UNIVERSITY OF READING

School of Mathematics, Meteorology and Physics

Numerical Evaluation of Oscillatory Integrals

Chloe Ward

August 2008

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

Abstract

The direct wave scattering problem has long been of interest for study in many disciplines such as engineering and geology. High frequency problems present many difficulties, since oscillatory behaviour is difficult to evaluate numerically. Numerical methods which can manage high frequencies and have a fast rate of convergence are desirable.

This dissertation reviews one such method, from recent literature, for the numerical solution of problems of wave scattering by convex obstacles. A standard numerical method is also implemented, providing some numerical results to investigate the effects of high frequencies on such a problem.

Acknowledgements

Firstly, I would like to say thank you to Dr. Stephen Langdon for his support and ideas while supervising this project. Also, thank you to Prof. Simon Chandler-Wilde for his additional advice as co-supervisor. I would also like to say thank you to EPSRC for funding this MSc.

Many thanks should also go to all my fellow MSc students for the sharing of sweets and laughs throughout the many hours spent in the computer room. I feel I should especially acknowledge Chris Warner who has been a fountain of knowledge and a great friend throughout the year.

Lastly, thank you to my family, friends and boyfriend for their encouragement and support over the years.

I wish to dedicate this project to the memory of my Grandad.
"To cut a long story short..."

Contents

Chapter 1

Introduction

1.1 Motivation

Scattering theory has a long history and has played an important role in mathematical and engineering research for over a hundred years [1]. Modelling radar, sonar and ultrasound, mentioned in [2], are just a few of the vast span of applications provoking the strong interest in acoustic and electromagnetic wave scattering problems. Such problems are often formulated as the Helmholtz equation

$$\triangle u(x) + k^2 u(x) = 0 \quad \text{in} \quad \mathbb{R}^m \backslash \Omega, \tag{1.1}$$

where $m = 2, 3$, corresponding to the two-dimensional (2D) and three-dimensional (3D) cases respectively and Ω is the convex scattering obstacle.

When solving these problems numerically, standard methods require a fixed number of discretization points per wavelength λ. For example the finite element method involves the division of the entire domain into small finite segments such as triangles or quadrilaterals. The behaviour of variables in each element is then defined and assembly leads to a large matrix system which must then be solved using a numerical solver such as Gaussian elimination or Gauss-Seidel. Alternatively we can consider the boundary element method. This is the numerical integration over the boundary which has been divided into small boundary segments (or elements). This yields unsymmetric, full matrices [8]. This means at higher frequencies k, where $k = 2\pi/\lambda$, the number of elements required for a "good" approximation rapidly becomes very large making the method computationally expensive. Although there exist faster implementations of such standard methods, as the size of the domain being considered gets larger relative to a small wavelength λ, current numerical methods are dependent on λ, hence many high frequency problems remain unsolved.

Consequently methods capable of evaluating high frequency problems would be greatly advantageous, particularly a method which has a bounded computational complexity as the wavenumber tends to infinity. The production of such methods would enable the solution of arbitrarily large scattering problems.

1.2 Aims and Outline

We study the ideas and numerical method presented by [3] regarding the numerical solution of scattering by convex obstacles. The paper does not directly state the numerical method used but gives the key ideas which will be discussed in the following Chapter 2. As mentioned above, implementing conventional numerical methods leads to the growth in computational complexity as wavelength λ decreases. The literature paper [3] we study here, applies a method involving a fixed number of degrees of freedom, independent of wavenumber k tending to infinity.

Firstly, we will implement a simple Nyström method for solving a direct scattering problem for which the convex scatterer is a circle. This will provide a preliminary test to investigate the effect of higher frequencies on the problem. Hence we will produce some justification for the requirement of a method such as that presented in [3].

We will then describe the steps to implement the method of [3]. Each of the separate parts towards the full method will then be implemented using MATLAB. A description of the codes used will then be provided along with an indication of how the functions work.

Chapter 2

Literature

In this chapter we elaborate on some of the ideas referred to in Section 1.2, present the details of the problem and provide a review of the literature paper [3].

2.1 Setting up the problem

Suppose an incident plane wave $u^i(x) = e^{ikx\cdot\alpha}$, $x \in \mathbb{R}^m$, where k denotes the wavenumber and $|\alpha| = 1$, is scattered by the bounded convex obstacle Ω. A scattered wave $u^s(x)$ is produced which satisfies the Sommerfeld radiation condition

$$\lim_{r\to+\infty} r\left(\frac{\partial u^s(x)}{\partial r} - iku^s(x)\right) = 0, \qquad r = |x|.$$

This condition guarantees that, physically speaking, the scattered wave is outgoing.

We seek the total wave

$$u(x) = u^i(x) + u^s(x),$$

which is a solution to the Helmholtz equation, given in equation (1.1). We will consider the case of the Dirichlet boundary condition

$$u(x) = 0 \quad \text{on} \quad \Gamma,$$

where Γ is the boundary of the convex obstacle Ω. So we will be considering the *direct* scattering problem, in which we determine $u^s(x)$ from the known $u^i(x)$. The *inverse* scattering problem would be to determine the obstacle function from the known behaviour of the scattered field $u^s(x)$. The fundamental solution of the Helmholtz equation (1.1) denoted here by $\Phi(x, y)$ is

$$\Phi(x,y) := \begin{cases} \frac{i}{4} H_0^{(1)}(k|x-y|) & \text{in 2D case,} \\ \frac{\exp(ik|x-y|)}{4\pi|x-y|} & \text{in 3D case,} \end{cases} \tag{2.1}$$

for $x, y \in \mathbb{R}^m$ and $x \neq y$, where $H_0^{(1)}$ denotes the Hankel function of the first kind and zero order. This can be expressed as the sum

$$H_0^{(1)}(z) = J_0(z) + iY_0(z), \qquad z \in \mathbb{R}^m \tag{2.2}$$

where $J_0(z)$ and $Y_0(z)$ are Bessel functions of zero order, first and second kind respectively [4]. The Hankel function is oscillatory. Using [5], we can state the asymptotic behaviour of these Bessel functions as $z \to 0$,

$$\begin{aligned} J_0(z) &\sim 1, \\ Y_0(z) &\sim \frac{2}{\pi} \ln(z). \end{aligned} \tag{2.3}$$

This means

$$H_0^{(1)}(z) \sim \frac{2}{\pi} \ln(z) \qquad \text{as } z \to 0.$$

Hence we see that $H_0^{(1)}(z)$ tends to $-\infty$, as $z \to 0$, and we conclude that $\Phi(x, y)$ in the 2D case is singular at $x = y$. Noting that clearly it is also true in the 3D case that $\Phi(x,y)$ is singular for $x=y$.

2.2 Reformulation

We want to reformulate the problem as a boundary-integral equation. Using Green's representation theorem we obtain

$$u(x) = u^i(x) + \int_\Gamma \Phi(x,y)\frac{\partial u(y)}{\partial n}\mathrm{d}s(y), \qquad x \in \Omega,$$

where $n(x)$ is the outward pointing normal. The normal is external to the computational domain, which in this case is outside the scatterer, hence we take the normal direction *into* the scattering obstacle. The scattering problem is then formulated as the boundary integral equation

$$\begin{aligned}\frac{1}{2}\frac{\partial u(x)}{\partial n} = \left(\frac{\partial u^i(x)}{\partial n} + i\gamma u^i(x)\right) &+ \int_\Gamma \frac{\partial \Phi(x,y)}{\partial n(x)}\frac{\partial u(y)}{\partial n}\mathrm{d}s(y) \\ &+ i\gamma\int_\Gamma \Phi(x,y)\frac{\partial u(y)}{\partial n}\mathrm{d}s(y), \end{aligned}\tag{2.4}$$

where γ is a positive arbitrary constant, referred to as the *coupling parameter*, ensuring that equation (2.4) is well-posed, provided $\gamma \neq 0$. If $\gamma = 0$, we cannot guarantee uniqueness. We can re-write equation (2.4) as

$$\frac{1}{2}\mu(x) = \left(\frac{\partial u^i(x)}{\partial n} + i\gamma u^i(x)\right) + \int_\Gamma \frac{\partial \Phi(x,y)}{\partial n(x)}\mu(y)\mathrm{d}s(y) + i\gamma\int_\Gamma \Phi(x,y)\mu(y)\mathrm{d}s(y)\tag{2.5}$$

where now we have now denoted our unknown $\partial u(x)/\partial n$ by $\mu(x)$ to simplify notation. The paper [3] proposes an ansatz of the form

$$\mu(x) = \mu_{\text{slow}}(x)e^{ikx\cdot\alpha},\tag{2.6}$$

for a convex obstacle, where the new unknown function $\mu_{\text{slow}}(x)$, $x \in \Gamma$, is slowly oscillatory. We can now substitute the ansatz (2.6) into equation (2.5) to obtain

$$\begin{aligned}\frac{1}{2}\mu_{\text{slow}}(x)e^{ikx\cdot\alpha} = \frac{\partial u^i(x)}{\partial n} + i\gamma u^i(x) \quad &+ \quad \int_\Gamma \frac{\partial\Phi(x,y)}{\partial n(x)}\mu_{\text{slow}}(y)e^{iky\cdot\alpha}\mathrm{d}s(y)\\ &+ \quad i\gamma\int_\Gamma \Phi(x,y)\mu_{\text{slow}}(y)e^{iky\cdot\alpha}\mathrm{d}s(y).\end{aligned} \tag{2.7}$$

Thus division of (2.7) by $e^{ikx\cdot\alpha}$ to simplify yields

$$\begin{aligned}\frac{1}{2}\mu_{\text{slow}}(x) \quad &- \quad \int_\Gamma \frac{\partial\Phi(x,y)}{\partial n(x)}\mu_{\text{slow}}(y)e^{ik\alpha\cdot(y-x)}\\ &- \quad i\gamma\int_\Gamma \Phi(x,y)\mu_{\text{slow}}(y)e^{ik\alpha\cdot(y-x)}\mathrm{d}s(y) = ikn(x)\cdot\alpha + i\gamma,\end{aligned} \tag{2.8}$$

since $u^i(x) = e^{ikx\cdot\alpha}$ and $\frac{\partial u^i(x)}{\partial n} = ikn(x)\cdot\alpha$. This can be more neatly written in operator form as

$$\frac{1}{2}\mu_{\text{slow}}(x) - (K\mu_{\text{slow}})(x) - i\gamma(H\mu_{\text{slow}})(x) = ikn(x)\cdot\alpha + i\gamma, \tag{2.9}$$

where K and H denote integral operators defined as

$$\begin{aligned}(K\mu_{\text{slow}})(x) \quad &= \quad \int_\Gamma \frac{\partial\Phi(x,y)}{\partial n(x)}e^{ik\alpha.(y-x)}\mu_{\text{slow}}(y)\mathrm{d}s(y)\\ (H\mu_{\text{slow}})(x) \quad &= \quad \int_\Gamma \Phi(x,y)e^{ik\alpha.(y-x)}\mu_{\text{slow}}(y)\mathrm{d}s(y).\end{aligned} \tag{2.10}$$

We have set the new unknown $\mu_{\text{slow}}(x)$, in equation (2.9), to be a slowly oscillatory function. However, the kernels of the integrals (2.10) are still highly oscillatory. Using a standard numerical method at this stage would still require a number of discretization points dependent on the wavenumber k. This will be demonstrated in Chapter 3 by the implementation of a simple Nyström method. The paper [3] extends the method of stationary phase and produces a convergent method which enables the solution of problems involving arbitrary frequencies.

Method of Stationary Phase

We now give a brief explanation of the key idea of stationary phase as used in the numerical method by [3].

The oscillatory integrals we want to evaluate are of the general form

$$\int_0^{2\pi} f(x)e^{ik\phi(x)}\mathrm{d}x, \tag{2.11}$$

where $f(x)$ and $\phi(x)$ are not oscillatory. The numerical method in [3] involves high-frequency problems. As $k \to \infty$, the rapid oscillations in the exponential term tend to cancel one another out. We can write

$$\int_0^{2\pi} f(x)e^{ik\phi(x)}\mathrm{d}x = \int_0^{2\pi} \frac{f(x)}{ik\phi'(x)} \left[ik\phi'(x)e^{ik\phi(x)}\right] \mathrm{d}x. \tag{2.12}$$

Now integrating equation (2.12) by parts we obtain

$$\frac{1}{ik}\left[\frac{f(x)}{\phi'(x)}e^{ik\phi(x)}\right]_0^{2\pi} - \frac{1}{ik}\int_0^{2\pi} e^{ik\phi(x)}\left(\frac{f(x)}{\phi'(x)}\right)' \mathrm{d}x. \tag{2.13}$$

Due to the 2π-periodicity, the first term of equation (2.13) cancels. We repeat the integration by parts procedure n times with the following outcome

$$\int_0^{2\pi} f(x)e^{ik\phi(x)}\mathrm{d}x = O\left(\frac{1}{k^n}\right), \qquad \text{for all } n,$$

provided $\phi'(x) \neq 0$. At the points where $\phi'(x) = 0$, for which $\phi(x)$ varies most slowly, the cancellation has least effect [6]. This idea is at the core of the method of stationary phase. We estimate the integral by taking the sum of evaluations over a small neighbourhood around each stationary point.

We can now find the critical points of integrals (2.10). As stated in [3], the kernels oscillate as

$$e^{ik[|x-y|+\alpha\cdot(y-x)]} = e^{ik\phi}, \tag{2.14}$$

as $k \to \infty$, where x and y are the target point and source point on Γ, the boundary of the scattering obstacle. By [4], this arises from

$$H_0^{(1)}(z) = e^{iz}.\left[e^{-iz}H_0^{(1)}(z)\right],$$

where $e^{-iz}H_0^{(1)}(z)$ is not oscillatory. One of the critical points is the target point x, where the kernel is singular. The other critical points are the stationary points, where the derivative of the phase function vanishes. The phase function in this case being $\phi = |x-y| + \alpha\cdot(y-x)$. We consider our convex scatterer to be a circle of radius a. The critical points of a phase as above can be found by using a polar parameterization

$$x = x(\theta_0) = ae^{i\theta_0} \qquad \text{and} \qquad y = y(\theta) = ae^{i\theta}. \tag{2.15}$$

We can substitute this into phase ϕ, and assume $\alpha = (1, 0)$. This is then differentiated and the stationary points can be found. This procedure is carried out in Appendix A of [3].

2.3 Localization

We have already established in the previous Section 2.2 that the stationary points of $\phi(x)$ make the significant contributions to the integral. Thus it is about these critical points we want to *localize* our integration method. We will now consider the case for which $\phi(x) = x^p$, where $p \geq 1$ is a real number. As discussed in [3], this directly applies to the integrals in (2.10) that we will be evaluating. The asymptotic expansion of the phase in (2.14) by Taylor series is well represented by a phase $\phi(x) = x^p$ with $p = 1, 2, 3$ corresponding to the cases where the kernel is singular, the stationary points (excluding shadow boundaries) and shadow boundary points respectively. The shadow boundaries being the points, x at which $\alpha \cdot n(x) = 0$. In order to build the method in the literature [3], we first define a function

$$S(x, x_0, x_1) = \begin{cases} 1 & \text{if } x \leq x_0, \\ \exp\left(\frac{2e^{(-1/u)}}{u-1}\right) & \text{if } x_0 < x < x_1, \\ 0 & \text{if } x \geq x_1, \end{cases} \tag{2.16}$$

where

$$u = \frac{x - x_0}{x_1 - x_0}, \tag{2.17}$$

and $x_0 < x_1$ are real constants. Since u is as defined in equation (2.17), then $0 < u < 1$. Furthermore, by some simple analysis of the function (2.16) we observe that $S(x, x0, x1) \to 1$, as $x \to x_0$ from above, in other words as $u \to 0$. Similarly we can add that $S(x, x0, x1) \to 0$, as $x \to x_1$ from below or $u \to 1$. Also by considering the derivative dS/dx such that

$$\begin{aligned} \frac{dS}{dx} &= \frac{du}{dx}.\frac{dS}{du} \\ &= \frac{1}{x_1 - x_0}\left(2\exp\left[-\frac{1}{u} + \frac{2e^{(-1/u)}}{u-1}\right]\right)\left(\frac{u - 1 - u^2}{u^2(u-1)^2}\right), \end{aligned}$$

where $u(x)$ is as defined in equation (2.17). We find that as x approaches x_0 from above, dS/dx tends to zero. Moreover as x approaches x_1 from below, dS/dx tends to zero. Hence we have a smooth function (2.16).

In [3] another function $f_A(x)$ is defined. A lemma using this is then presented which will be used as the key idea behind the numerical method being investigated, where the function $f_A(x)$ acts as a mollifying function. This is given by the following expression:

$$f_A(x) = S(x, cA, A)(1 - S(x, -A, -cA)), \tag{2.18}$$

for real numbers $A > 0$ and $0 < c < 1$. In order to state the lemma we require an additional function $f_\varepsilon(x)$ which, for $0 < \varepsilon < A$, is expressed by

$$f_\varepsilon(x) = f_A\left(\frac{Ax}{\varepsilon}\right). \tag{2.19}$$

We now state the lemma from the literature. It means that instead of integrating over the interval $-A$ to A, we can evaluate over the smaller interval $-\varepsilon$ to ε, under certain conditions. The full proof is omitted and a brief outline of the steps involved is given instead. The details of the proof can be found in [3].

Lemma 2.1 *Given real numbers $A > 0$, $0 < \varepsilon < A$, $0 < c < 1$ and $p \geq 1$, and functions $f_A(x)$ and $f_\varepsilon(x)$ defined by (2.18) and (2.19) respectively,*

$$\int_{-A}^{A} f_A(x)e^{ikx^p}\mathrm{d}x = \int_{-\varepsilon}^{\varepsilon} f_\varepsilon(x)e^{ikx^p}\mathrm{d}x + O((k\varepsilon^p)^{-n}) \qquad \forall n \geq 1. \tag{2.20}$$

Proof. (*Sketch*) The proof begins by defining a function

$$g_{A,\varepsilon}(x) = f_A(x) - f_\varepsilon(x).$$

Then an error E is considered such that

$$\begin{aligned} E &\equiv \int_0^A f_A(x)e^{ikx^p}\mathrm{d}x - \int_0^\varepsilon f_\varepsilon(x)e^{ikx^p}\mathrm{d}x \\ &= \int_{c\varepsilon}^A g_{A,\varepsilon}(x)e^{ikx^p}\mathrm{d}x \end{aligned} \tag{2.21}$$

We can then make a substitution for $x^p = t$ and integrate by parts n times. We know that $g_{A,\varepsilon}(x) = 0$ for $x = c\varepsilon$ and $x = A$, as well as the derivatives vanishing at each of these points. Hence using a similar argument to that of integration by parts outlined in the previous section, equation (2.13), an upper bound can be found for $|E|$ of order given in (2.20). ∎

2.4 Partition of Unity

Each of the critical points is covered by a region of radius dependent on the wavelength λ and the type of critical point being considered. In particular, the point at which the kernel is singular is covered by a region of radius proportional to λ. The stationary points (where the derivative of phase ϕ is equal to zero) are covered by a region proportional to $\sqrt{\lambda}$, and $\sqrt[3]{\lambda}$ at the shadow boundaries.

As discussed in [3], the integral over Γ is split into a number of integrals over subsets of Γ. Given an integral:

$$\int_0^{2\pi} f(x)\mathrm{d}x,$$

where $f(x)$ has critical points C_i, $i = 1, 2 \dots, n$. This integral can then be split in the following way

$$\begin{aligned}\int_0^{2\pi} f(x) = &\int_0^{2\pi} \{\chi_1(x)f(x) + \chi_2(x)f(x) + \cdots + \chi_n(x)f(x)\} \, \mathrm{d}x \\ &+ \int_0^{2\pi} \{1 - \chi_1(x) - \chi_2(x) - \cdots - \chi_n(x)\} f(x) \mathrm{d}x,\end{aligned} \tag{2.22}$$

where $\chi_i(x)$ are mollifying functions, defined as $f_A(x)$ in equation (2.18). Using Lemma 2.1, this can then be approximated around the critical points of $f(x)$, so integral equation (2.22) is approximately equal to:

$$\begin{aligned}&\int_{C_1-\varepsilon}^{C_1+\varepsilon} \chi_1(x)f(x) + \int_{C_2-\varepsilon}^{C_2+\varepsilon} \chi_2(x)f(x) + \cdots \\ &\qquad + \int_{C_n-\varepsilon}^{C_n+\varepsilon} \chi_n(x)f(x) + \int_0^{2\pi} \text{highly oscillatory terms}\end{aligned} \tag{2.23}$$

The highly oscillatory terms integrated over the interval $[0, 2\pi]$ are neglected since they do not include any stationary points, therefore cancelling out. This uses the same integration by parts argument used above.

Example 2.2. Suppose there exists *one* stationary point at $x = c$ and $\chi(x)$ is a mollifying function as previously described, we have

$$\begin{aligned}\int_0^{2\pi} f(x)e^{ik\phi(x)}\mathrm{d}x &= \int_0^{2\pi} \chi(x)f(x)e^{ik\phi(x)}\mathrm{d}x + \int_0^{2\pi} [1-\chi(x)]\, f(x)e^{ik\phi(x)}\mathrm{d}x \\ &= \int_{c-\varepsilon}^{c+\varepsilon} \chi(x)f(x)e^{ik\phi(x)}\mathrm{d}x + \left[\frac{(1-\chi(x))}{ik\phi'(x)}f(x)e^{ik\phi(x)}\right]_0^{2\pi} \\ &- \frac{1}{ik}\int_0^{2\pi} e^{ik\phi(x)} \underbrace{\left[\frac{(1-\chi(x))}{\phi'(x)}f(x)\right]'}_{g(x)} \mathrm{d}x,\end{aligned}$$

after localizing the integration around the critical point and integrating by parts.

Then using the 2π-periodicity, we can cancel the second term so

$$\int_0^{2\pi} f(x)e^{ik\phi(x)}\mathrm{d}x = \int_{c-\varepsilon}^{c+\varepsilon} \chi(x)f(x)e^{ik\phi(x)}\mathrm{d}x - \frac{1}{ik}\int_0^{2\pi} e^{ik\phi(x)}g(x)\mathrm{d}x.$$

Integrating by parts is repeated to obtain

$$\int_0^{2\pi} f(x)e^{ik\phi(x)}\mathrm{d}x = \int_{c-\varepsilon}^{c+\varepsilon} \chi(x)f(x)e^{ik\phi(x)}\mathrm{d}x + \frac{1}{(ik)^n}\int_0^{2\pi} e^{ik\phi(x)}g_n(x), \qquad (2.24)$$

for all n. $\square$

In other words as $k \to \infty$ the second term on the right hand side of equation (2.24) tends to zero.

The details of implementing the method described here will follow in Chapter 5. A standard Nyström method will first be used to approximate the integrals in equation (2.10). Chapter 3 will discuss the implementation and Chapter 4 will provide the results of using the Nyström method.

Chapter 3

Nyström Method - Implementation

Firstly, we will implement a simple Nyström method to solve the problem above. This will demonstrate the effects of high wavenumbers k on the efficiency of a standard method such as this one. Throughout we will be considering a circular scatterer of radius a. The Nyström method is a straightforward approximation of an integral by a quadrature formula. The integral equation is replaced by the approximation equation and the solution reduces to a linear system [1].

We begin by describing the process of converting the problem into the polar parameterization form, enabling us to explain the programming steps. We then produce some numerical results using this method which are provided in Chapter 4.

3.1 Parametrization

We want to solve equation (2.10). We will look at the case for which $\gamma = 0$. As mentioned in Section 2.2, this will not guarantee a unique solution. However this does exclude the integral in which $\Phi(x,y)$ is part of the kernel, since this is undefined at $x=y$, whereas $\partial\Phi(x,y)/\partial n(x)$ is defined at $x=y$. Hence we assume that we will have a unique solution. Future work could allow for an arbitrary γ. Now we will solve

$$\frac{1}{2}\mu_{\text{slow}}(x) - \int_\Gamma \frac{\partial\Phi(x,y)}{\partial n(x)} e^{ik\alpha\cdot(y-x)}\mu_{\text{slow}}(y)\mathrm{d}s(y) = ikn(x)\cdot\alpha. \tag{3.1}$$

We use the polar parameterization in (2.15) and set $\alpha = (d_1, d_2)$, say. Then the normal $n(x)$, as described in Section 2.2, is given by $(-\cos\theta_0, -\sin\theta_0)$ Therefore writing the right hand side of equation (3.1) in terms of this parameterization gives

$$ikn(x)\cdot\alpha = -ik(d_1\cos\theta_0 + d_2\sin\theta_0). \tag{3.2}$$

Next we consider the kernel of the integral given by

$$\frac{\partial\Phi(x,y)}{\partial n(x)} e^{ik\alpha\cdot(y-x)},$$

where $\Phi(x,y) = \frac{i}{4}H_0^{(1)}(k|x-y|)$ in the $2D$ case. We will calculate the normal derivative of this function. First we set $x=(x_1,x_2)$ and $y=(y_1,y_2)$, and find that $|x-y| = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$ also

$$\Phi(x,y) = \frac{i}{4}H_0^{(1)}\left(k\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}\right). \tag{3.3}$$

To find the normal derivative we have

$$\begin{aligned}\frac{\partial\Phi(x,y)}{\partial n(x)} &= n(x)\cdot\nabla_x\Phi(x,y)\\ &= n_1(x)\frac{\partial\Phi(x,y)}{\partial x_1} + n_2(x)\frac{\partial\Phi(x,y)}{\partial x_2}\end{aligned} \tag{3.4}$$

denoting $n(x) = (n_1(x), n_2(x))$. We differentiate (3.3) with respect to x_1, using the fact, from [4], that

$$\frac{\partial}{\partial z} H_0^{(1)}(z) = -H_1^{(1)}(z),$$

and using the chain rule we obtain

$$\frac{\partial \Phi(z)}{\partial x_1} = \frac{i}{4} \frac{\partial z}{\partial x_1} \frac{\partial}{\partial z} H_0^{(1)}(z),$$

where $z = k\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$. We can then find the derivative with respect to x_1 given by

$$\frac{\partial \Phi(x,y)}{\partial x_1} = -\frac{ik(x_1 - y_1)}{4|x-y|} H_1^{(1)}(k|x-y|).$$

Similarly for the derivative with respect to x_2. We want to write these in terms of the polar parameterization, beginning with

$$\begin{aligned} |x-y| &= \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2} \\ &= \sqrt{a^2(\cos\theta_0 - \cos\theta)^2 + a^2(\sin\theta_0 - \sin\theta} \\ &= a\sqrt{2 - 2\cos(\theta - \theta_0)} \\ &= 2a \sin\left|\frac{\theta - \theta_0}{2}\right|. \end{aligned} \tag{3.5}$$

Substituting these details into (3.4) and making use of some trigonometric identities leads to

$$\frac{\partial \Phi(x,y)}{\partial n(x)} = \frac{ik}{4} \sin\left|\frac{\theta-\theta_0}{2}\right| H_1^{(1)}\left(2ak\sin\left|\frac{\theta-\theta_0}{2}\right|\right). \tag{3.6}$$

Now putting the exponential term of the kernel in polar form we obtain

$$\exp\left\{d_1(\cos\theta - \cos\theta_0) + d_2(\sin\theta - \sin\theta_0)\right\}. \tag{3.7}$$

This, together with the previous details calculated means we can express the kernel in the following way:

$$\begin{aligned}\frac{\partial\Phi(x,y)}{\partial n(x)} = \frac{ik}{4}\sin\left|\frac{\theta-\theta_0}{2}\right| H_1^{(1)}\left(2ak\sin\left|\frac{\theta-\theta_0}{2}\right|\right)\times \\ \exp\left\{d_1(\cos\theta-\cos\theta_0)+d_2(\sin\theta-\sin\theta_0)\right\}.\end{aligned} \tag{3.8}$$

Hence we are solving the equation

$$\frac{1}{2}\mu_{\text{slow}}(\theta_0) - \int_0^{2\pi}\tilde{K}(\theta_0,\theta)\mu_{\text{slow}}(\theta)\mathrm{d}\theta = f(\theta_0), \tag{3.9}$$

where

$$f(\theta_0) = -ik(d_1\cos\theta_0 + d_2\sin\theta_0),$$

and

$$\tilde{K}(\theta_0,\theta) = \frac{aik}{4}\sin\left|\frac{\theta-\theta_0}{2}\right| H_1^{(1)}\left(2ak\sin\left|\frac{\theta-\theta_0}{2}\right|\right).$$

Note the radius of the circle, a, appears in the kernel $\tilde{K}(\theta_0,\theta)$ since we are now evaluating the integral over the interval $[0,2\pi]$.

3.2 Implementation

To implement a basic Nyström method we must evaluate the integral using a quadrature rule. Recall that $\tilde{K}(\theta_0,\theta)$ and $\mu_{\text{slow}}(\theta)$ are 2π periodic. We approximate, for $0 \le \theta_0 \le 2\pi$,

$$\frac{1}{2}\mu_{\text{slow}}(\theta_0) - h\sum_{n=1}^{N}\tilde{K}(\theta_0,nh)\mu_{\text{slow}}(nh) = f(\theta_0),$$

applying the trapezoidal rule, where $h = 2\pi/N$. We want to find values for $\mu_{\text{slow}}(jh)$, for $j = 1,2,\ldots,N$. Currently we have one equation with N unknowns. Hence we set equation (3.10) to hold at each of the points $\theta_0 = h, 2h, \ldots, Nh$ and we get a matrix-vector formulation

$$\frac{1}{2}\begin{bmatrix} \mu_{\text{slow}}(h) \\ \mu_{\text{slow}}(2h) \\ \\ \vdots \\ \\ \mu_{\text{slow}}(Nh) \end{bmatrix} - h \begin{bmatrix} K(h,h) & K(h,2h) & \cdots & K(h,Nh) \\ K(2h,h) & K(2h,2h) & \cdots & K(2h,Nh) \\ \\ & \vdots & \vdots & \\ \\ K(Nh,h) & K(Nh,2h) & \cdots & K(Nh,Nh) \end{bmatrix} \begin{bmatrix} \mu_{\text{slow}}(h) \\ \mu_{\text{slow}}(2h) \\ \\ \vdots \\ \\ \mu_{\text{slow}}(Nh) \end{bmatrix}$$

$$= \begin{bmatrix} f(h) \\ f(2h \\ \\ \vdots \\ \\ f(Nh) \end{bmatrix} \tag{3.10}$$

We define approximations to $\mu_{\text{slow}}(jh)$ as μ_j, for $j = 1, \ldots, N$ and simplifying (3.10) we obtain the linear system

$$\begin{bmatrix} \frac{1}{2} - hK(h,h) & -hK(h,2h) & \cdots & -hK(h,Nh) \\ -hK(2h,h) & \frac{1}{2} - hK(2h,2h) & \cdots & -hK(2h,Nh) \\ & & \ddots & \\ -hK(Nh,h) & -hK(Nh,2h) & \cdots & \frac{1}{2} - hK(Nh,Nh) \end{bmatrix} \begin{bmatrix} \mu_1 \\ u_2 \\ \vdots \\ \mu_N \end{bmatrix} = \begin{bmatrix} f(h) \\ f(2h) \\ \vdots \\ f(Nh) \end{bmatrix}. \tag{3.11}$$

We can implement this in MATLAB, by creating functions for each of $\tilde{K}(\theta_0, \theta)$ and the right hand side of equation (3.11), $f(\theta_0, \theta)$, using the details calculated in Section 3.1.

Problem

The Hankel function, $H_1^{(1)}(z)$, of the first order is singular at $z = 0$ thus MATLAB will return an error. This means we must look at the series expansion of this Hankel function to input an approximation for the instances where $z = 0$ occurs. Recall

$$H_1^{(1)}(z) = J_1(z) + iY_1(z) \tag{3.12}$$

from equation (2.2) in Section 2.1. From [4], we know each of the Bessel functions can be expressed as a series. The Bessel function of the first kind can be expressed in the following way

$$J_1(z) = \frac{z}{2}\sum_{k=0}^{\infty} \frac{\left(-\frac{z}{4}\right)^k}{k\Gamma(2+k)}.$$

Expanding this for a few terms gives

$$J_1(z) = \frac{z}{2}(1 + c_1 z^2 + c_2 z^4 + O(z^6)),$$

where c_n are constants. Moreover, the Bessel function of the second kind can be expressed as

$$Y_1(z) = -\frac{2}{z\pi} + \frac{2}{\pi}\ln\left(\frac{z}{2}\right)J_1(z) + O(z)$$

and

$$zY_1(z) = -\frac{2}{\pi} + \frac{2z}{\pi}\ln\left(\frac{z}{2}\right)J_1(z) + O(z^2).$$

We know that $zJ_1(z) = O(z)$ hence

$$zY_1(z) = -\frac{2}{\pi} + \frac{2z^2}{\pi}\ln\left(\frac{z}{2}\right)J_1(z)+)(z^2).$$

Now consider taking limits of $zJ_1(z)$ and $zY_1(z)$ as $z \to 0$ and substituting this into equation (3.12) to find that

$$zH_1(z) = -\frac{2i}{\pi}, \quad \text{as } z \to 0$$

Therefore, using equation (3.8), we approximate the normal derivative of $\Phi(\theta_0, \theta)$ by $-1/4\pi$, at the points where $\theta_0 = \theta$.

Once this has been accounted for in the coding we can look at some results for different wavenumbers k and investigate the effects on the approximation as k increases. This is demonstrated in the following Chapter 4.

Chapter 4

Nyström Method - Results

4.1 Results

In this chapter we present the results obtained using the Nyström method to solve the boundary integral equation (3.1) and calculate errors. We include plots displaying the results alongside observations and finally a brief summary of what has been found using this method.

First recall the notation $\alpha = (d_1, d_2)$ from Chapter 3, where α gives the direction of the incident wave, $u^i(x)$. Throughout the implementation we have assumed, without loss of generality, $\alpha = (1, 0)$ and we have taken the radius of the scatterer to be 1. Also note on the following plots the x-axis is labelled 't' which denotes the discretization points, such that $t = 2\pi/N$, where N is the number of nodes. The y-axis, denoted 'μ', is the approximate solution the MATLAB program produces, we are plotting the absolute value. We investigate the effect of the size of the wavenumber k by plotting approximate solutions to the integral equation for various numbers of nodes, N.

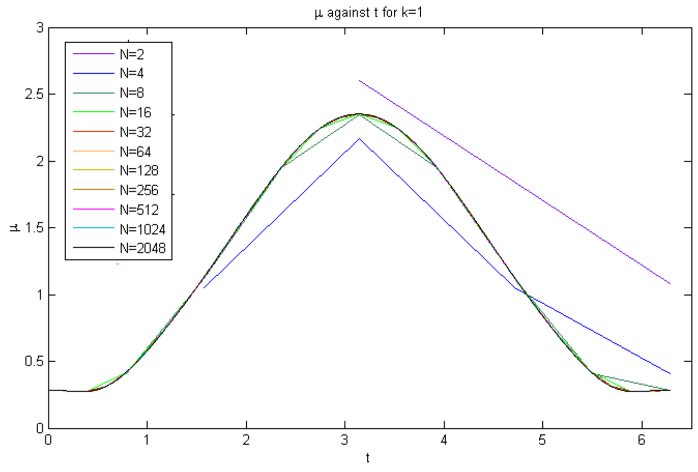

Figure 4.1: plot displaying approximate solution against discretization points for $k = 1$

We see in Figure 4.1 that for $N = 32$ and above, the solution, loosely speaking, appears to converge to the true solution. In comparison, Figures 4.2 and 4.3, for $k = 5$ and $k = 10$ respectively show that the number of nodes required before we notice this "convergence" is much greater.

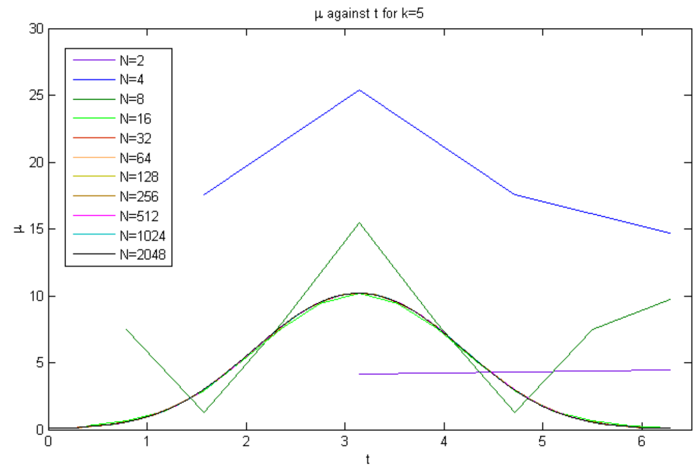

Figure 4.2: plot displaying approximate solution against discretization points for $k = 5$

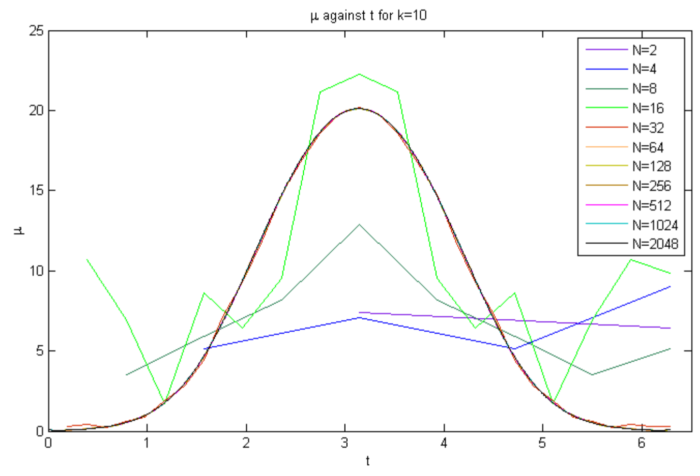

Figure 4.3: plot displaying approximate solution against discretization points for $k = 10$

Now looking at $k = 50$ and $k = 100$, shown in Figure 4.4 and Figure 4.5 respectively, we see that for smaller numbers of nodes, N, the results are spurious and highly oscillatory.

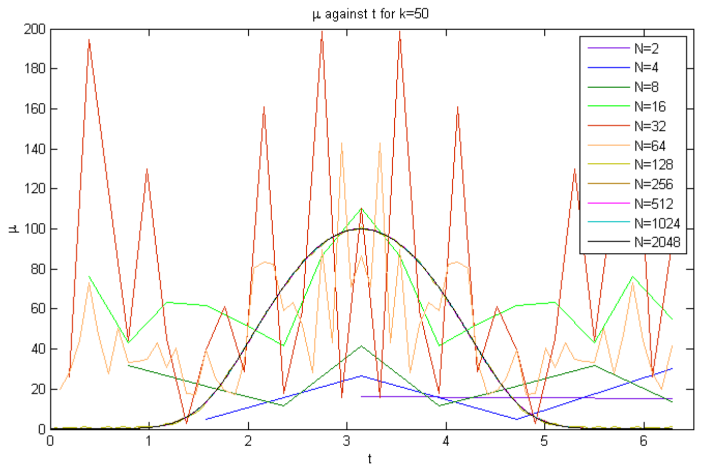

Figure 4.4: plot displaying approximate solution against discretization points for $k = 50$

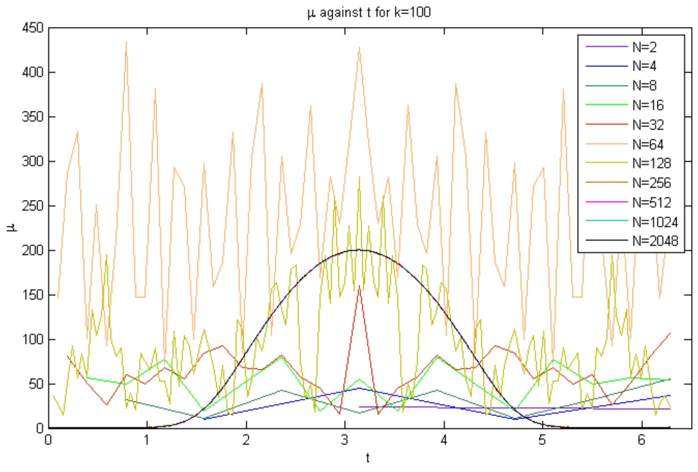

Figure 4.5: plot displaying approximate solution against discretization points for $k = 100$

From Figure 4.7 we find that as k increases the peaks become narrower. Also, the solutions are approaching zero around the shadow boundaries. In other words the points θ_0 such that $\alpha \cdot n(\theta_0) = 0$. Recall $n(\theta_0) = (-\cos\theta_0, -\sin\theta_0)$ and we have taken $\alpha = (1, 0)$. Therefore we want the points $0 \leq \theta_0 \leq 2\pi$ such that $\cos(\theta_0) = 0$. Thus the shadow boundaries are at $\pi/2$ and $3\pi/2$. The fact that the approximation to the solution is tending to zero around these points is expected since as $k \to \infty$,

$$\frac{\partial u}{\partial n} \to 2\frac{\partial u^i}{\partial n}$$

on the illuminated boundary and

$$\frac{\partial u}{\partial n} \to 0 \tag{4.1}$$

on the shadow boundary, as discussed in [3] and depicted in Figure 4.6.

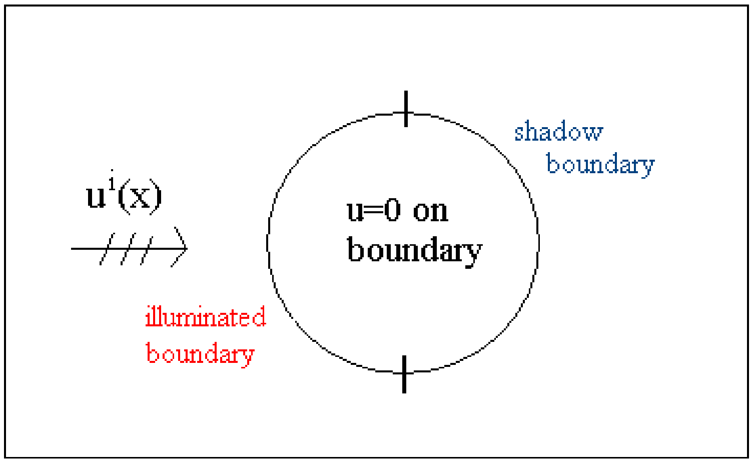

Figure 4.6: Figure depicting the shadow boundaries

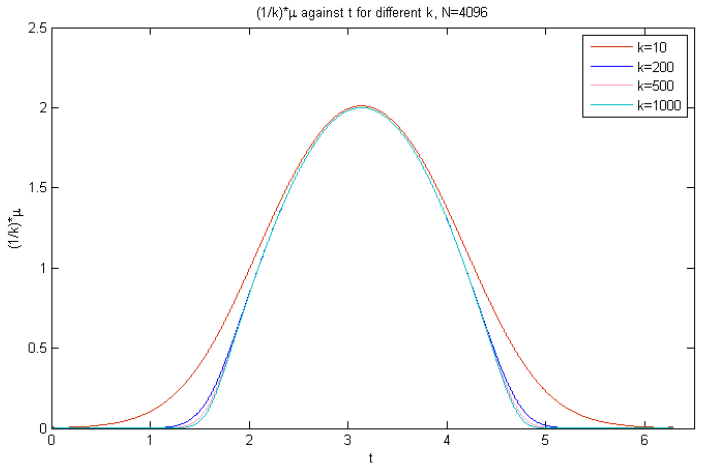

Figure 4.7: plot displaying approximate solution against discretization points, for $N = 4096$ and various k

4.2 Errors

We can further interpret the results by investigating the errors. We calculate the error using the L^2-norm

$$\|u - u_n\|_2 = \left[\int_0^{2\pi} |u - u_n|^2 \right]^{\frac{1}{2}}, \tag{4.2}$$

where u is the exact solution and u_n is the approximation. We approximate this integral using the trapezium rule

$$\|u - u_n\|_2 \approx \sqrt{h \sum |u - u_n|^2}.$$

However, we do not have the exact solution and hence use the values obtained for the largest value of nodes we previously calculated, $N = 2048$. This means a direct comparison of this solution to that of a smaller number of nodes would not be possible. For example, comparing with $N = 4$, would only be possible at 4 points, which would not give us a good estimate for the error between them. Therefore we interpolate for a number of points in between. We could interpolate linearly but we can use a more suitable method for the problem.

In this case we will use a trigonometric interpolating polynomial. In other words, the function going through each of our data points has to be a trigonometric polynomial; a sum of sines and cosines, the general form being

$$p(t) = \sum_{j=1}^{N/2} a_j \cos(jt) + b_j \sin(jt),$$

where a_j, b_j are constants. The MATLAB code used gives us $u_h, u_{2h}, \ldots, u_{Nh}$, which are approximations to $\mu_{\text{slow}}(jh)$ as defined in Section 3.2. To use the trigonometric interpolating polynomial we must have $N = 2m$, even. We have the equally spaced points $t_j = j\pi/N$, $j = 1, \ldots, N$. Referring to [7], the operator P_N is then defined by

$$P_N u(t) = \sum_{s=1}^{N} u(t_j) l_s^{(N)}(t), \tag{4.3}$$

such that

$$l_s^{(N)}(t) = \frac{1}{N}\left\{1 + 2\sum_{n=1}^{m-1} \cos(n(t-t_s)) + \cos(m(t-t_s))\right\}. \tag{4.4}$$

The function $l_s^{(N)}(t_j)$ satisfies the following

$$l_s^{(N)}(t_j) = \begin{cases} 1 & k = j, \\ 0 & k \neq j, \end{cases}$$

as this guarantees the interpolating function passes through the data points we have calculated.

These functions are then implemented using MATLAB, programs for which are shown in appendix A. We obtain the error for a different number of nodes, N, using the L^2-norm shown in equation (4.2), where the interpolated solution for $N = 2048$ is the approximation to the exact solution. Table 4.1 displays the errors calculated in this way.

A plot displaying this data is shown in Figure 4.8. We notice from this that the set of errors follows a similar trend for each k. However, the errors for $k = 1$ and $k = 10$ decrease much more rapidly as the number of nodes increases.

Table 4.1: Table of errors $k = 1, 10, 100$

Error			
N	k=1	k=10	k=100
2	2.3488e+000	2.3222e-001	2.2745e+002
4	3.4438e-001	2.5340e+001	2.3960e+002
8	1.0066e-002	1.8052e+001	2.3644e+002
16	1.0954e-003	2.2949e+001	2.5565e+002
32	1.3354e-004	6.6524e-001	2.6929e+002
64	1.6592e-005	7.2622e-002	6.0345e+002
128	2.0707e-006	8.8045e-003	2.1252e+002
256	2.5827e-007	1.0906e-003	1.2635e+000
512	3.1838e-008	1.3421e-004	9.5934e-002
1024	3.5375e-008	1.4906e-005	9.8394e-003

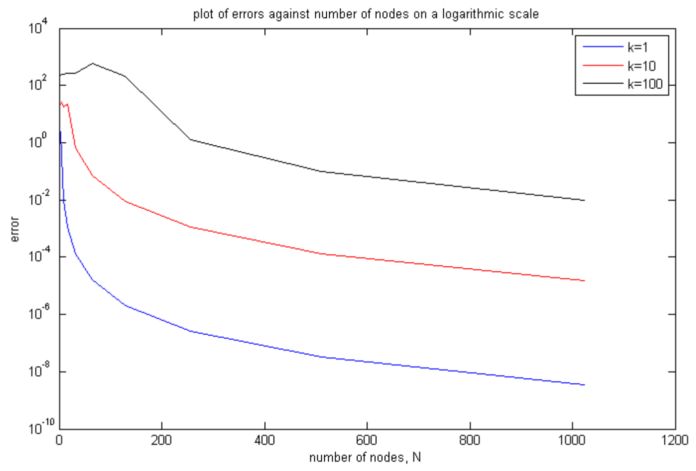

Figure 4.8: plot displaying errors against number of nodes, N, for $k = 1, 10, 100$ on a logarithmic scale

Consider the values in the error Table 4.2, for $k = 10$, for N in between $N = 16$ and $N = 32$. By calculating and plotting errors for intermediate N, we can investigate at which point the error becomes 'small'. We can also determine the number of nodes N needed for the approximation to closely resemble the true solution. Figure 4.9 displays the data in Table 4.2 for $k = 10$. It is clear from this that at around $N = 22$, the error becomes significantly smaller than that of the approximation for $N = 18$ or $N = 20$.

Table 4.2: Table of errors, $k = 10$

N	error k=10	N	error k=10
2	2.3222e+001	22	3.8555e+000
4	2.5340e+001	24	2.2437e+000
6	2.3265e+001	26	1.4164e+000
8	1.8052e+001	28	1.0689e+000
10	1.6944e+001	30	8.3290e-001
12	1.8722e+001	32	606524e-001
14	2.6061e+001	34	5.4192e-001
16	2.2949e+001	36	4.4856e-001
18	5.4806e+001	38	3.7591e-001
20	1.1268e+001	40	3.1831e-001

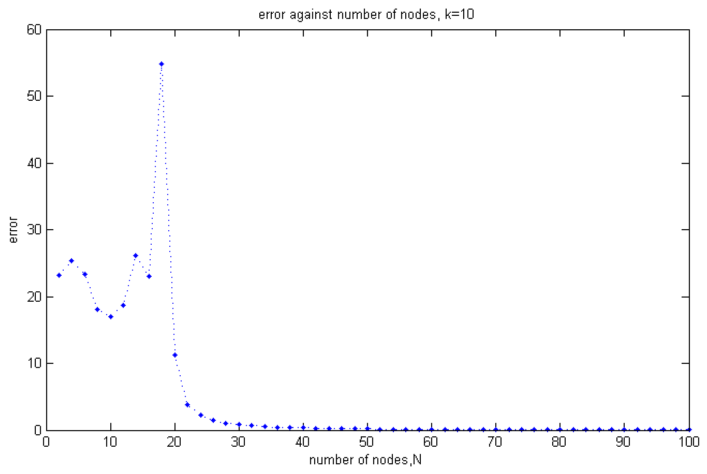

Figure 4.9: plot displaying errors against number of nodes, N, for k=10

We now consider Figure 4.5. It is difficult for the naked eye to distinguish between approximations beyond $N = 256$. However, it is clear the approximation for $N = 128$ is highly oscillatory. Hence we will look at errors for numbers of nodes between these two approximations, determining for which number of nodes the error in the approximation becomes 'small'. Observe Table 4.3 and Figure 4.10 displaying this data. We see that for $N = 208$ the error decrease a lot from that for $N = 176$ or $N = 192$.

Table 4.3: Table of errors, $k = 100$

N	error k=100
128	2.1252e+002
144	1.3420e+002
160	1.2260e+002
176	2.3096e+002
192	6.4333e+001
208	8.7420e+000
224	2.9045e+000
240	1.7988e+000
256	1.2635e+000

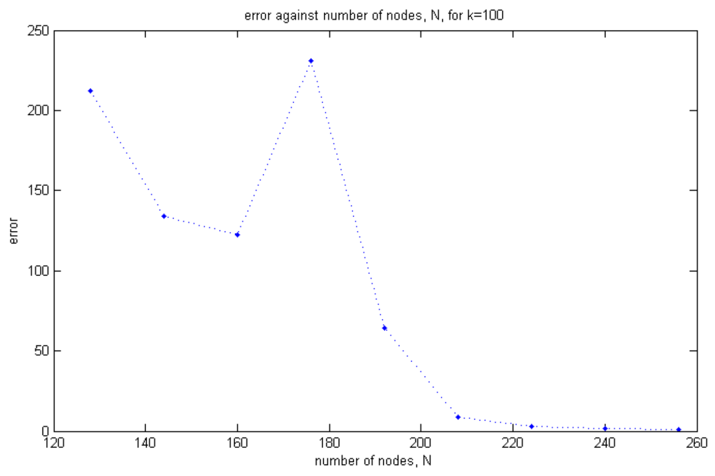

Figure 4.10: plot displaying errors against number of nodes, N, for k=100

Relative errors

We see from Figures 4.1, 4.3 and 4.5 that the number at which the peak of the approximate solution appears depends on k. In other words, we see for $k = 10$, $N = 2048$ the approximate solution reaches a peak with $\mu \approx 20$, whereas for $k = 100$, $N = 2048$, the peak is with $\mu \approx 200$. Therefore to compare the errors more fairly we need to calculate a relative error. This is performed using L^2 norms as follows

$$\text{relative error} = \frac{\|u - u_n\|_2}{\|u\|_2},$$

recalling that u is the 'exact' solution, u_n the approximate solution.

Using the above we obtain Table 4.4. This is depicted in Figure 4.11 based on a logarithmic scale for fairer comparison.

Table 4.4: Table of relative errors, k=1,10,100

N	k=1	k=10	k=100
2	6.061e-001	8.8332e-001	9.0288e-001
4	9.6857e-002	9.6385e-001	9.5111e-001
8	2.8312e-003	6.8665e-001	9.3858e-001
16	3.0808e-004	8.7293e-001	1.0148e+000
32	3.7558e-005	2.5304e-002	1.0690+000
64	4.6665e-006	2.7624e-003	2.3955+000
128	5.8231e-007	3.3490e-004	8.4362e-001
256	7.2637e-008	4.1484e-005	5.0155e-003
512	8.9544e-009	5.1051e-006	3.8082e-004
1024	9.9491e-010	5.6698e-007	3.9059e-005

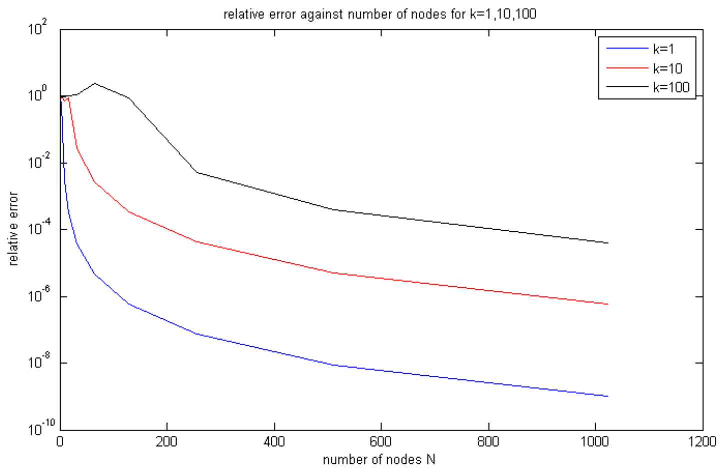

Figure 4.11: plot displaying relative errors against number of nodes for k=1,10,100

From the relative errors we observe the differences between each case of k. For $k = 1$ the errors quickly become within an acceptable 1% of the approximate solution with $N = 2048$, with roughly a 0.04% error at just $N = 16$. However, the errors for $N = 100$, are not within the 1% tolerance until we have used 256 nodes.

4.3 Summary

As we had expected from the literature the number of discretization points needed depends on the size of the wavenumber k. As k increases, the number of nodes, N, needed for a "good" approximation increases proportional to this. For large k, the number of nodes N needed greatly increases and the scheme fails since the computer lacks the required amount of storage. The computational times also greatly increase.

We also found that the scheme behaved as expected at the shadow boundaries. As k increases we suppose the scheme would produce a sharper shape, as described in Section 4.1, equation (4.1). However, to get a "good" approximation for a large wavenumber k the number of nodes required is large. In this case, the scheme fails again and the program is slow for wavenumbers beyond $k = 1000$.

We now consider the errors we have calculated. The errors produced and the plots representing these, confirm the ideas already mentioned here. In particular we notice that that we need at least two degrees of freedom per wavelength before we get an approximation that even remotely resembles the true solution. However we need more than this to get an accurate approximation, therefore the method breaks down for small wavelengths.

Chapter 5

Improved Method

5.1 Implementation

In this chapter we will call upon the theory and ideas discussed in the literature paper [3], as reviewed in Chapter 2. We will use this to describe the steps towards implementing the method presented in [3].

Recall the integral equation (3.1) we want to solve

$$\frac{1}{2}\mu_{\text{slow}}(x) - \int_{\Gamma} \frac{\partial \Phi(x,y)}{\partial n(x)} e^{ik\alpha\cdot(y-x)} \mu_{\text{slow}}(y) \mathrm{d}s(y) = ikn(x)\cdot\alpha. \tag{5.1}$$

In Chapter 3 we presented the standard Nyström method to solve this integral equation along with results of the implementation of this in Chapter 4. Here we parameterized the integral equation to get equation (3.9) and then simply replaced the integral

$$\int_0^{2\pi} \tilde{K}(\theta_0,\theta)\mu_{\text{slow}}(\theta)\mathrm{d}\theta,$$

where $\tilde{K}(\theta_0, \theta)$ is as defined in Chapter 3, by the sum

$$h \sum_{j=1}^{N} \tilde{K}(\theta_0, jh) \mu_{\text{slow}}(jh),$$

where $h = 2\pi/N$. We then set the equation to hold for all $\theta = nh$, for $n = 1, \ldots, N$. More precisely

$$\mu_{\text{slow}}(mh) + h \sum_{j=1}^{N} \tilde{K}(mh, jh) \mu_{\text{slow}}(jh) = f(mh),$$

for $m = 1, \ldots, N$, where the function $f(mh)$ is as defined in Chapter 3, equation (3.9). This gave us a linear system for unknown μ_m defined as an approximation to $\mu_{\text{slow}}(mh)$.

However we know this method has difficulty approximating the integral at high frequencies. Although $\mu_{\text{slow}}(\theta)$ is not oscillatory, the kernel $\tilde{K}(\theta_0, \theta)$ is highly oscillatory, hence the trapezoidal rule cannot approximate accurately.

This means we need a more efficient scheme for evaluating the integral. We now describe the steps needed for the method used in [3]. Again we must first parameterize equation (5.1), using the same polar parameterization we used for the Nyström method, shown in equation (2.15). The formulation of the right hand side is the same as in equation (3.2). Also, we know from Chapter 2 that the kernel of the integral behaves like

$$\begin{aligned} e^{ik[|x-y|+\alpha\cdot(y-x)]} &= e^{ik[|x-y|+\alpha\cdot y]} . e^{-ik\alpha\cdot x} \\ &= e^{ik\psi} . e^{-ik\alpha\cdot x}, \end{aligned} \tag{5.2}$$

where we have denoted $\psi = |x - y| + \alpha \cdot y$. We have written the equation (5.2) in this manner to follow [3], in order to use the formula given by Appendix A of [3] for the critical points of the phase ψ. This formula will be written in the explicit form shortly.

Recall $\alpha = (d_1, d_2)$ so we can put the function ψ into a polar form to get the following:

$$\psi = 2a \sin \left| \frac{\theta - \theta_0}{2} \right| + a(d_1 \cos \theta_0 + d_2 \sin \theta_0). \tag{5.3}$$

We assume $\alpha = (1, 0)$ as used in the MATLAB implementation of the Nyström method, therefore

$$\psi = 2a \sin \left| \frac{\theta - \theta_0}{2} \right| + a \cos \theta_0.$$

As forementioned the critical points for the phase ψ are given in [3] as

$$0 \leq \theta - \theta_0 = \pi - 2\theta_0 + 4n\pi \leq 2\pi$$

or

$$0 \leq \theta - \theta_0 = \frac{1}{3}(\pi - 2\theta_0) + \frac{4}{3}\pi n \leq 2\pi.$$

We can now rewrite the integral equation as

$$\frac{1}{2}\mu_{\text{slow}}(\theta_0) - \int_0^{2\pi} \hat{K}(\theta_0, \theta)\mu_{\text{slow}}(\theta)\mathrm{d}\theta = f(\theta_0), \tag{5.4}$$

where

$$\hat{K}(\theta_0, \theta) = a \exp[ik\psi] \exp[-ika \cos \theta_0] \tag{5.5}$$

and $f(\theta_0)$ is as defined in equation (3.9).

The first step of this improved method is to replace the integral in equation (5.4) by

$$\int_0^{2\pi} \hat{K}(\theta_0, \theta) \left[P_n \mu_{\text{slow}}(\theta)\right] \mathrm{d}\theta, \tag{5.6}$$

recalling from Chapter 4 that $P_n\mu_{\text{slow}}(\theta)$ denotes the trigonometric interpolating polynomial such that

$$P_n\mu_{\text{slow}}(\theta) = \sum_{j=1}^{N} \mu_{\text{slow}}(jh) l_j^N(\theta) \tag{5.7}$$

and $l_j^N(\theta)$ is defined in the same way as equation (4.4). This enables us to only evaluate $\mu_{\text{slow}}(jh)$, for $j = 1, \ldots, N$ whereas $P_n\mu_{\text{slow}}(\theta)$ can be evaluated for any θ. Therefore this is an approximation to the integral in equation (5.4) or more precisely

$$\int_0^{2\pi} \hat{K}(\theta_0, \theta)\mu_{\text{slow}}(\theta)\mathrm{d}\theta \approx \int_0^{2\pi} \hat{K}(\theta_0, \theta) \left[P_n\mu_{\text{slow}}(\theta)\right] \mathrm{d}\theta.$$

We refer to the functions $S(x, x_0, x_1)$ and $f_A(x)$ defined in Section 2.3. We now replace the integrand

$$g(\theta_0, \theta) = \hat{K}(\theta_0, \theta) \left[P_n\mu_{\text{slow}}(\theta)\right]$$

by the sum

$$\sum_{q=1}^{N_s} \chi_q(\theta_0, \theta) g(\theta_0, \theta),$$

where χ_q are mollifying functions of the same form as $f_A(x)$ and N_s is the number of critical points. Each of the mollifying functions have specific widths depending on the type of critical point it envelopes. These are separated into the following three categories:

(i) $\frac{\beta}{\lambda}$ around $\theta_0 = \theta$,

(ii) $\frac{\beta}{\sqrt{\lambda}}$ around stationary points,

(iii) $\frac{\beta}{\sqrt[3]{\lambda}}$ around shadow boundaries, where λ is the wavelength, $\lambda = 2\pi/k$, for some constant β. Substituting into the integral (5.6) gives an approximation

$$\sum_{q=1}^{N_s} \int_0^{2\pi} \chi_q(\theta_0, \theta)\hat{K}(\theta_0, \theta) \left[P_n \mu_{\text{slow}}(\theta)\right] \mathrm{d}\theta.$$

Then using Lemma 2.1 we can localize the integration around the critical points as follows:

$$\sum_{q=1}^{N_s} \int_{C-\varepsilon}^{C+\varepsilon} \chi_q(\theta_0, \theta)\hat{K}(\theta_0, \theta) \left[P_n \mu_{\text{slow}}(\theta)\right] \mathrm{d}\theta,$$

where C is a critical point and 2ε is the small region around the critical point.

These integrals can then be approximated using the trapezoidal rule:

$$\sum_{q=1}^{N_s} \hat{h} \sum_{z=1}^{N_t} \chi_q(\theta_0, z\hat{h})\hat{K}(\theta_0, z\hat{h}) \left[P_n \mu_{\text{slow}}(z\hat{h})\right],$$

where N_t is the number of discretization points used for the trapezoidal rule and $\hat{h} = 2\varepsilon/N_t$. Then substituting for $P_n\mu\text{slow}(z\hat{h})$ using equation (5.7) we obtain the full equation

$$\int_0^{2\pi} \hat{K}(\theta_0, \theta)\mu_{\text{slow}}(\theta)\mathrm{d}\theta \approx \sum_{q=1}^{N_s} \hat{h} \sum_{z=1}^{N_t} \chi_q(\theta_0, z\hat{h})\hat{K}(\theta_0, z\hat{h}) \left[\sum_{j=1}^{N} \mu_{\text{slow}}(jh) l_j^N(z\hat{h})\right].$$

After some rearranging, the right hand side can be written as a general form

$$\sum_{j=1}^{N} G_j(\theta_0)\mu_{\text{slow}}(jh),$$

where

$$G_j(\theta_0) = \sum_{q=1}^{N_s} \hat{h} \sum_{z=1}^{N_t} \chi_q(\theta_0, z\hat{h})\hat{K}(\theta_0, z\hat{h}) l_j^N(z\hat{h}).$$

and recalling that $h = 2\pi/N$.

Using equation (5.4), the whole scheme becomes

$$\frac{1}{2}\mu_{\text{slow}}(\theta_0) - \sum_{j=1}^{N} G_j(\theta_0)\mu_{\text{slow}}(jh) = f(\theta_0)$$

and we set this to hold at $\theta = mh$, as we did for the standard method in Section 3.2. Thus producing to the linear system:

$$\begin{bmatrix} \frac{1}{2} - G_1(h) & -G_2(h) & \cdots & -G_N(h) \\ -G_1(2h) & \frac{1}{2} - G_2(2h) & \cdots & -G_N(2h) \\ \vdots & & & \vdots \\ -G_1(Nh) & -G_2(Nh) & \cdots & \frac{1}{2} - G_N(Nh) \end{bmatrix} \begin{bmatrix} \mu_h \\ \mu_{2h} \\ \vdots \\ \mu_{Nh} \end{bmatrix} = \begin{bmatrix} f(h) \\ f(2h) \\ \vdots \\ f(Nh) \end{bmatrix}, \quad (5.8)$$

where $\mu_h, \mu_{2h}, \ldots, \mu_{Nh}$ are defined as approximations to μ_{slow}.

5.2 Preliminary Tests of Functions

Here we will present how the functions for various components of the implementation work. The MATLAB code for each function is provided in the Appendices.

We begin with the mollifying function

$$f(x) = S(x, x_0, x_1).\,(1 - S(x, -x_1, -x_0))\,,$$

as defined in Chapter 2, Section 2.3. This function envelopes the critical points, localizing the integration. Hence enabling us to integrate over a smaller region.

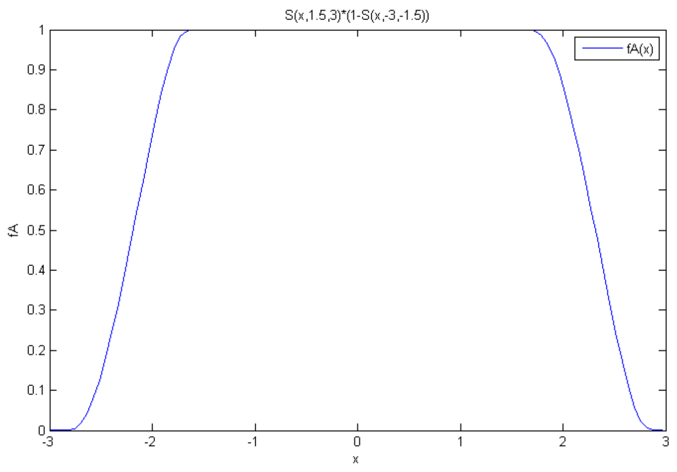

Figure 5.1: Displaying an example of the mollifying function

Figure 5.1 displays an example of the shape of this function. We have used the code of Appendix B. We have simply centred this enveloping function around the origin. This would of course need to be adjusted to centre around the critical point, covering an interval of width as listed in Section 5.1.

We now consider the trigonometric interpolating polynomial as defined in Chapter 4, equation (4.3). The solution points we obtain for the approximation to μ_{slow} are only calculated for each node. This means we cannot easily compare approximate solutions for different numbers of nodes. Hence we used this method of interpolating. We then obtain a set of intermediate points which smoothly join the points at each node produced by MATLAB for our approximate solution. The code in Appendix A has used 5000 points of interpolation but naturally any number can be used depending on the restrictions of the computer.

Figures 5.2 and 5.3 are examples of using the codes in Appendix A.1 and A.2. We can see clearly from these figures that the interpolating polynomial is a smooth function going through each data point from the approximation.

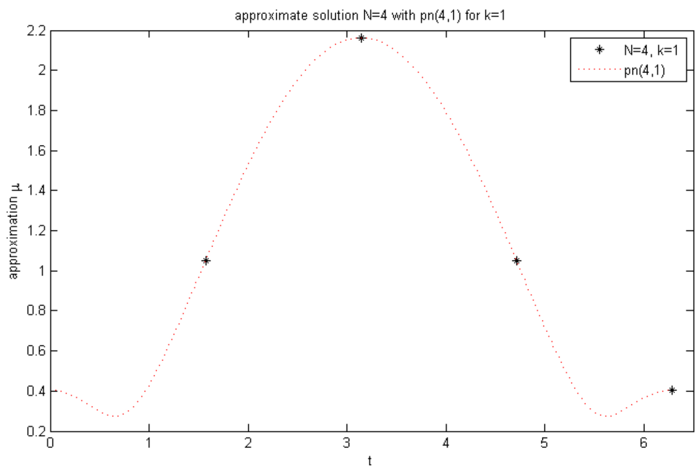

Figure 5.2: Demonstrating the trigonometric interpolating polynomial with N=4, k=1

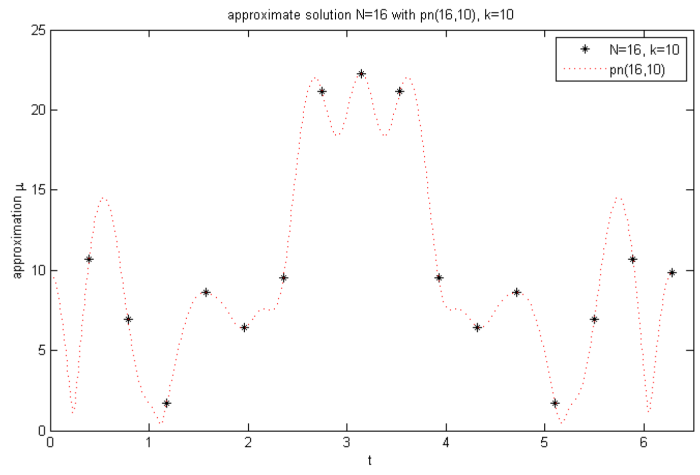

Figure 5.3: Demonstrating the trigonometric interpolating polynomial with N=16, k=10

The codes for the functions $\hat{K}(\theta_0, \theta)$, given in equation (5.5) and $\psi(\theta_0, \theta)$, given in equation (5.3), are also provided in Appendix B.4 and B.3 respectively.

Chapter 6

Summary, Conclusions and Future Work

In this chapter we summarize the work carried out in this dissertation. We indicate the main results and discuss areas for further work on the topic.

6.1 Summary

The main aims of this project were to investigate the motivation behind the method presented by [3], produce some results for a standard numerical method to solve the direct scattering problem and describe the steps to implement the method of [3].

We introduced this dissertation by describing some motivation for studying the problems of numerical evaluation of oscillatory integrals. Chapter 2 provides a literature review which describes the theory and ideas behind the numerical method developed in [3]. We then implemented a standard Nyström method in Chapter 3 and produced results which are presented in Chapter 4. The details for the implementation of the method from the literature were then discussed in Chapter 5. This also contained a brief outline and preliminary testing of functions which can be used for the implementation of the numerical method in [3].

6.2 Conclusions

The results of using the standard Nyström method, in Chapter 4, agreed with the literature. We found that the number of nodes needed for a "good" approximation increased proportional to wavenumber k. We observed in Section 4.3 that we require at least two degrees of freedom per wavelength before we get an approximate solution resembling the true solution to the naked eye. It is clear, from the errors calculated in Chapter 4, that more degrees of freedom per wavelength would be needed to get an accurate solution.

As k increases and more nodes are needed, the computational complexity is much greater and the program failed to provide "good" approximations for k much higher than 2000, due to the number of nodes needed being too great and a shortage of computer memory.

The literature paper [3] had not given the method which it was presenting explicitly and left details of the theory and method of implementing this to be determined by the reader. We have covered areas of the theory in further detail. In addition to this, Chapter 5 provides a firm foundation for further study as it closes the gaps in the details of the implementation omitted by [3].

6.3 Future Work

There is some scope for further work on the Nyström method implemented. Ideally we would want to run the program for larger values of N and k. This was difficult because of the nature of the method as well as the time and computational constraints. Where there was some investigation into the number of nodes for which the approximation becomes within a certain error tolerance, for $k = 10$ and $k = 100$, this could be studied in more depth. There could also be an investigation into circular scatterers of varying radius a. Furthermore a more thorough error analysis could be carried out. This method could also be extended for more complicated convex obstacles so as to build a set of results for various cases which could then be compared to results obtained by the numerical method in [3].

The functions and corresponding MATLAB codes needed for the full implementation of the numerical method in [3] are documented in Chapter 5 and the Appendices A and B. Therefore further study would involve implementing a full program for this numerical method using the details given in this dissertation. Firstly a program which solves the problem as described for the Nyström method, in other words for a circular scatterer of radius a, could be programmed.

Consider Chapter 5, Section 5.1 and the different cases of critical points listed for which the width of the interval covered by the mollifying function is specified. Each type of critical point has a width related to the wavelength λ. In reference to this list in Section 5.1, the constant β could be varied in further investigations. An error analysis of the results produced could then be carried out. Naturally the next step would then be to extend the method for different convex obstacles.

Finally, we can remark that the dissertation has fulfilled the aims as outlined in Chapter 1, Section 1.2.

Bibliography

[1] COLTON D., KRESS R., *Inverse Acoustic and Electromagnetic Wave Scattering*, Springer, 1998, 2nd Edition.

[2] CHANDLER-WILDE S.N., GRAHAM I.G., Boundary integral methods in high frequency scattering. `http://www.reading.ac.uk/nmsruntime/saveasdialog.asp?lID=26518&sID=90309`

[3] BRUNO O.P., GEUZAINE C.A., MONRO J.A, REITICH F., Prescribed error tolerances within fixed computational times for scattering problems of arbitrarily high frequency: the convex case, *Phil. Trans. R. Soc. Lond. A.*,**362**, 2004, 629-645.

[4] ABRAMOWITZ M., STEGUN I.A., *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, Washington, D.C.: U.S. G.P.O., 1965, 4th printing.

[5] `http://www.efunda.com/math/bessel/bessel.cfm`, Bessel Function: Bessels Differential Equation, last checked 28.07.2008.

[6] REED M., SIMON B., *Methods of Modern Mathematical Physics: Scattering Theory*, Volume III, Academic Press, 1979.

[7] LANGDON S., GRAHAM I.G., Boundary integral methods for singularly perturbed boundary value problems, *IMA Journal of Numerical Analysis* **21**, 2001,217-237.

[8] BECKER A.A., *The Boundary Element Method in Engineering: A complete course*,M^{c}Graw-Hall, 1992.

Appendix A

Matlab codes - Errors

A.1 l.m

The following code produces the function $l_j^N(\theta)$ as defined in Chapter 4, equation (4.4). We have denoted N as an even number of nodes.

```
function L = l(y,u,m)

%Lagrange interpolating functions to be used to calculate pn
%N=2m
L=(1/(2*m))*(1+cos(m*(u-y)));

        for n=1:m-1
            L = L + (1/m)*cos(n*(u-y));
        end;
```

A.2 Pn.m

This code produces the trigonometric polynomial function using the code for $l_j^N(\theta)$ as above. The number of points of interpolation has been selected at 5000 but this can be amended to any number of points. The codes for calculating the approximate solution are omitted.

```
%code for trigonometric interpolating polynomial function pn
function Pn = pn(N,k)

a=1; %radius of circle
d1=1; %incident direction
d2=0;

h=2*pi/N; %steplength
m=N/2;
x = mat_A(N,a,k,d1,d2)\rhs_vec(N,k,d1,d2); %our approx solution
Pn = zeros(5000,1);
u = (1:5000)*((2*pi)/5000);

for j=1:5000
    for q=1:N
        Pn(j) = Pn(j) + x(q)*l(q*h,u(j),m);
    end;
end;
```

Appendix B

Matlab codes Improved Method

B.1 S.m

```
function s = S(t,t0,t1)

%t0<t1,     where t0,t1 are scalars to input

s = zeros(size(t));
%creating a vector of zeros, the same size as input vector t

s(t<=t0) = 1;
%where the entries of t are less than t0 replace those entries in vector s
%by 1

select = t>t0 & t<t1;
%choosing entries of t which are between t0 and t1

u=(t(select)-t0)./(t1-t0);
s(select)=exp(2*exp(-1./u)./(u-1));
```

```
%for each entry chosen we produce u and then replace selected
%entries in s with the exponential function calculated using u.
```

B.2 fA.m

```
function func = fA(x,x0,x1)

%to get the function f(x)=S(x,x0,x1)*(1-S(x,-x1,-x0)) as written in
%literature paper Bruno et al.

func = S(x,x0,x1).*(1-S(x,-x1,-x0));
```

B.3 psi.m

```
function p = psi(theta0, theta, a, d1, d2)

%This function is for the total phase.
%psi=abs(x-y)+alpha.y
%alpha=(1,0);
%input:
%theta0       0<theta0<2pi
%theta        0<theta<2pi
%a            radius of the circle
%d1,d2        alpha=(d1,d2), take alpha=(1,0)

p = 2*a*sin(0.5*abs(theta-theta0))+a*(d1*cos(theta0)+d2*sin(theta0));
```

B.4 K_new.m

A function for $\hat{K}(\theta_0, \theta)$ in equation (5.5).

```
function g = K_new(theta0,theta,a,k)


%K=a*exp(i*k*(psi-acos(theta0)))


g = a*exp(i*k*psi(theta0,theta,a,d1,d2))*exp(-i*k*a*cos(theta0));
```