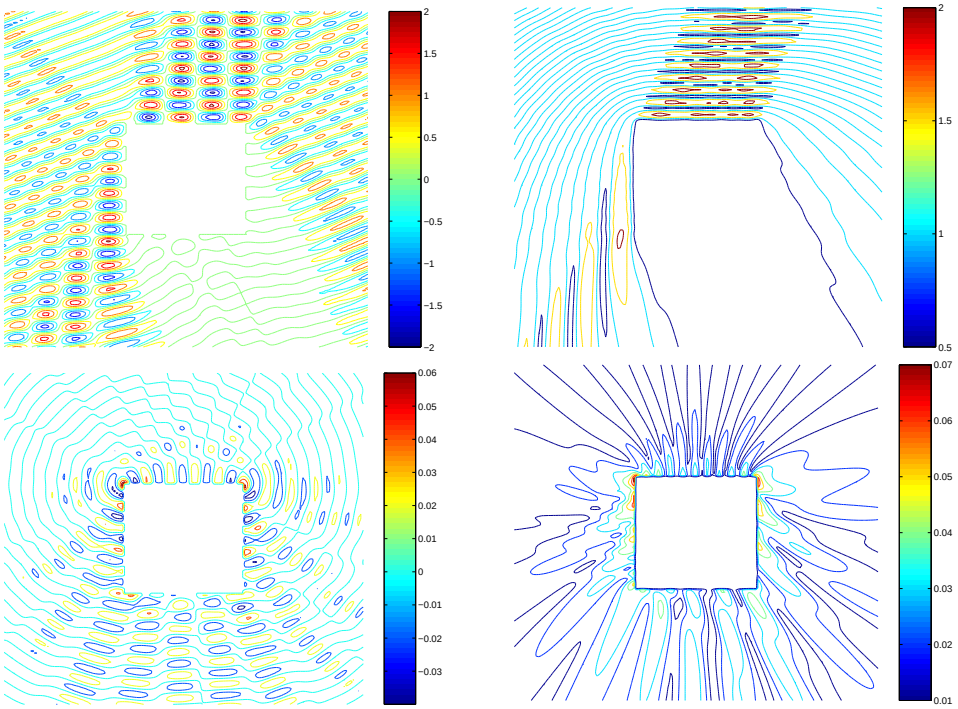

Fig. 5.4: Square, coordinates $(0,0)$ $(2\pi,0)$ $(2\pi,2\pi)$ $(0,2\pi)$. $\theta = \pi/8$, $k = 5$

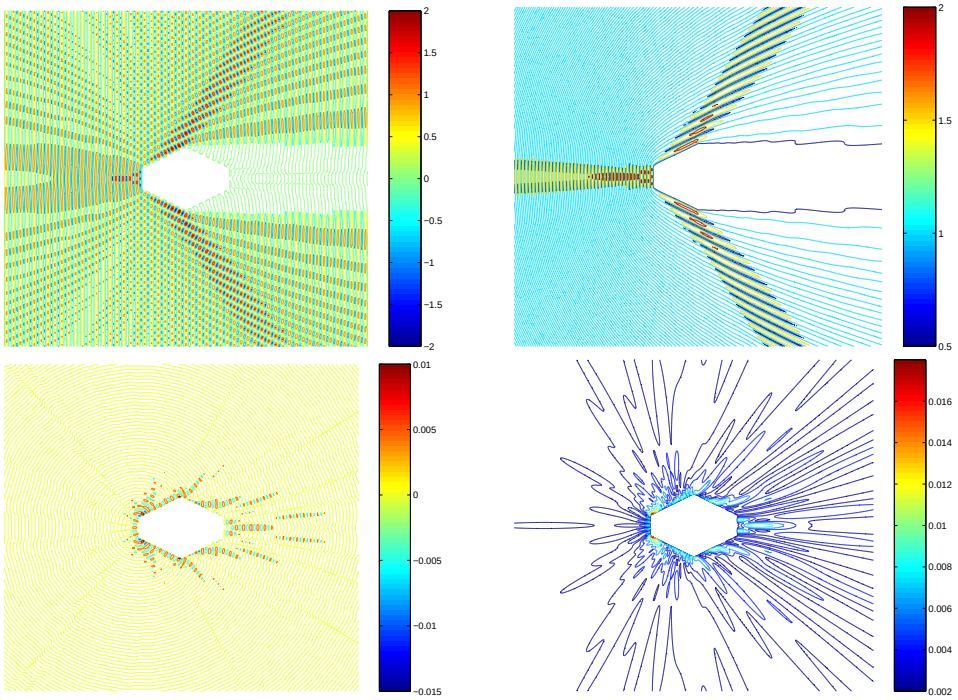

Fig. 5.5: Hexagon, coordinates $(0,0)$ $(2,1)$ $(2,2)$ $(0,3)$ $(-2,2)$ $(-2,1)$. $\theta = \pi/2$, $k = 20$

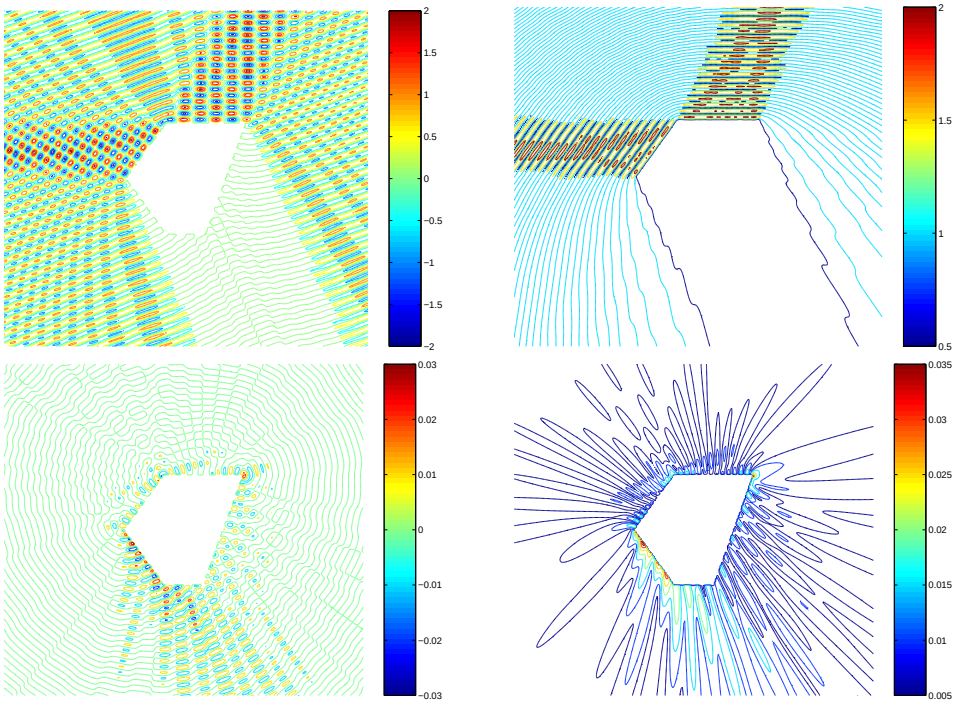

Fig. 5.6: Polygon, coordinates $(0,0)$ $(\frac{2}{3}\pi,-\pi)$ $(\frac{4}{3}\pi,-\pi)$ $(2\pi,\pi)$ $(\frac{2}{3}\pi,\pi)$. $\theta = \pi/8$, $k = 10$

6. CONCLUSIONS AND FURTHER WORK

We have implemented and presented a new collocation method for solving problems of high frequency scattering by convex polygons using a recently developed mesh and non-standard basis functions. We have also presented results of the accuracy and computational cost of the method in comparison with the Galerkin method.

The results obtained are very promising for this new method. We have established that the error in our solution converges with an approximate estimated order of convergence of one as we would expect. Also, we have shown that the error in the solution is not dependent on the wavenumber k and so is ideal for scattering on polygons as k grows larger. We have also demonstrated the computational advantage of the method over using the Galerkin method, this is seen more so for higher wavenumbers due to only doubling computational cost if we were to double the value of k, whereas the computational time with the Galerkin method quadruples. Although we have successfully implemented this method, many areas remain to be improved both in the current implementation and in how we can optimise this, and also how we can develop the method further to improve the convergence rate.

The first point we shall suggest improvement on is concerned with generating the amount of nodes for the numerical integration, specifically when a collocation point is near the support of the basis function when each is on a different side. We currently impose a set number of nodes predetermined by the user over these intervals, although this can be dependent on k it is not ideal as the amount of points we need will also be dependent on the angle of the corner, although what we have is sufficient for the program as it stands, it would be a welcome improvement. Ideally, a graded mesh can be written into the program as defined in section 4.4.2 where the amount of mesh points is dependent on the wavenumber and angle and not on a user entered value, this will improve the computational efficiency time for running our program.

Another improvement we could make is to do with the evaluation of the high-oscillatory integrals. We have discussed how the GQ is not designed for oscillating functions and so by learning more about the properties of the oscillating functions,

a method may be developed in that the evaluation of the integrals is a lot more efficient using fewer nodes over each interval of integration (see [9], [10]). The numerical integration in the program is where the majority of the computation time is taken up and so improving on this would improve time taken to solve the problem considerably.

We also talked about how a singular system may occur if a collocation point is exactly the same as a mid-point in our basis functions, and so the development of another mesh as discussed in chapter 3 will eliminate this problem. This will also improve the conditioning of the system if we can guarantee that the collocation points on *gridx* and *gridy* are at least some distance apart, say ϵ.

Finally, we wish to discuss what the next step forward would be in developing the method further. The approximation using piecewise polynomials can, on average, at best give us an order of convergence of one. So as we double the degrees of freedom the error in our approximation will on average at best halve. The future development of the method will be using higher order piecewise polynomials to approximate by, giving us a better order of convergence. Using a linear approximation will increase the estimated order of convergence to be two, and even higher order approximations will increase this order of convergence further. We have defined our basis functions as in chapter 3 for the piecewise polynomial case. One way in which we can approximate using linears is by using "hat functions" and still use the same technique of removing the oscillation of the incident wave as we have done here. The new basis functions would then take the form

$$\frac{\partial u}{\partial \mathbf{n}}(s) = \sum_{j=1}^{m} \left[e^{ik(s-c)} \rho_j(s) \right],$$

where the ρ_j is now the hat function which will take the form

$$\rho_j(s) = \begin{cases} \frac{s-s_{j-1}}{s_j - s_{j-1}} & s \in [s_{j-1}, s_j] \\ \frac{s_{j+1}-s_j}{s_{j+1}-s_j} & s \in [s_j, s_{j+1}] \\ 0 & \text{otherwise} \end{cases}$$

This will allow to increase the estimated order of convergence for our method to be two, and there is also scope to approximate using even higher order polynomials to obtain better rates of convergence.

BIBLIOGRAPHY

[1] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*, Dover, 1972.

[2] S. Amini, *On the choice of the Coupling Parameter in Boundary Integral Formulations of the Exterior Acoustic Problem*, Applicable Analysis, Vol. 35 pp. 75-92, 1990.

[3] A.J. Burton and G.F. Miller, *The application of integral equation methods to the numerical solution of some exterior boundary-value problems*, Proc. R. Soc. Land A. 323, 201-210, 1971.

[4] S.N. Chandler-Wilde and S. Langdon, *A Galerkin boundary element method for high frequency scattering by convex polygons*, in preparation, 2005.

[5] D. Colton and R. Kress, *Inverse Acoustic Scattering and Electromagnetic Scattering Theory* Springer Verlag, 1992.

[6] E. Giladi and J.B. Keller *Hybrid Numerical Asymptotic Method for Scattering Problems*, Journal of Computational Physics, 174, 226-247, 2001.

[7] W. Hackbusch, *Integral Equations*, Birkhuser, 1995.

[8] Joseph. B. Keller *Geometrical Theory of Diffraction*, Journal of the Optical Society of America, 1961.

[9] A. Iserles, *On the numerical quadrature of highly-oscillating integrals I: Fourier Transforms*, IMA Journal of Numerical Analysis, 24, 365-391, 2004.

[10] A. Iserles, *On the numerical quadrature of highly-oscillating integrals II: Irregular oscillators*, IMA Journal of Numerical Analysis, 25, 25-44, 2005.

[11] S. Langdon, *Domain Embedding Boundary Integral Equation Methods and Parabolic PDEs*, Ph. D., University of Bath, 1999.

[12] S. Langdon and S.N. Chandler-Wilde, *BEM for High Frequency Scattering in Advances in Boundary Integral Methods*, Proceedings of the 5th UK conference on Boundary Integral Methods, 2-11, 2005.

[13] E. Perrey-Debain, O. Laghrouche, P. Bettess and J. Trevelyan *Plane wave basis finite elements and boundary element methods for three dimensional wave scattering*, Phil. Trans. R. Soc. Land. 362,561-577, 2004.

[14] L. Ritter, *A Comparison of Galerkin and Collocation Methods on a Wave Scattering Problem*, Thesis Diploma, University of Hannover 2000.

[15] I.H. Sloan, *Boundary Element Methods in Theory and Numerics of Ordinary and Partial Differential Equations*, (Ed. M. Ainsworth, J. Levesley, W.A. Light and M. Marletta) Clarendon Press, Oxford, 143-180, 1995.

[16] M. Webber, *The Point Source Method in Inverse Acoustic Scattering*, MSc Dissertation, University of Reading, 2004.