Lightning Helmholtz Solver

Candidate Number: 1060399

University of Oxford

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science in Mathematical Sciences

Trinity 2022

To my parents, Anne and John, and my girlfriend Helen who have all supported me greatly throughout this academic year.

Acknowledgements

I would like express my gratitude to Professor Nick Trefethen for his supervision throughout the duration of this thesis and the academic year as a whole.

Abstract

Gopal and Trefethen introduced "lightning PDE solvers" in 2019 to solve partial differential equations with high speed and accuracy on two-dimensional domains with corner singularities. Extensive progress has been made since for the Laplace equation including the development of `laplace.m`, a lightning Laplace solver. Very little beyond a proof of concept has been established in this regard for the Helmholtz equation. We initiate the development for an analogous lightning Helmholtz solver, `helmholtz.m`. 8-10 digits of accuracy is achieved for many problems and solutions are often computed to 4-digit accuracy or better in less than one second of laptop time.

Contents

1 Introduction **1**

1.1 The Helmholtz equation 1

1.2 Exact solutions of the Helmholtz equation 2

1.2.1 Interior domain 4

1.2.2 Exterior domain 4

1.3 Scattering 5

1.3.1 Sound-soft obstacle 6

1.3.2 Sound-hard obstacle 7

1.4 Approximation theory 7

1.5 Approximating solutions of the Helmholtz equation 10

1.5.1 Condition number 10

1.5.2 Method of fundamental solutions 10

1.5.3 Lightning solvers 12

2 Numerical Implementation **14**

2.1 The algorithm 14

2.1.1 Setting up the problem 15

2.1.2 Choosing the poles 17

2.1.3 Choosing the boundary points 18

2.1.4 Solving the least-squares problem 19

2.1.5 Adaptivity 19

2.1.6 posterior check on a finer mesh 20

2.2 Numerical examples 20

3 Discussion **26**

3.1 Lightning Laplace 26

3.2 Numerical Investigation 27

3.2.1 Adaptive increase of poles 27

3.2.2 Comparison of other forms of the solution 28
3.2.3 Boundary points 28
3.2.4 Runge vs newman points 30
3.2.5 Condition number 30
3.3 Demonstration of physical phenomena 31
3.3.1 Circular L 31
3.3.2 C-shaped domain 32
3.3.3 Diffraction 32
3.3.4 Exterior of the domain 33
3.4 Conclusion 33

A helmholtz.m **38**

Bibliography **49**

List of Figures

1.1 Real part of a basis for the outgoing solution due to 2^n-pole sources . 6
1.2 Scattering from a sound-soft object 7

2.1 Example output of the Helmholtz lightning solver 16
2.2 Demonstration of the steps feature and of the adaptive choice of poles 22
2.3 Boundary matching 23
2.4 Original squares 23
2.5 A variety of examples 24
2.6 Lightning Helmholtz solver for a variety of domains 25
2.7 A domain that fails to achieve root-exponential convergence 25

3.1 Demonstration of the adaptive feature 27
3.2 Comparison of two different forms of the approximation 29
3.3 Linear boundary points 30
3.4 Convergence as a function of Newman and Runge terms 31
3.5 Condition number 32
3.6 Circular L-shape 32
3.7 C-shape 34
3.8 triangle; interesting reflection 35
3.9 Demonstration of diffraction 35
3.10 Effect of wavenumber on shadow region 36
3.11 Two point sources 37
3.12 Behind the region 37

Chapter 1

Introduction

The Helmholtz equation is an elliptic partial differential equation (PDE) that has many applications to physics including accoustics and is named after German physicist Hermann von Helmholtz 1821-1894. The lightning method for solving partial differential equations with exceptional speed and accuracy has shown to be greatly successful since it's introduction in [7]. Plenty of fruitful work has been completed on the lightning method for the Laplace equation and a state-of-the-art implementation has been developed named `laplace.m`, which solves typical problems in less than 1 second on a desktop to 8-digit accuracy. There are many existing methods for numerically solving the Helmholtz equation numerically. The standard two are finite element method and boundary integral equations. This thesis explores our work to develop `helmholtz.m`, an analogous version of LAPLACE.M for solving the Helmholtz equation using the lightning method. This is a much harder problem than that for Laplace and not much work has been done in this regard so far. Our implementation are in MATLAB [15] and experiments were performed in 16-digit floating-point arithmetic.

1.1 The Helmholtz equation

Our journey begins with the scalar/acoustic wave equation

$$\left(\Delta - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right) U(\mathbf{x}, t) = 0 \tag{1.1}$$

where $\Delta := \sum_{i=1}^{n} \frac{\partial^2}{\partial \mathbf{x}_i^2}$ is the laplace operator, U is called the *wave function* which depends on space and time coordinates, and c is the *wavespeed*. We consider time-harmonic solutions of eq. (1.1), that is, solutions of the form

$$U(\mathbf{x}, t) = u(\mathbf{x}) e^{-i\omega t} \tag{1.2}$$

which varies periodically with time t and angular frequency ω. Substituting eq. (1.2) into eq. (1.1) provides us with the *Helmholtz equation* (also called the reduced wave equation)

$$\Delta u + k^2 u = 0 \tag{1.3}$$

where $k = \omega/c$ is fixed and known as the *wave number*.

1.2 Exact solutions of the Helmholtz equation

We concern ourselves with solving eq. (1.3) in two dimensions [11]. To do so, we apply separation of variables in polar coordinates (r, θ), defined by

$$\mathbf{x} = (x, y), \quad x = r\cos\theta, \quad y = r\sin\theta, \quad r = \sqrt{x^2 + y^2}$$

In this coordinate system, the Helmholtz equation becomes

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} + k^2 u(r, \theta) = 0 \tag{1.4}$$

By assuming u is of the form $u(r, \theta) = R(r)\Theta(\theta)$ and subbing in to eq. (1.4), we get

$$\Theta(\theta)\frac{d^2 R}{dr^2} + \frac{\Theta(\theta)}{r}\frac{dR}{dr} + \frac{R(r)}{r^2}\frac{d^2\Theta}{d\theta^2} + k^2 R(r)\Theta(\theta) = 0 \tag{1.5}$$

Dividing by $u(r, \theta)$, multiplying by r^2, and rearranging terms provides us with

$$\frac{r^2}{R(r)}\frac{d^2 R}{dr^2} + \frac{r}{R(r)}\frac{dR}{dr} + k^2 r^2 = -\frac{1}{\Theta(\theta)}\frac{d^2\Theta}{d\theta^2} \tag{1.6}$$

The left-hand side of eq. (1.6) depends only on r whereas the right-hand side depends only on θ, so both sides must be equal to a constant, say ν^2, which provides us with the following two equations

$$\frac{r^2}{R(r)}\frac{d^2 R}{dr^2} + \frac{r}{R(r)}\frac{dR}{dr} + k^2 r^2 = \nu^2 \tag{1.7}$$

$$-\frac{1}{\Theta(\theta)}\frac{d^2\Theta}{d\theta^2} = \nu^2 \tag{1.8}$$

multiplying eqs. (1.7) and (1.8) by $R(r)$ and $-\Theta(\theta)$ respectively and rearranging provides

$$r^2\frac{d^2 R}{dr^2} + r\frac{dR}{dr} + (k^2 r^2 - \nu^2)R(r) = 0 \tag{1.9}$$

$$\frac{d^2\Theta}{d\theta^2} + \nu^2\Theta(\theta) = 0 \tag{1.10}$$

The solution to eq. (1.10), $\Theta(\theta)$ is a linear combination of $e^{\pm i\nu\theta}$. We seek solutions with $\Theta(\theta)$ being 2π-peridodic in θ so we constrain the constant $\nu = n$ such that $n \in \mathbb{Z}$.

Concerning eq. (1.9), we first consider solutions for arbitrary ν. By substituting $z = kr$ and setting $R(r) = B(z)$, eq. (1.9) can be rewritten as

$$z^2\frac{d^2B}{dz^2} + z\frac{dB}{dz} + (z^2 - \nu^2)B(z) = 0 \tag{1.11}$$

which is *Bessel's differential equation.* Being a second-order linear differential equation, eq. (1.11) has two linearly independent solutions.

Definition 1 (The standard Bessel functions)**.** The standard solutions to Bessel's equation are known as the *Bessel functions.* The *Bessel function of the first kind of order* ν, denoted $J_\nu(z)$ is defined as follows [14, Appendix B]

$$J_\nu(z) := \sum_{m=0}^{\infty} \frac{(-1)^m}{\Gamma(m+1)\Gamma(m+\nu+1)} \left(\frac{z}{2}\right)^{2m+\nu} \tag{1.12}$$

We note that Bessel functions of the first kind of integer order are finite at the origin $z = 0$ and are in fact entire (of order n), that is, analytic on the whole complex plane.

The *Bessel function of the second kind of order* ν, denoted $Y_\nu(z)$ are solutions that have a branch point singularity at the origin $z = 0$. They are also known as Neumann functions or Weber functions and are defined by the following for non-integer values of ν

$$Y_\nu(z) := \frac{J_n(z)\cos(n\pi) - J_{-n}(z)}{\sin(n\pi)} \tag{1.13}$$

For an integer $n \in \mathbb{Z}$, $Y_n(z)$ is defined by the limit

$$Y_n(z) := \lim_{\alpha \to n} Y_\alpha(z) \tag{1.14}$$

With this formulation, we deduce that linear combinations of the functions $J_n(kr)$ and $Y_n(kr)$ provide solutions $R(r)$ to eq. (1.9). Therefore $J_0(kr)$ and $Y_0(kr)$ are linearly independent solutions to eq. (1.4) for $n = 0$ and the following are linearly independent solutions for each $n \in \mathbb{Z}_{\neq 0}$

$$J_n(kr)e^{in\theta}, \quad J_n(kr)e^{-in\theta}, \quad Y_n(kr)e^{in\theta}, \quad Y_n(kr)e^{-in\theta}$$

Solutions to eq. (1.4) can be constructed by taking linear combinations of these terms. However, the solutions we allow will depend on the domain of the problem.

We note that the following property of Bessel functions [5, Eq. 10.4.1] allows us to restrict our attention to $n \in \mathbb{Z}_{\geq 0}$ rather than $\mathbb{Z}$

$$J_{-n}(z) = (-1)^n J_n(z), \quad Y_{-n}(z) = (-1)^n Y_n(z), \quad n \in \mathbb{Z} \tag{1.15}$$

Equation (1.16) now takes the following form

$$J_n(z) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!(m+n)!} \left(\frac{z}{2}\right)^{2m+n} \tag{1.16}$$

1.2.1 Interior domain

Suppose we wish to solve the Helmholtz equation on the interior of a disc, that is to find $u(r, \theta)$ that satisfies eq. (1.4) where $0 \leq r \leq R$ for some $R > 0$. The physically relevant solution will be finite at the origin $r = 0$. The function J_n is finite at the origin whereas Y_n has a singularity, so we would seek a solution of the form

$$u(r, \theta) = a_0 J_0(kr) + \sum_{n=1}^{\infty} J_n(kr)(a_n^+ e^{in\theta} + a_n^- e^{-in\theta}) \tag{1.17}$$

for some a priori unknown coefficients a_0 and $a_n^{\pm}, n \in \mathbb{N}$.

1.2.2 Exterior domain

To solve the problem on an exterior domain, there must be a condition imposed at infinity. Introduced in 1912 by Arnold Sommerfeld [18], the physically relevant condition is the *Sommerfeld radiation condition* which ensures that the solution corresponds to a wave that is propogating outwards. Without such a condition, the solution may not be unique, making the problem ill-posed. In two dimensions the Sommerfeld radiation condition takes the form

$$\lim_{r\to\infty} \sqrt{r} \left(\frac{\partial}{\partial r} - ik\right) u(r, \theta) = 0 \quad \text{uniformly in } \theta \tag{1.18}$$

A solution that satisfies such condition is said to be *radiating*. None of the independent solutions $J_n(kr)e^{\pm in\theta}, Y_n(kr)e^{\pm in\theta}$ satisfy the Sommerfeld radiation condition, but the linear combination $(J_n(kr) + iY_n(kr))e^{\pm in\theta}$ does, which motivates the following formulation

$$H_n^{(1)}(z) := J_n(z) + iY_n(z), \quad n \in \mathbb{Z}_{\geq 0} \tag{1.19}$$

which is called the *Hankel function of the first kind of order n*, named after Hermann Hankel. Since J_n is finite at the origin whereas Y_n has a singularity, $H_n^{(1)}$ has a singularity at the origin $r = 0$. Being a linear combination of the independent solutions of Bessel's equation by definition, $H_n^{(1)}(z)$ clearly solves Bessel's equation too. Therefore, the following functions are linearly independent outgoing solutions of the Helmholtz equation, eq. (1.4)

$$H_0^{(1)}(kr), \quad H_n^{(1)}(kr)e^{in\theta}, \quad H_n^{(1)}(kr)e^{-in\theta}, \quad n \in \mathbb{N} \tag{1.20}$$

Thus, the outgoing solutions to the Helmholtz equation on an exterior domain are of the form

$$u(r,\theta) = b_0 H_0^{(1)}(kr) + \sum_{n=1}^{\infty} H_n^{(1)}(kr)(b_n^+ e^{in\theta} + b_n^- e^{-in\theta}) \tag{1.21}$$

for some a priori unknown coefficients b_0 and $b_n^{\pm}, n \in \mathbb{N}$. This type of series is called a *multipole expansion*. The function $H_0^{(1)}(kr)$ represents the field due to a monopole source at the origin $r = 0$. $H_1^{(1)}(kr)e^{i\theta}$ represents the field due to a dipole source etc. The higher order solutions $H_n^{(1)}(kr)e^{in\theta}$, $n \in \mathbb{N}$ represent the field due to multipole sources [11].

Figure 1.1 shows plots the real part of the first basis function of the outgoing solution due to 2^n-pole sources i.e $\mathrm{Re}(H_0^{(1)}(|z|)e^{in\theta}), n = 1, \ldots, 6$. The figures are plotted with wave number $k = 1$ for simplicity. We can see how $H_0^{(1)}$ is finite at the origin and that $H_n^{(1)}$ has a singularity for $n \in \mathbb{N}$. The figure also demonstrates the reduction in geometric symmetry as n increases, and the increasing strength of the singularity at the origin for $n \in \mathbb{N}$. The real part of the other basis function $\mathrm{Re}(H_n^{(1)}(|z|)e^{-in\theta})$ is a π rotation of the plotted function around the z-axis. A linear combination of these two functions can then represent the outgoing solution due to 2^n-pole sources.

Remark. We briefly note that an equivalent formulation can be described for solutioins to the time-harmonic wave equation of the form $U(\mathbf{x}, t) = \mathrm{Re}(e^{i\omega t}u(\mathbf{x}))$ where the solutions of the Helmholtz equation would involve the following alternative linear combination that satisfies the Sommerfeld radiation condition

$$H_n^{(2)} := J_n(z) - iY_n(z), \quad z > 0, n \in \mathbb{Z}_{\geq 0} \tag{1.22}$$

which is known as the *Hankel function of the second kind of order n*.

1.3 Scattering

The perhaps most relevant physical problem that the Helmholtz equation can represent is the scattering of a time-harmonic wave from an obstacle. Consider a bounded obstacle $D \subset \mathbb{R}^2$ and a given incident field u^i that satisfies the Helmholtz equation. We wish to find the scattered field u^s such that the total field $u^t = u^i + u^s$ satisfies the Helmholtz equation. The total field u^t must also satisfy a boundary condition for the problem to have a unique solution and be well-posed. Since this is an exterior problem, the solution must satisfy a condition at infinity as discussed in section 1.2.2. That is, the scattered field u^s must satisfy the Sommerfeld radiation condition, eq. (1.18)

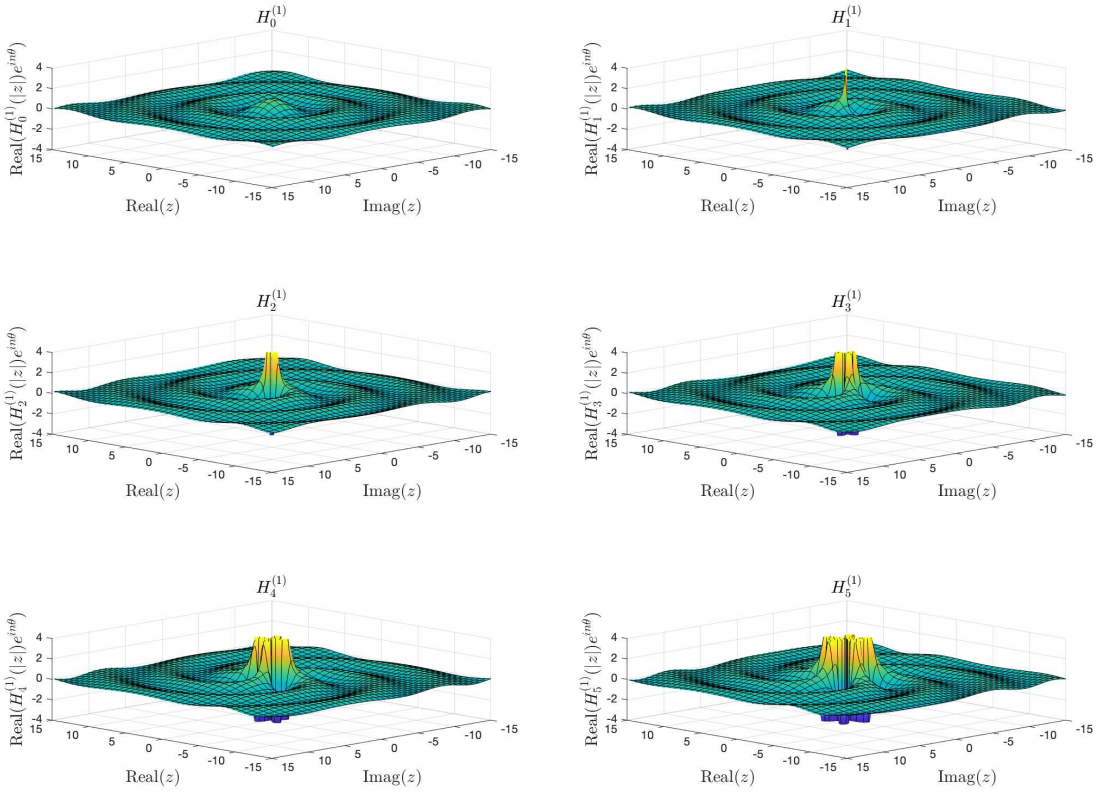

Figure 1.1: Real part of a basis for the outgoing solution due to 2^n-pole sources

to ensure outward propogation. The boundary condition for the total field u^t will reflect the physical properties of the obstacle D, we describe two common cases in this section [17].

1.3.1 Sound-soft obstacle

Scattering of an incident field from a soft-sound object is represented by the condition that the total field vanishes at the boundary. That is, the total field satisfies the homogeneous Dirichlet boundary condition

$$u^t(z) = 0, \quad z \in \partial D \tag{1.23}$$

Figure 1.2 demonstrates the decomposition of the total field u^t into the incident and scattered fields, u^i and u^s respectively.[1]

The first image shows the field of a point source wave incident from the upper right hand corner. The scattering of the incident field from the square object is shown

[1] A video demonstrating the evolution of the time-harmonic wave equation for a sound-soft scattering problem can be found at `https://youtu.be/jxcDfvGnU8A`

in the second image. Finally, the last image shows the total combined field, a linear combination of the incident and scattered fields. The red arrows indicate the direction of propogation of the fields. The incident field is incoming whereas the scattered wave is outgoing from the object.

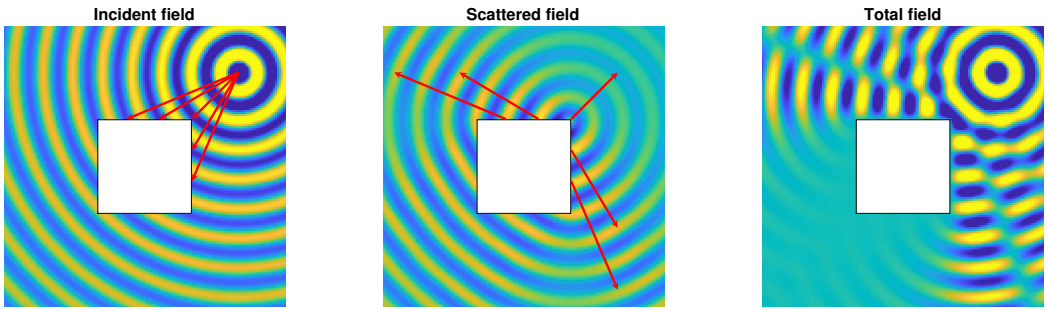

Figure 1.2: Scattering from a sound-soft object

1.3.2 Sound-hard obstacle

Scattering of an incident field from a sound-hard object is captured by the condition that there is no change in the total field in the direction normal to the surface at the boundary. That is, the total field satisfies the homogeneous Neumann boundary condition

$$\frac{\partial u^t}{\partial n} = 0, \quad z \in \partial\Omega \tag{1.24}$$

where n is the unit normal pointing outward from the boundary $\partial\Omega$. This boundary condition corresponds to a perfectly reflective obstacle.

1.4 Approximation theory

Obtaining numerical results to mathematical problems hinges upon computers which execute algorithms. So far, all of the mathematics that we have discussed has been continuous, that is the problems have involved real or complex variables. In a 1992 essay, Trefethen proposed the following definition [19]

> "Numerical analysis is the study of algorithms
> for the problems of continuous mathematics."

Computers use algorithms to obtain such results and since they have finite memory, for most problems they must approximate. As such, approximation theory has a

substantial grounding in numerical analysis. In fact, in almost all areas of numerical analysis the discussion resolves to approximation theory eventually [9, Sec. IV.21].

Approximation theory concerns the construction of approximations to general functions with simpler functions such as polynomials or rational functions. Donald Newman established a pivotal result for the basis of our work in 1964 by considering the supremum norm approximation to $f(x) = |x|$ on $[-1, 1]$ with polynomials and rational functions [16]. Newman showed that degree n polynomial approximations at best obtain convergence at the rate of $\|f - p_n\|_\infty = \mathcal{O}(n^{-1})$ as $n \to \infty$ whereas degree n rational functions can attain $\|f - r_n\|_\infty = \mathcal{O}(e^{-C\sqrt{n}})$ for $C > 0$, so-called *root-exponential convergence*. Newman achieved this by placing poles of the approximaant in the complex plane clustered exponentially near the singularity at $x = 0$. This section states theorems introduced by Gopal and Trefethan [8] which establish the existence of root-exponentially good approximants for a wide range of problems.

Definition 2 (rational function)**.** A rational function of *type* (m, n) is a function $r \in \mathcal{R}_{mn}$ that can be written as $r = p_m/q_n$ where $p_m \in \mathcal{P}_m$ and $q_n \in \mathcal{P}_n$ are polynomials of degree m and n respectively.

$$r(x) = \sum_{k=0}^{m} a_k x^k \Bigg/ \sum_{k=0}^{n} b_k x^k \tag{1.25}$$

The degree of r is $\deg(r) = \max\{m, n\}$.

For our purposes, the form $r = p/q$ for rational functions will be inconvenient. It will be more appropriate to consider rational functions in their partial fraction representation, guaranteed by the following theorem [20, Chapter 23], [10, Chapter 7]

Theorem 1 (Partial fraction representation)**.** *Let $r \in \mathcal{R}_{mn}$ be a rational function of type (m, n) for $m, n \geq 0$. Then r has a unique representation of the form*

$$r(x) = p_0(x) + \sum_{k=1}^{\mu} p_k((x - \xi_k)^{-1}) \tag{1.26}$$

$$p_k((x - \xi_k)^{-1}) = \sum_{j=1}^{\nu_k} \frac{a_{k,j}}{(x - \xi_k)^j} \tag{1.27}$$

where p_0 is a polynomial of exact degree ν_0 for some $\nu_0 \leq m$ (unless $p = 0$) and $\{p_k\}$, $1 \leq k \leq \mu$, are polynomials of exact degrees $\nu_k \geq 1$ with $p_k(0) = 0$ and $\sum_{k=1}^{\mu} \nu_k \leq n$.

The function p_0 is the *polynomial part* of r, and $p_k((x - \xi_k)^{-1})$ is it's *principal part* at ξ_k.

Theorem 2. *Consider the slice-of-pie region*

$$A_\theta = \{z \in \mathbb{C} : |z| < 1, -\theta < \arg(z) < \theta\} \tag{1.28}$$

Let f be a bounded analytic function in the slit disk A_π that satisfies $f(z) = \mathcal{O}(|z|^\delta)$ as $z \to 0$ for some $\delta > 0$, and let $\theta \in (0, \pi/2)$ be fixed. Then for some $\rho \in (0,1)$ depending on θ, but not f, there exist type $(n-1, n)$ rational functions $\{r_n\}, 1 \le n < \infty$, such that

$$\|f - r_n\|_\Omega = \mathcal{O}(e^{-C\sqrt{n}}) \tag{1.29}$$

as $n \to \infty$ for some $C > 0$, where $\Omega = \rho A_\theta$ and $\|\cdot\|_\Omega$ denotes the supremum norm over Ω. Moreover, each r_n can be taken to have simple poles only at

$$\beta_j = -e^{-\sigma_j/\sqrt{n}}, \quad 0 \le j \le n-1 \tag{1.30}$$

where $\sigma > 0$ is arbitrary.

Theorem 2 establishes root-exponential resolution of corner singularities. The following theorem builds upon this to establish root-exponential convergence for global approximations in a convex polygon Ω by adding together local pieces to resolve the corner singularities and a polynomial term to handle smooth components away from the corners.

Theorem 3. *Let Ω be a convex polygon with corners $w_1, \ldots w_m$, and let f be an analytic function in Ω that is analytic on the interior of each side segment and can be analytically continued to a disk near each w_k with a slit along the exterior bisector there. Assume f satisfies $f(z) - f(w_k) = \mathcal{O}(|z - w_k|^\delta)$ as $z \to w_k$ for each k for some $\delta > 0$. There exist degree n rational functions $\{r_n\}, 1 \le n < \infty$, such that*

$$\|f - r_n\|_\Omega = \mathcal{O}(e^{-C\sqrt{n}}) \tag{1.31}$$

as $n \to \infty$ for some $C > 0$. Moreover, each r_n can be taken to have finite poles only at points exponentially clustered along the exterior bisectors at the corners, with arbitraty clustering parameter σ as in eq. (1.30), as long as the number of poles near each w_k grows at least in proportion to n as $n \to \infty$.

Proofs of theorems 2 and 3 can be found in [8]. To prove theorem 3 without the assumption of convexity is quite tricky. However, Gopal and Trefethan believe that theorems 2 and 3 remain valid without the assumptions of $\theta < \pi/2$ and convexity respectively.

1.5 Approximating solutions of the Helmholtz equation

There are many well-studied methods for numerically solving partial differential equations (PDEs) in general, namely boundary integral equations, boundary element method, method of fundamental solutions. Of these, the method of fundamental solutions will be of most interest to us.

We concern ourselves with constructing approximants to the unique solution u of the homogenous Helmholtz equation on a two-dimensional domain Ω exterior to polygon P such that the inhomogenous Dirichlet boundary condition is satisfied. That is, we wish to approximate u that satisfies the following

$$\Delta u(z) + k^2 u(z) = 0, \qquad z \in \Omega \tag{1.32}$$

$$u(z) = h(z), \quad z \in P \tag{1.33}$$

where we use complex notation $z = x + iy$ for simplicity as before.

1.5.1 Condition number

Conditioning refers to the sensitivity of a problem to purturbations in inputs. For a system of linear equations $Ax = b$, conditioning is often measured by the *matrix condition number*, $\kappa(A) = \|A\|\|A^{-1}\|$. For the 2-norm, this can be equaivalently be defined as the ratio of the largest singular value to the smallest singular value of the matrix A. If the matrix condition number is high, the system is said to be *ill-conditioned* whereas if the matrix condition number is low, the system is said to be *well-conditioned*. The methods we will discuss in detail often involve matrices that are highly ill-conditioned

1.5.2 Method of fundamental solutions

Consider a general boundary-valued problem governed by the following

$$Lu(z) = 0, \qquad z \in \Omega \subseteq \mathbb{C} \tag{1.34}$$

$$Bu(z) = h(z), \quad z \in \partial\Omega \subseteq \mathbb{C} \tag{1.35}$$

where L is a linear elliptic partial differential operator, Ω is an open, bounded region with boundary $\partial\Omega$, and the boundary conditions are specified by the operator B and function h together. We introduce the method of fundamental solutions with the following definition

Definition 3 (fundamental solution)**.** A fundamental solution G of a linear partial differential operator L is a solution of the following inhomogenous equation

$$LG(z, z_j) = -\delta(z, z_j) \quad z, z_j \in \mathbb{C} \tag{1.36}$$

where δ is the Dirac delta distribution [5, Eq. 1.16(iii)]. We note that G is a priori only assumed to be a distribution and that $G(z, z_j)$ is singular for $z = z_j$ so the points z_j are called singular points or source terms which are usually monopole. Furthermore, the fundamental solutions of a differential operator is not unique.

The idea of the method of fundamental solutions (MFS) was introduced by Kupradze and Aleksidze in 1964 [12]. It has since been developed to approximate solutions of elliptic partial differential equations for which fundamental solutions exist [6]. This includes the Laplace equation [1] and the Helmholtz equation [2, 13]. The method is meshless since it does not involve discretization nor integration. The idea of the MFS is to approximate a solution to the boundary-value problem by a linear combination of fundamental solutions that originate from source points outside of the domain. Specifically, calculate approximants u_N that satisfy eq. (1.34) as follows

$$u(z) \approx u_N(z) := \sum_{j=1}^{N} a_j G(z, z_j), \quad z \in \overline{\Omega} \tag{1.37}$$

where $G(z, z_j)$ are fundamental solutions of L, $\{z_j\}_{j=1}^N$ are singularities which are placed outside the closure of the domain $\overline{\Omega} := \Omega \cup \partial\Omega$, and $\{a_j\}_{j=1}^N$ are a priori unknown coefficients. The singularities, $\{z_j\}_{j=1}^N$ can be either fixed a priori or determined along with the coefficients $\{a_i\}_{i=1}^N$. If the singularities are to be determined, the resulting minimisation problem is nonlinear whereas if they are fixed a priori then the problem becomes linear. The approximant u_N clearly satisfies eq. (1.34) since $\{z_j\}_{j=1}^N \cap \overline{\Omega} = \emptyset$. To satisfy the boundary condition, M boundary points $\{z_i : z_i \in \partial\Omega\}_{i=1}^M$ are chosen. It is typical to take $M \gg N$ to take advantage of the overcomplete basis and solve for the coefficients by a least squares fit.

We return to the Helmholtz problem and describe MFS approximants the function u that solves eqs. (1.32) and (1.33). Recall from section 1.2 that $H_0^{(1)}(kr)$ is an outgoing solution of the Helmholtz equation and has a singularity at the origin. More generally, $H_0^{(1)}(k|z - z_j|)$ represents the field due to a monopole source located at the point z_j. In fact, fundamental solutions of the Helmholtz equation are given by [3]

$$G(z, z_j) = \frac{i}{4} H_0^{(1)}(k|z - z_j|) \tag{1.38}$$

and so the method of fundamental solutions for the Helmholtz problem seeks approximate solutions u_N of the form

$$u(z) \approx u_N(z) := \frac{i}{4} \sum_{j=1}^{N} a_j H_0^{(1)}(k|z - z_j|), \quad z \in \Omega,\ z_j \in \mathbb{C} \setminus \overline{\Omega} \tag{1.39}$$

where N is the number of charge points z_j. This approximation can be described as a linear combination of monopole soruces located at z_j. The strengths of each source is determined by the fitting of the boundary data, matching the sampled values collectively as best as possible.

1.5.3 Lightning solvers

Lightning solvers are a variant of the method of fundamental solutions which was introduced by Gopal and Trefethen [8]. Since it's introduction, the method has primarily been applied to solving the Laplace equation, a special case of the Helmholtz equation for $k = 0$. However for the Helmholtz equation interesting problems now require complex expansion coefficients to match boundary data. The method has also been applied to Stokes flow [4] and some potential flow? and some preliminary experiments have already been underway for the Helmholtz equation [7].

The purpose of section 1.4 was to motivate the idea behind the lightning method which we introduce for the exterior Helmholtz problem. The lightning method approximates the solution u that satisfies eqs. (1.32) and (1.33) with a rational function. As discussed in section 1.5.2, if the singularities (or source terms) are not fixed a priori then approximating a solution with the MFS becomes a difficult nonlinear problem. The function u has singularities at the vertices of P and so motivated by Newman's result, the lightning method prescribes complex singularities of the approximant in the interior of P a priori in a configuration that is exponentially clustered near each vertex. This pivotal construction allows our method to obtain root-exponentially convergent approximations of the solution i.e errors $O(\exp(-C\sqrt{n}))$ with $C > 0$. Moreover, we refer to these source points as *lightning poles* due to their analogy with lightning rods and the "lightning fast" rate of convergence they provide. With additional terms to deal with the smoother part of the domain, our rational approximation

of u takes the form

$$u(z) \approx \overbrace{\sum_{j=1}^{N_1} H_1^{(1)}(k|w_j|)\left(a_j \operatorname{Re}\frac{w_j}{|w_j|} + b_j \operatorname{Im}\frac{w_j}{|w_j|}\right)}^{\text{"NEWMAN"}} + \overbrace{+ c_0 H_0^{(1)}(k|w_*|) + \sum_{j=1}^{N_2} H_j^{(1)}(k|w_*|)\left(c_j \operatorname{Re}\frac{w_*^j}{|w_*^j|} + d_j \operatorname{Im}\frac{w_*^j}{|w_*^j|}\right)}^{\text{"RUNGE"}} \tag{1.40}$$

where $\{a_j\}_{j=1}^{N_1}, \{b_j\}_{j=1}^{N_1}, \{c_j\}_{j=0}^{N_2}, \{d_j\}_{j=1}^{N_2}$ are a priori unknown complex coefficients ($N = 2N_1 + 2N_2 + 1$ in total), $w_j = z - z_j$ and $w_* = z - z_*$ such that $\{z_j\}$ are the poles of the approximant and $z_* \in \overline{P}$ is the polynomial expansion point, fixed roughly in the centre of the interior of the polygon P.

We describe the two series in eq. (1.40) as *"Newman"* and *"Runge"* respectively, similar to [7], though the analogy isn't quite as direct for our case. The first series captures the idea motivated by Newman's result and resolves the singularities at the vertices. The second series represents the fact that polynomials are effective approximants for the smooth part of the problem, an idea that goes back to Runge in 1885.

The appearance of $H_n^{(1)}$ terms can be explained by the fact that we know that they correspond to outgoing solutions (of?) since they satisfy the Sommerfeld radiation condition. The Hankel functions in the Newman series are of order 1 to represent dipole source terms, analogous to the lightning approximation to the Laplace equation [7].

The complex coefficients $\{a_j\}_{j=1}^{N_1}, \{b_j\}_{j=1}^{N_1}, \{c_j\}_{j=0}^{N_2}, \{d_j\}_{j=1}^{N_2}$ are determined such that a least-squares best fit of the boundary data is obtained across a sample of M boundary points. As is usual with the MFS, $M \gg N$ for the lightning method as well. However unlike the MFS,the boundary points are exponentially clustered near the vertices of P to track the effect of the singularities being placed in a similar manner.

Multipole expansions The Runge series can be thought of as a decomposition of a multipole expansion around the point $z_* \in \overline{\Omega}$ Figure 1.1 shows the multipole sources for the first 6 sources. The solutions are going to be linear combinations of these terms such that the boundary conditions are satisfied.

Chapter 2

Numerical Implementation

In this chapter we provide the details and illustrate examples of our *"lightning Helmholtz solver"* algorithm, `helmholtz.m`. Equation () on a domain exterior to a polygon P for small or medium wavenumbers $k > 0$ with the lightning method introduced in the previous chapter. Our implementation is carried out in MATLAB and is largly based on the *"lightning Laplace solver"*, LAPLACE.M. We work with the following incident fields for sampling the boundary in our experiments
propogating plane wave from an angle θ

$$u^i_\theta(z) = -\exp\left(-i\mathrm{Re}\left[kze^{-i\theta}\right]\right) \quad \theta \in [0, 2\pi) \tag{2.1}$$

where θ is the incident angle of propogation
point source radiating from a point z_*

$$u^i_{z_*}(z) = -H^{(1)}_0\left(k|z - z_*|\right) \quad z_* \in \Omega \tag{2.2}$$

where z_* is the location of the point source.

We work with Dirichlet boundary conditions $u(z) = h(z), z \in P$ for a boundary sampling function h.

2.1 The algorithm

In this section we outline the details of our implementation of the lightning method for solving the Helmholtz equation. Very robust with extensive user inputs, has an adaptive procedure for placing poles, can be applied to a wide variety of problems. Figure 2.1 shows an example of the two figures that our algorithm produces. The first figure consists of four plots showing the convergence, error on the boundary, pole distribution, and a plot of the solution. The convergence curve shows the error as

measured by the supremum-norm of the least-squares grid on the boundary on a log scale against $\sqrt{N}$ where N is the total degrees of freedom.

The plot of the error on the boundary displays the error of the boundary points in black and the error of the a posterior finer mesh check in red.

The plot of the pole distribution shows the total number of poles allocated to each corner of the polygon for the final solution. The angles are measured between the point wc and the corner with respect to the real axis.

The plot of the solution on the right hand side is of the real part of the constructed total field due to the solved scattering. The title of the plot shows the wavenumber for the problem, the dimensions of the matrix used in the least-squares problem and the number of poles that were prescribed. The number of columns of the matrix A corresponds to the total degrees of freedom N.

The solution is obtained through an iterative procedure in which the number of poles, Runge points, and boundary points are increased until convergence to a specified tolerance level is obtained. A pseudocode of our algorithm is described by algorithm 1.

2.1.1 Setting up the problem

a Lines 1 to 3: The first task for the algorithm is to treat the inputs. A user must specify a domain Ω, a wavenumber k, and indicate whether the incident field will be a plane or point source unless boundary data is inputted. While much of our focus is on the scattering problem, our lightning Helmholtz solver can handle any valid user-specified boundary sampling function. In fact, the boundary data h can be specified pointwise for each side of the domain. Users also have the option to specify the tolerance ϵ, surpress 3D plotting, surpress all plots, plot only the scattered field, specify the font size, turn off adaptive mode, and specify either the direction of incident propogation θ for a plane wave or the point of oscillation/propogation z_* for a point source.

Regarding the domain, a user can specify either a polygon by setting the vertices $w_1, \ldots, w_V$, a circular polygon by further specifying convex or concave radii of curvature between desired pairs of consecutive points, a random polygon with a specified number of sides, a random circular polygon, or lastly, choosing one of the inbuilt domains. A characteristic length scale L associated with the domain Ω is defined by

$$L = \max\left(\max_{w\in\partial\Omega} \mathrm{Re}(w) - \min_{w\in\partial\Omega} \mathrm{Re}(w),\ \max_{w\in\partial\Omega} \mathrm{Im}(w) - \min_{w\in\partial\Omega} \mathrm{Im}(w)\right) \tag{2.3}$$

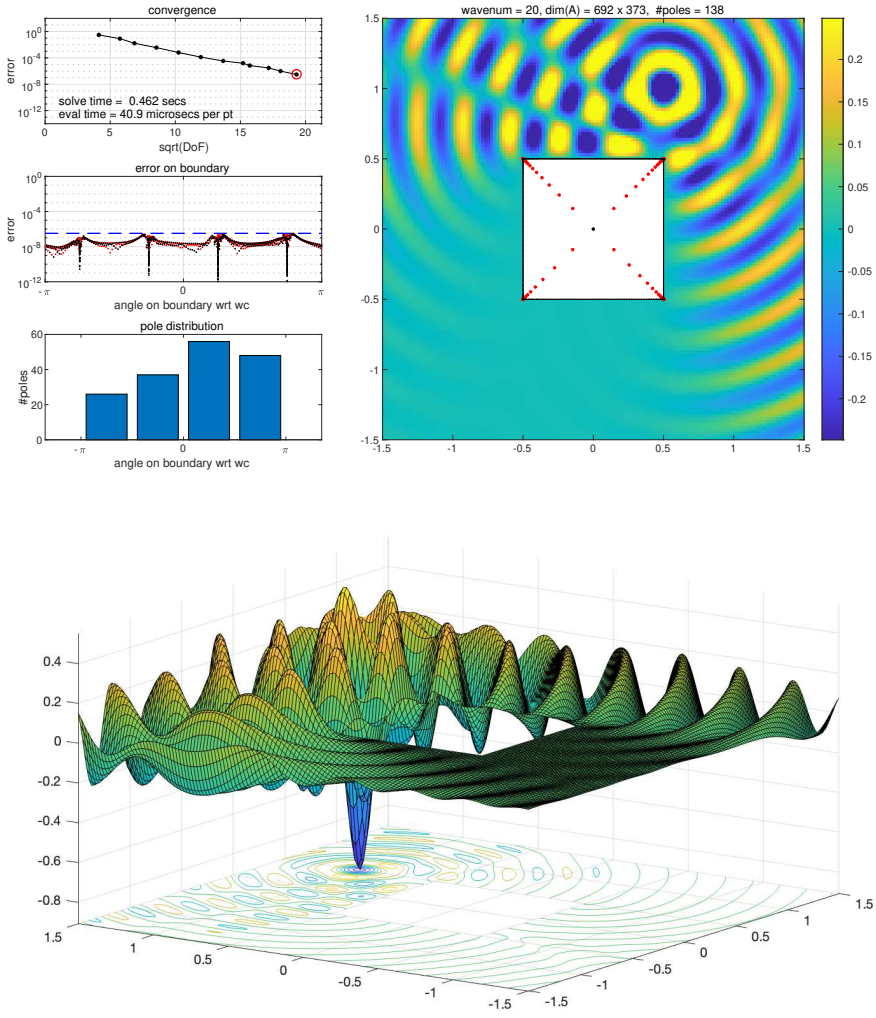

Figure 2.1: Example output of the Helmholtz lightning solver

The Runge point z_* is placed at the centroid of the interior of the boundary $\partial\Omega$ which is guaranteed to be in the interior if it is convex. If $\partial\Omega$ is not convex and the centroid is not in the interior then the user is requested to provide z_*. Lastly, the error is initialised at infinity, the initial number of poles per vertex are fixed, and failure stopping conditions are set. The problem is then solved in steps until either convergence within the tolerance or a failiure stopping condition is fulfilled.

Algorithm 1: Lightning Helmholtz Solver

```
1  Define wavenumber k, boundary ∂Ω, corners w_1, ... w_V, boundary sampling
     function h, tolerance ε;
2  Define characteristic length L, interior point z_*;
3  Set initial error e^(0) ← ∞, initial number of poles at each corner, maximum
     step number t_max, and maximum degrees of freedom;
4  for t ← 1 to t_max do
5      for v ← 1 to V do
6          Fix n poles, {β_vn^(t)} for corner w_v, exponentially clustered;
7          Choose M_v^(t) boundary sampling points;
8      Set number of Runge terms, N_2^(t);
9      Form M × N? matrix A and vector b;
10     Solve the least-squares problem arg min_x ||Ax − b||_2;
11     Calculate the local and global errors: e_v^(t), e^(t);
12     Increase the number of poles at each corner;
13     if e^(t) < ε then
14         break;
15     if e^(t) < e^(0) then
16         save the best result so far and set e^(0) ← e^(t);
17     if t = t_max or N too large then
18         error, failed to converge, return best
19 Estimate maximum error by doing an a posteriori error check on finer mesh;
```

2.1.2 Choosing the poles

Line 6: Let $N_1^{(t)} := N_{1,1}^{(t)} + \cdots + N_{1,V}^{(t)}$ be the decomposition of the number of poles at each vertex at step t. We prescribe the poles at step t according to

$$\beta_{vn}^{(t)} = w_v + \frac{1}{2} L e^{i\theta_v} \alpha_{vn}^{(t)} / \max_l(\alpha_{vl}^{(t)}), \quad \alpha_{vn}^{(t)} = e^{-\sigma(\sqrt{N_{1,v}^{(t)}} - \sqrt{n})}, \quad v = 1, \ldots, V, \quad n = 1, \ldots, N_{1,v}^{(t)} \tag{2.4}$$

We note that poles are placed with stronger exponential clustering here than they are for Laplace. This resulted in a speed-up for the Helmholtz problem. where θ_v is the angle of the interior bisector of corner w_v with respect to the real axis and L is the length scale associated with the domain Ω. We choose $\sigma = 4$. (explore this maybe). The number of poles associated to each vertex is increased in steps. Figure 2.2 demonstrates the adaptive feature of our algorithm. shows the placement of the poles throughout the steps of the algorithm. We can also see the adaptive feature working here

2.1.3 Choosing the boundary points

line 7: As mentioned in section 1.5.3 boundary points are placed exponentially clustered near the vertices to resolve the poles. For every pole z_j near a vertex w_v six boundary points are defined, three on either side of the vertex at distances $\delta/3$, $2\delta/3$, and δ away from w_v where $\delta_{v,j} := |z_j - w_v|$

To resolve the exponentially clustered singularities near the vertices, boundary points are also three boundary points placed on both sides of the of vertex for every pole.

In contrast to the implementation for lightning laplace, we found the problem to be sensitive to the number of linearly distributed boundary points. So we place the linear points on both sides of the vertex to make it more symmetric.

Two types of boundary points are set, exponentially clustered and linearly spaced points. To describe the procedure, consider the decomposition of the two types of points into M_1 and M_2 To describe the locations of the boundary points

UNLIKE LAPLACE, WE ADD THE LINEAR POINTS TO BOTH SIDES OF THE CORNER FOR EACH POINT WHICH MAKES IT MORE SYMMETRIC contrastwe found it beneficial to place

decompose the boundary points int clustered and linearly spaced points $M = M_1 + M_2$. $M_1^{(t)} = M_{1,1}^{(t)} + \cdots + M_{1,V}^{(t)}$. $M_2^{(t)} = M_{2,1}^{(t)} + \cdots + M_{2,V}^{(t)}$ similar to the (sectionref), decompose the number of boundary points as $M^{(t)} = M_1^{(t)} + \cdots + M_V^{(t)}$. Furthermore, decompose the number of points at each corner into $M_v^{(t)} =$

the boundary points are clustered exponentially near the corners

$$M_{1,v}^{(t)} = \begin{cases} N_{1,v}^{(t)} & \text{if not curve} \\ 3N_{1,v}^{(t)} & \text{curved} \end{cases} \tag{2.5}$$

defining $\delta_{vn}^{(t)} = \frac{1}{2}|w_v - w_{v+1}|\alpha_{vn}^{(t)}$.

Through numerical experimenation, we found that curved sides benefited from extra, stronger exponentially clustered points so points are placed also at $2\delta/3, 1\delta/3$ if the side is curved. We found that including these terms for all sides lead to unnecessary slow down.

we found this better than placing the points at the same positions of the poles like it is done in laplac.m this is because when we placed the poles, we used 'greased helmholtz' which does even stronger clustering of the poles but we dont want this to be the case for the boundary points too.

line 8 The number of Runge terms is increased given by

$$N_2^{(t)} = 4t \tag{2.6}$$

2.1.4 Solving the least-squares problem

Lines 9 and 10: We outline in more detail the form of the matrix $A \in \mathbb{C}^{M\times N}$ used in our calculations

$$A_{ij} = \begin{cases} H_0(k|Z_i - z_j|) & 1 \le j \le N_1 \\ H_1(k|Z_i - z_j|)\dfrac{Z_i - z_j}{|Z_i - z_j|} & N_1 < j \le 2N_1 \\ H_j(k|Z_i - z_*|)\dfrac{Z - z_*}{|Z_i - z_*|} & 2N_1 < j \le 2N_1 + N_2 \end{cases} \tag{2.7}$$

note: check if cauchy-like better fits this matrix and also check the actual calculations

the least-squares problem is then set up and the coefficients are solved to cancel out an incident signal?

The problem is then solved using MATLAB's backslash operator which employs QR decomposition with column-pivoting.

Figure 2.3 shows an example of the real values of the field that has to be cancelled at the boundary. The plot on the left hand side is for a plane wave and the right hand side is for a point source, both of wave length 50. the coefficients are chosen as to cancel this incident field.

The local error at vertex w_v is defined by

$$e_v = \max_{i\in\mathcal{I}_v} |r_i|, \quad \text{where } r = Ax - b \tag{2.8}$$

and the global error is defined by

$$e = \|Ax - b\|_\infty \tag{2.9}$$

2.1.5 Adaptivity

Line 12: This feature has been adapted from `laplace.m` adaptively increase the number of poles at corners which have a greater error throughout the iterations. This refers to line 12

an index set for boundary points representing the indicex of the closest corner for that point each corner we associate an index set of the boundary points that are closest to that corner?

$$\mathcal{I}_v = \left\{ 1 \le j \le M : v = \arg\min_{1\le l\le V} |z_j - w_l| \right\}, \quad v = 1, \dots V \tag{2.10}$$

The increase in the number of poles at vertex k depends on the ratio of the local error to the global error

$$N_{1,v}^{(t+1)} = \begin{cases} N_{1,v}^{(t)} \left\lceil 1 + \sqrt{N_{1,v}^{(t)}} \right\rceil & \text{if } e_v^{(t)}/e^{(t)} > q \text{ and } N_{1,v}^{(t)} < \text{maxpols} \\ \max\left(N_{1,v}^{(t)}, \left\lceil \frac{t}{2} \right\rceil\right) & \text{otherwise} \end{cases} \tag{2.11}$$

where $N_{1,v}^{(t)}$ is the number of poles at corner v at step t. All of our experiments were performed using the adaptive feature with $q = 0.5$ unless stated otherwise. If the *slow feature* is activated then $q = 0$.

2.1.6 posterior check on a finer mesh

Line 19 We evalutate on a finor mesh to obtain a pseudo a posteriori estimate for the maximum error but... To obtain a rigorous a posteriori estimate, a modification of the maximum principle would be required.

2.2 Numerical examples

In this section we demonstrate various examples of our algorithm for Helmholtz scattering. We explore the implementation of the lightning helmholtz solver through the sound-soft scattering problem around an object bounded by a polygon. Our lightning Helmholtz solver can solve the homogeonous sound-soft Helmholtz scattering problem around an object. For a given incident field that satisfies the Helmholtz equation, we wish to find u^s such that

$$\Delta u^s + k^2 u^s = 0, \quad z \in \Omega \tag{2.12}$$

$$u^s + u^i = 0, \quad z \in \partial\Omega \tag{2.13}$$

The lightning Helmholtz solver can solve the general homogeneous Helmholtz problem with inhomogenous boundary conditions $u(z) = h(z)$. This corresponds to taking $h = -u^i$ in our algorithm as described. We construct the total field $u^t = u^i + u^s$ and find the corresponding time-independent wave equation $U(z,t) = \mathrm{Re}\,[e^{-i\omega t}u^t(z)]$. We typically plot the solution to the wave equation at $t = 0$.

We first demonstrate our results for the plane wave u_θ^i with angle 30° and point source $u_{z_*}^i$ radiating from $z_* = 1/2 + 1i$. This can be seen in fig. 3.8 In this section

we first compare our results to that of when the method was first proposed for the Helmholtz equation in (ref) both cases were computed with adaptive increase of poles.

The solution is accurate to 9 and 10 digits of accuracy for the plane and point wave sources respectively. The straight line in the middle plot shows the root-exponential convergence after the k-dependent resolution of the wave. If the tolerance is relaxed to 10^{-4}, the solve time is 0.310s and 0.534s for the plane and point sources respectively. (and maybe for 10^{-6}? This is quicker than the 2 seconds for the original paper.

Figure 2.6 shows a variety of domains for the scattering problem with wavenmber $k = 50$

2.6 all diagrams are for wavenumber $k = 50$

Figure 2.7 shows an example of a region for which the algorithm has not converged very well. In this example, the domain contains a narrow slit which may cause the fail.

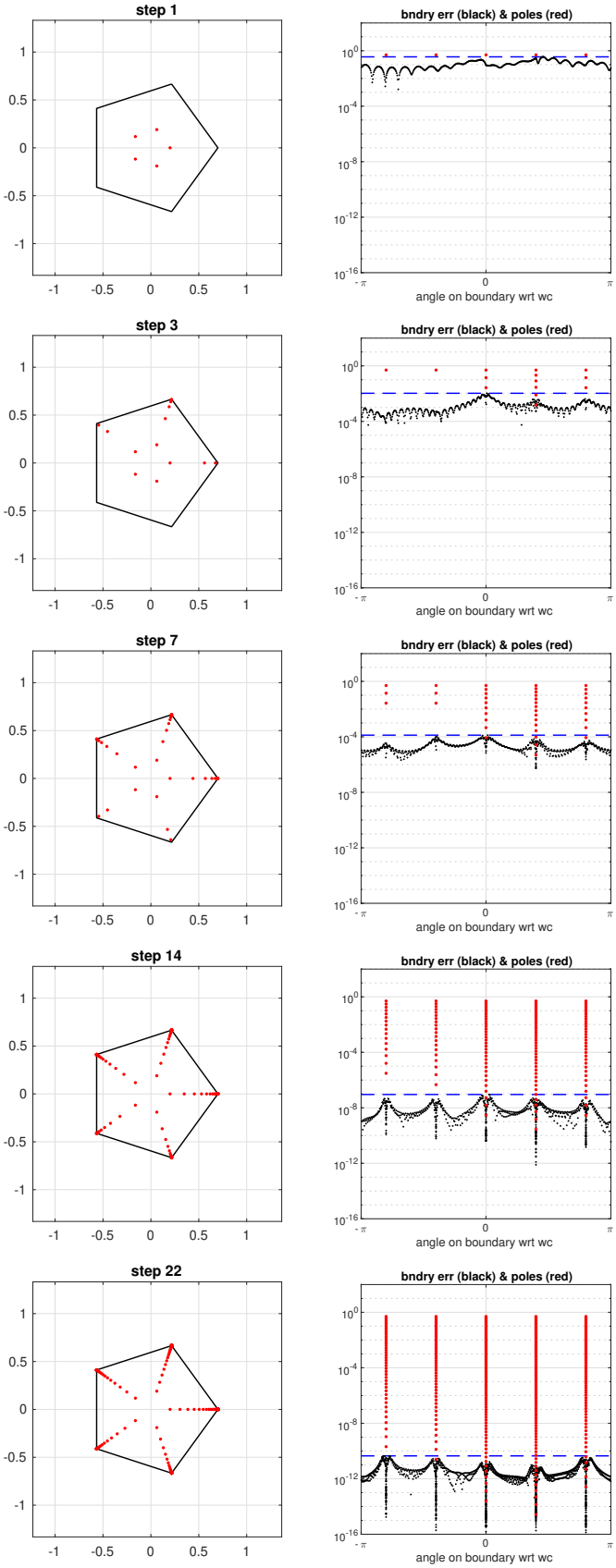

Figure 2.2: Demonstration of the steps feature and of the adaptive choice of poles

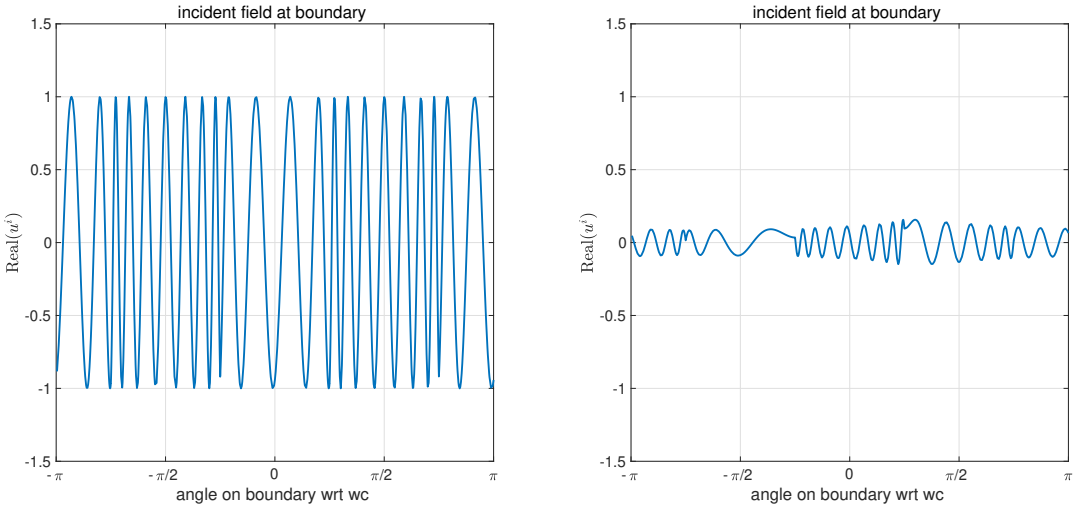

Figure 2.3: Boundary matching

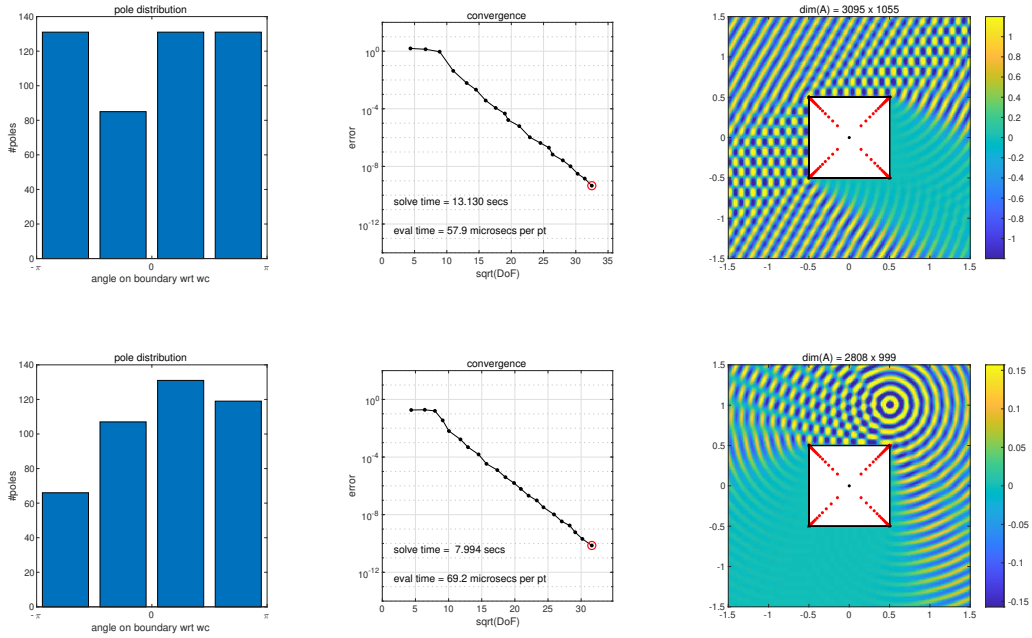

Figure 2.4: Original squares

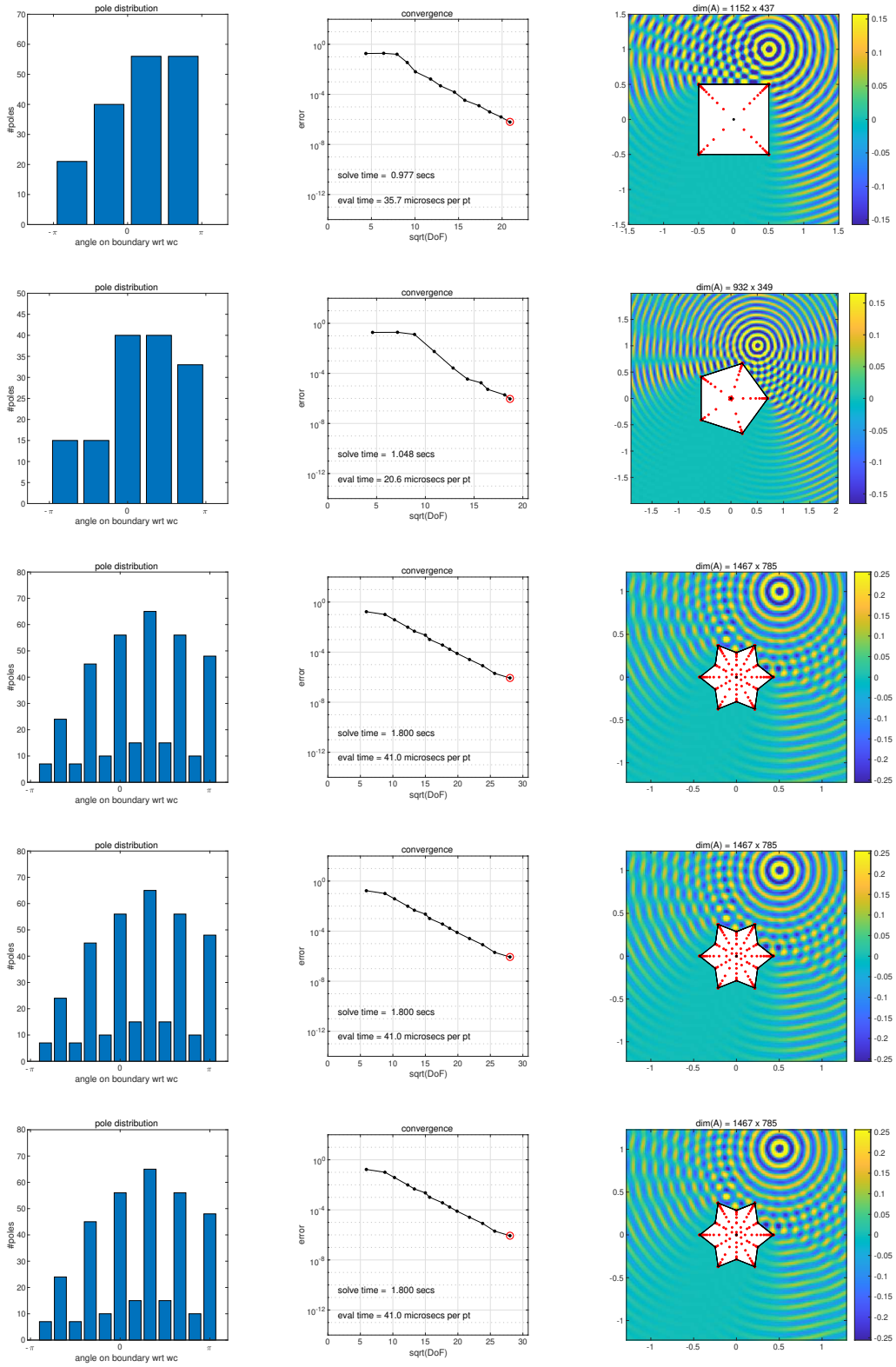

Figure 2.5: A variety of examples

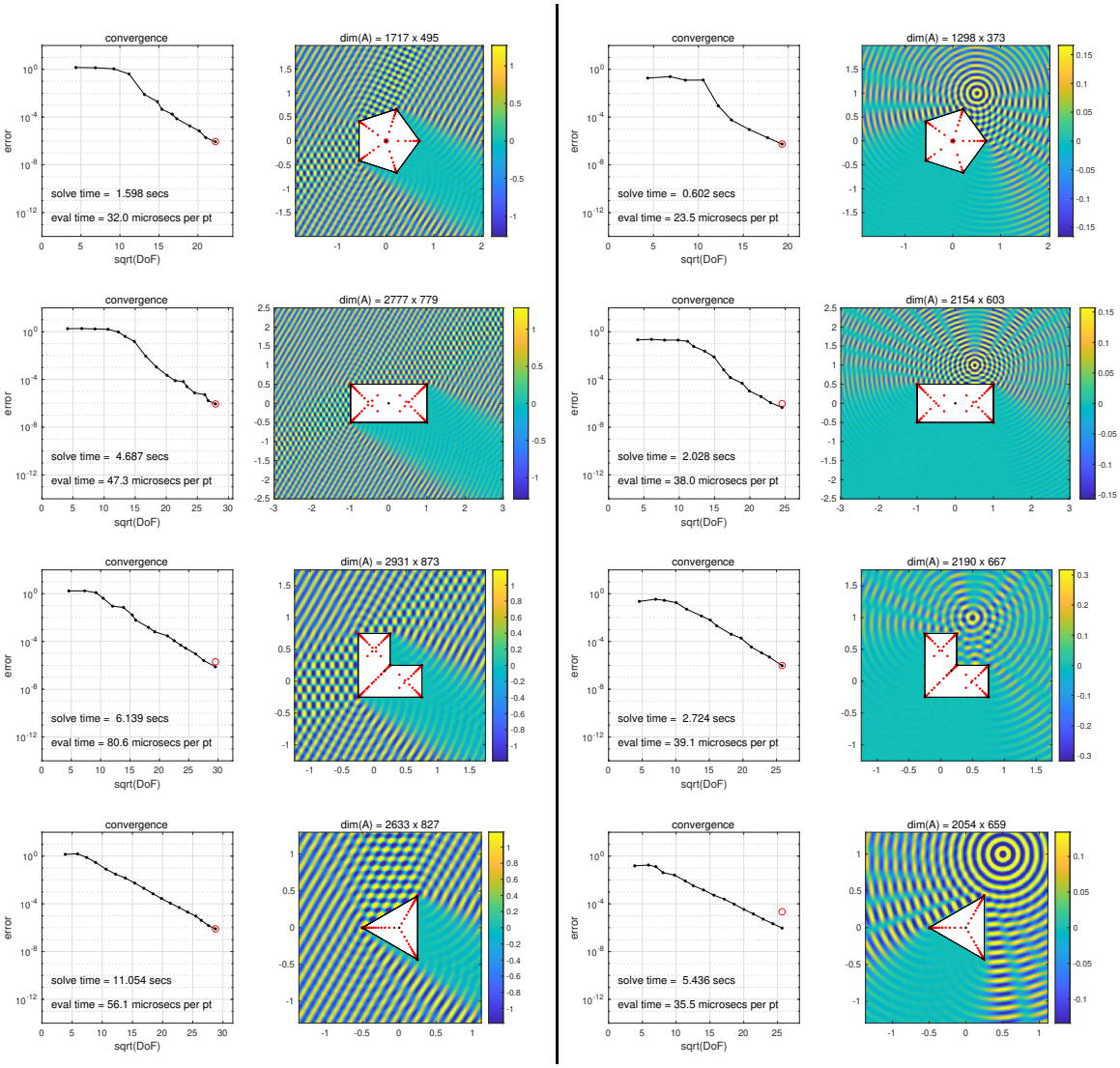

Figure 2.6: Lightning Helmholtz solver for a variety of domains

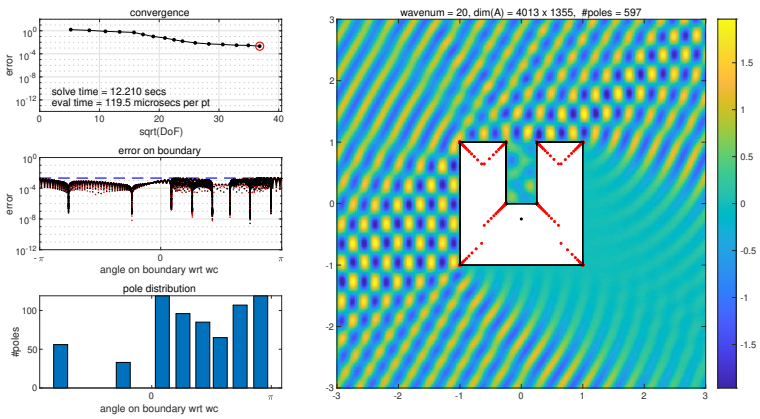

Figure 2.7: A domain that fails to achieve root-exponential convergence

Chapter 3

Discussion

3.1 Lightning Laplace

The Laplace equation is the steady-state heat equation and has many applications (). As discussed in 1.5.3, the lighnting method was first introduced for the context of the Laplace equation. The development of the the Helmholtz lightning solver has been based largely on a similar method for the Laplace equation which we briefly outline in this section. Consider the Laplace equation on a domain bounded by a polygon with boundary sampling as follows

$$\Delta u(z) = 0, \qquad z \in \Omega \tag{3.1}$$

$$u(z) = h(z), \quad z \in \partial\Omega \tag{3.2}$$

To approximate a solutuion to this problem, Gopal and Trefethen [7] assume that the solution takes the following form

$$r(z) = \overbrace{\sum_{j=1}^{N_1} \frac{a_j}{z - z_j}}^{\text{"Newman"}} + \overbrace{\sum_{j=0}^{N_2} b_j (z - z_*)^j}^{\text{"Runge"}} \tag{3.3}$$

$$u(z) \approx \operatorname{Re} r(z) \tag{3.4}$$

here the source terms are dipoles in comparison to the MFS where the source terms are monopoles since the fundamental solution of the laplace equation is given by

$$\phi(x) = -\frac{1}{2\pi} \log(|z|) \tag{3.5}$$

However HELMHOLTZ IS A MUCH HARDER PROBLEM since the solutions are complex and require complex coefficient..

3.2 Numerical Investigation

3.2.1 Adaptive increase of poles

A significant feature of our lightning Helmholtz solver is it's ability to adaptively increase the number of poles at each vertex throughout the iterations. This leads to great speed up as seen in fig. 3.1 which demonstrates an example of the adaptive increase in the number of poles. The top figure represents the *slow* solution, where the increase in the number of poles at each step was the same for each corner. The increase for the bottom figure is governed by (above). Both solutions have roughly the same degrees of freedom. However with an adaptive choice in the number of poles, the solution achieves 8 digits of accuracy in comparison to 5 digits as obtained with the slow version. When the tolerance is fixed to $\epsilon = 10^{-4}$, the adaptive version converges within 1.41 seconds in comparison to 2.50 seconds and for $\epsilon = 10^{-6}$, they obtain 2.66 and 4.05 respectively.

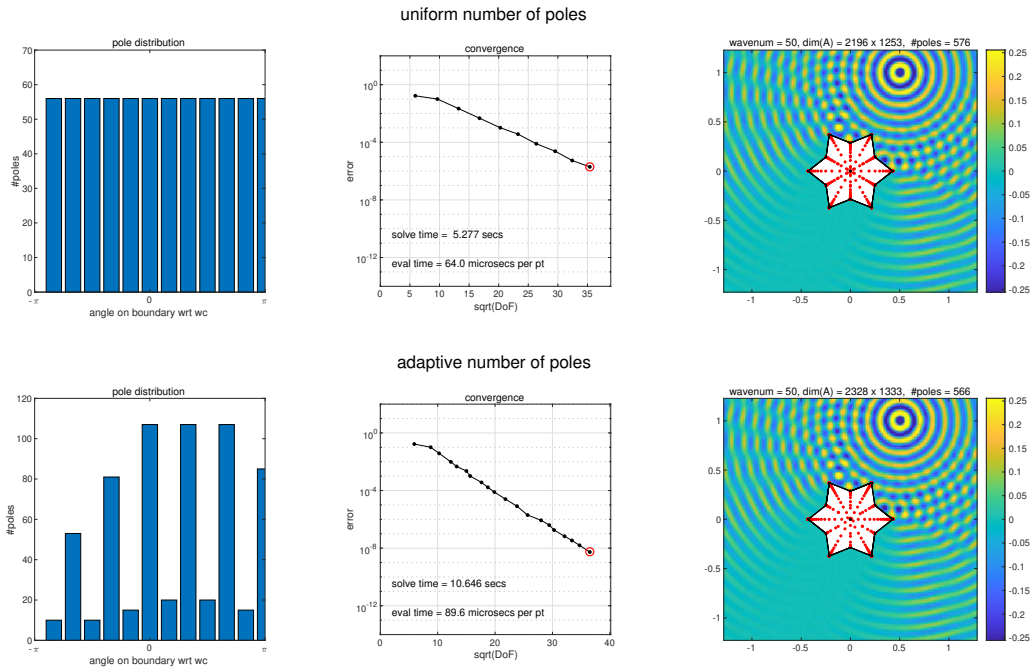

Figure 3.1: Demonstration of the adaptive feature

3.2.2 Comparison of other forms of the solution

In the preprint version of [7], the following rational function was proposed as an approximation to the solution of the Helmholtz equation

$$u(z) \approx \sum_{j=1}^{N_1} a_j H_0(k|w_j|) + b_j H_1(k|w_j|) k \frac{w_j}{|w_j|} + \sum_{j=0}^{N_2} c_j H_j(k|w_*|) \frac{w_*^j}{|w_*^j|} \tag{3.6}$$

We compared approximants of this form to those of the form that we focused on for our implementation eq. (1.40). Figure 3.2 shows the convergence for the solutions of these two forms due to point oscillation incident fields of various wavelengths. The plots in the left-hand column are for eq. (3.6) and those in the right-hand column correspond to eq. (1.40). The solutions in the right-hand column exhibit much cleaner root-exponential behaviour than those in the left-hand column. Furthermore, they successfully converged to the set tolerance of $\epsilon = 10^-8$, wheras eq. (3.6) failed to converge beyond four digits for a wavenumber of $k = 90$.

3.2.3 Boundary points

We expected the number of boundary points to be affected by the wavenumber but nothing came to fruition from exploring this. A more conclusive relationship might appear upon an ideal parameter exploration though we haven't looked too deeply into this.

additionally we place points linearly too. we found that more linear points needed to be placed for curved. if the side is curved, we add more points. this was found to be beneficial when exploring the c shaped region. when solutions failed to converge, we noticed that the error was signficiantly higher on the curved boundaries.

$$M_{2,v}^{(t)} = A\sqrt{|w_v - w_{v+1}|}\max\left(30, N_{1,v}^{(t)}\right) \tag{3.7}$$

where A is the following scaling factor for when a side is curved:

$$A = \begin{cases} \frac{3}{2}\left(1 + \frac{|w_v - w_{v+1}|}{|r_v|}\right) & \text{if side curved} \\ 1 & \text{otherwise} \end{cases} \tag{3.8}$$

We found that the number of linear points should depend on the length of the side, the wavenumber. Extensive numerical exploration found that the following factor provides a robust solution across a wide variety of domains:

furthermore we found that if a side was curved, there should be another factor increase of points. We investigated the problem for a plane wave with wavenumber

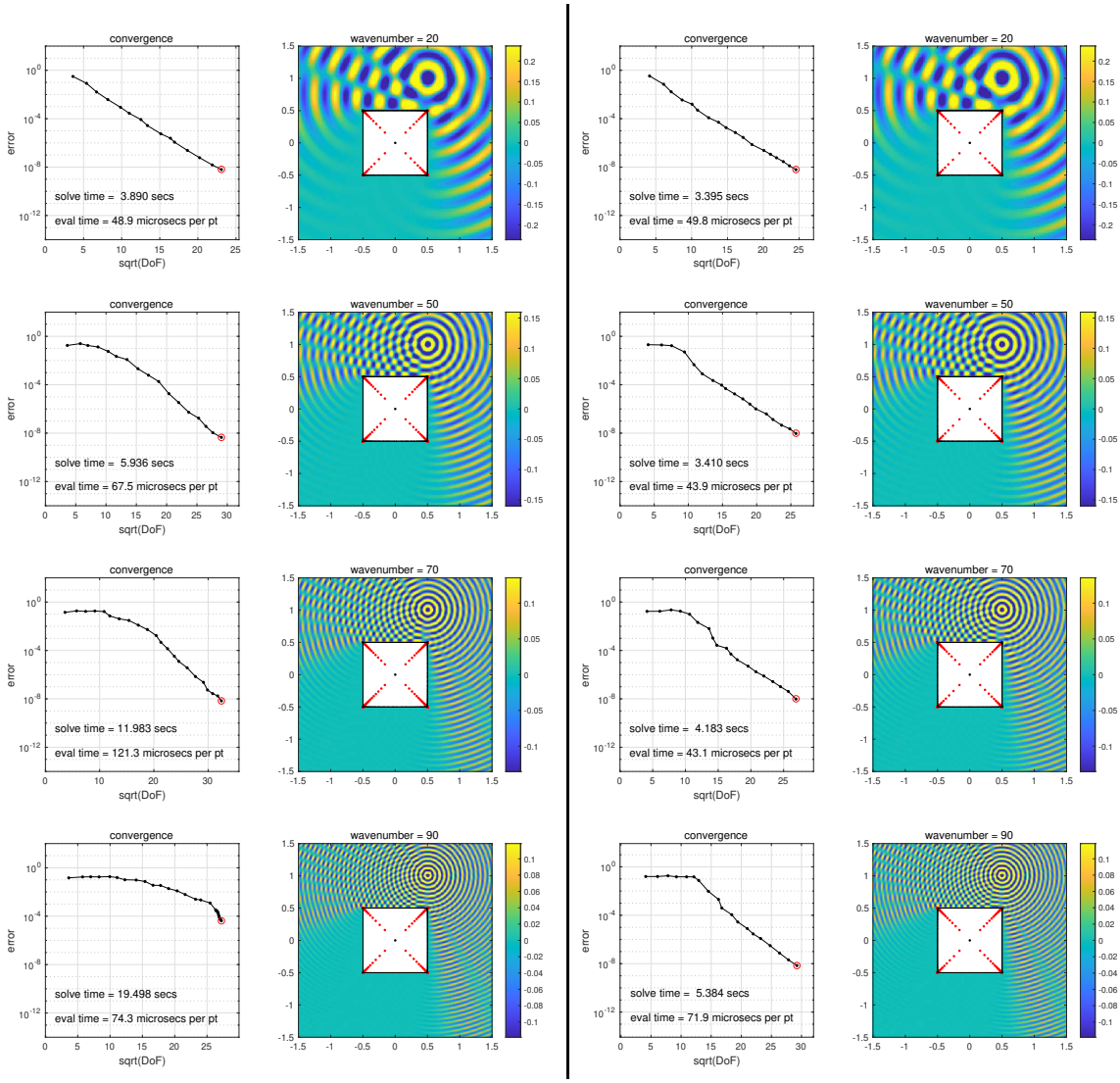

Figure 3.2: Comparison of two different forms of the approximation

30 on a c shape domain with radii 1.05, 1.5 and 3 to isolate the dependency. This is shown in fig. 3.3. Each row of plots corresponds to the various radii. The error on the boundary without the additional points due to the curve is shown in the first column of plots. There is a large spike in the finer mesh error at the curve which appears to have a dependence on the radius of the curve. The error on the boundary for our proposed boundary points in fig. 3.3 is plotted in the second column for the corresponding radius. We can see the spike in the error has been fixed across the range of radii, demonstrating the robustness.

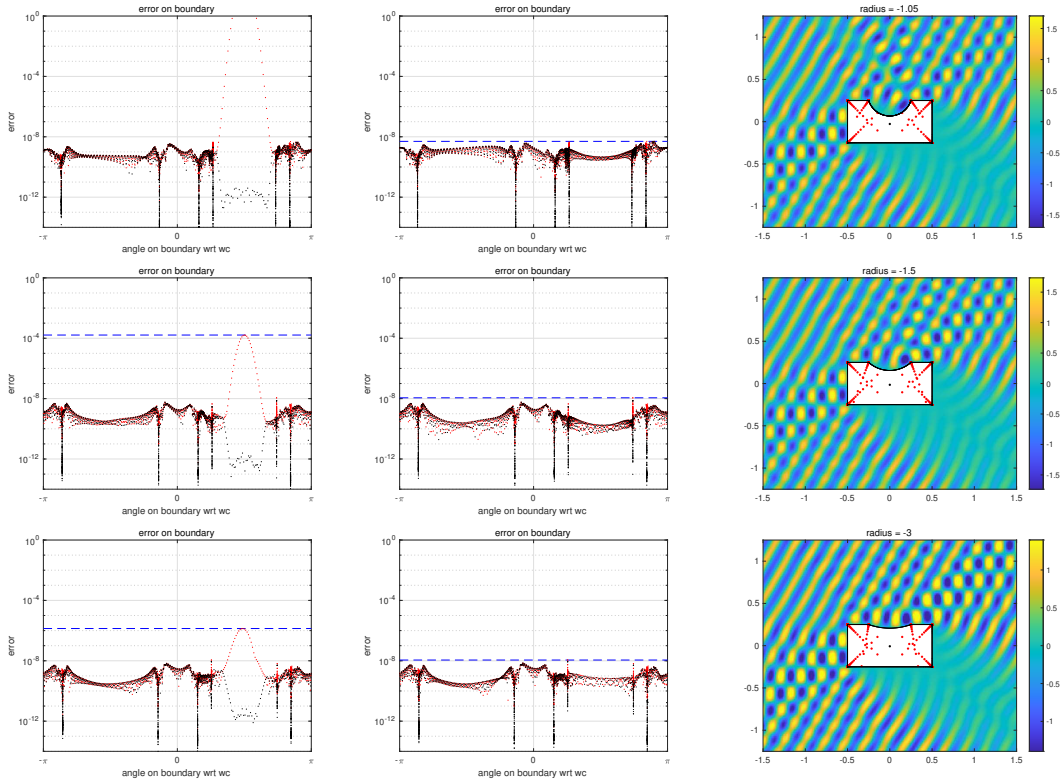

Figure 3.3: Linear boundary points

3.2.4 Runge vs newman points

It is important to investigate the relationship between the number of Runge terms as a function of the number of Newman points in our solution. shows the convergence of the solution for a square with wavenumber $k = 20$. The equally spaced contour lines correspond to clean root-exponential convergence as the number of poles increases.

3.2.5 Condition number

We mentioned in section 1.5.1 that the encountered methods will involve highly ill-conditioned matrices. Figure 3.5 highlights the scale of the conditioning of the systems. The evolution of the condition number of the matrix and the norm of the vector are plotted as functions of the degrees of freedom of the system. The experiment was computed for a planewave with wavenumber $k = 50$ for the square. The condition number is exceptionally large, and indeed much larger than those for the Laplace lightning method which obtains condition numbers on the scale of 10^{16} for the same degrees of freedom [8]. The vector norm remains under control. It is remarkable that solutions are obtained to such accuracy with such large condition numbers.

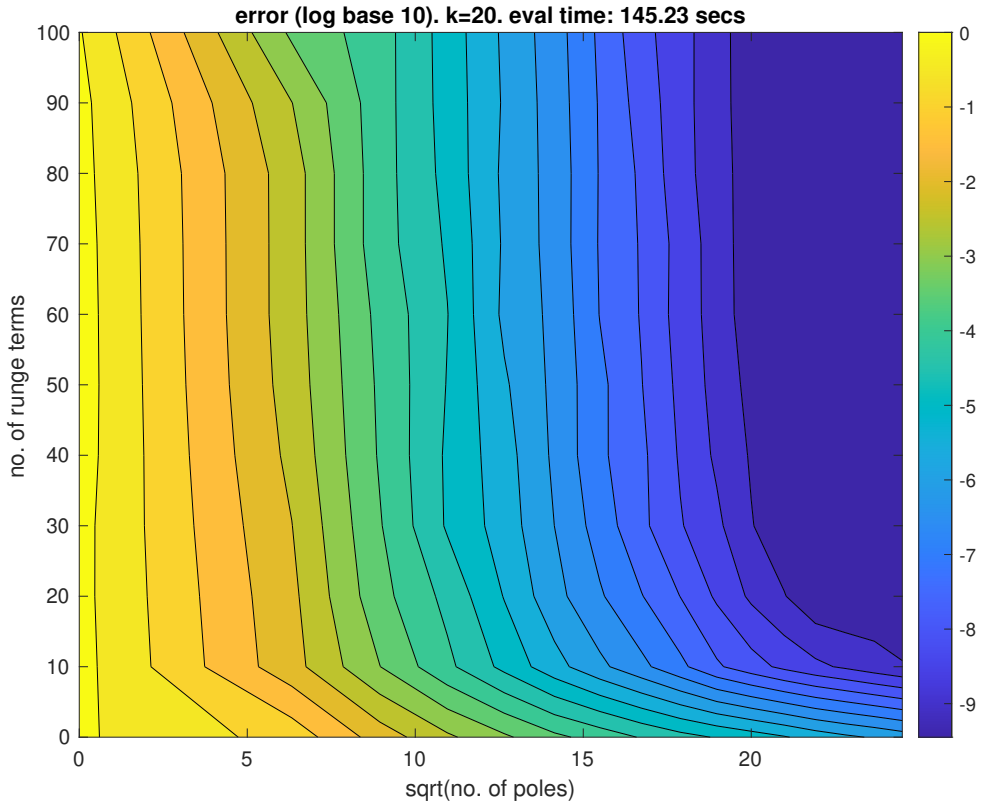

Figure 3.4: Convergence as a function of Newman and Runge terms

3.3 Demonstration of physical phenomena

The high level of accuracy in our solution allows us to demonstrate a variety of physical phenomena associated with waves.

3.3.1 Circular L

We explored the effect of the point source location with respect to a domain of a square with quarter circle of radius 1 cut out of the top left hand corner, in other words, a circular L-shape. The first image shows the point source inside the radius of the cut out circle. We can see here in this image the strong constructive interference in the reflection. The second plot shows the point source at exact centre of the circle. We expect that if the surface was sound-hard that the wave would experience total cancellation within the sector. The third image The fourth image is of a plane wave incoming at an angle of 45° Figure 3.6 all solutions are computed to 8 digits for a wave number of 50.

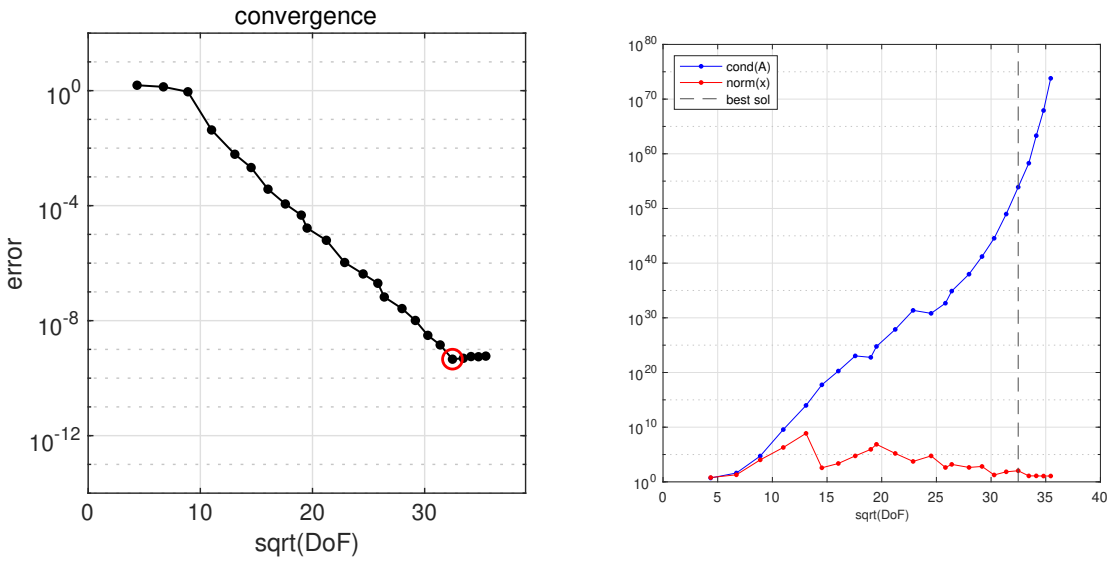

Figure 3.5: Condition number

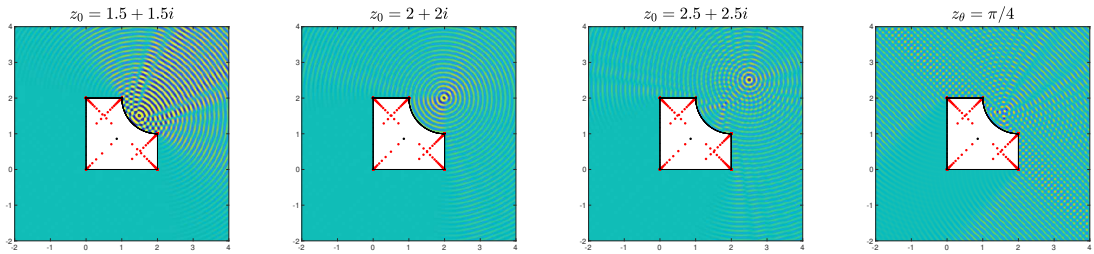

Figure 3.6: Circular L-shape

3.3.2 C-shaped domain

As established, the method appeared to be senstive to curved domains with respect to the number of boundary points.

explore how the solution is affected by the wavenumber for a domain with a C shape fig. 3.7 shows examples for scattering from a a c-shape.

3.3.3 Diffraction

Figure 3.9 shows an example of diffraction. The first image is the normal solution and the second is zoomed into the shadow region with the colourbar scaled to highlight the phenomena. The constructive and destructive interference can be seen quite clearly.

Figure 3.10 provides a fruitful insight into the effect that the wavenumber has on the shadow region beyond the obstacle. We can see that as the wavenumber increases, the diffractive effect diminishes and the boundary between the shadow region and the

wave becomes increasingly well-defined.

3.3.4 Exterior of the domain

We explored the region behind the square and its relation to the number of runge terms in the solution. The plots in fig. 3.12 from left to right have 131, 31, 21, and 3 runge terms respectively. Scattering from a point source with $k = 50$ is plotted.

3.4 Conclusion

We begun development of the lightining Helmholtz Solver, `helmholtz.m` which has shown to be quite successful for a vast array of problems. Many of the problems we encountered were able to be computed in a few seconds of laptop time to 6-digit accuracy. The solver does not resolve well for scattering problems with a wavenumber greater than around $k = 50$. The method is currently restricted to only two dimensions and as such seeking a generalisation to higher dimensions could reveal quite interesting results.

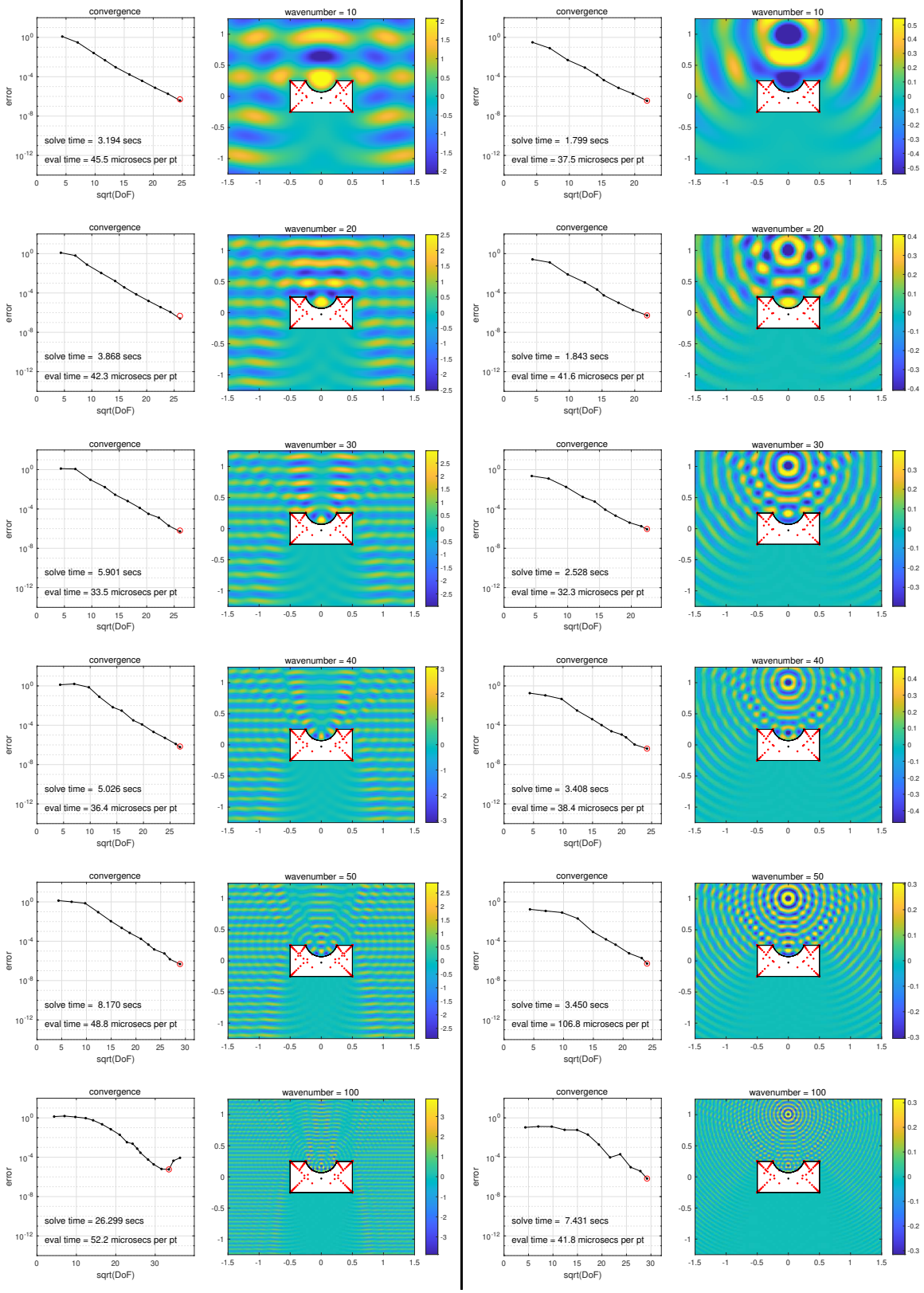

Figure 3.7: C-shape

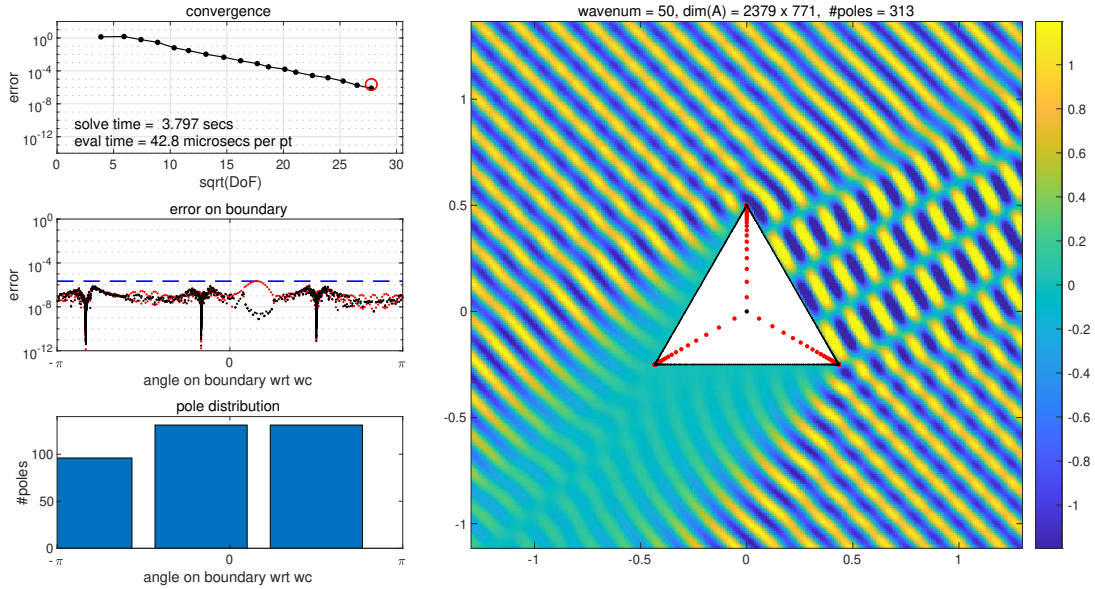

Figure 3.8: triangle; interesting reflection

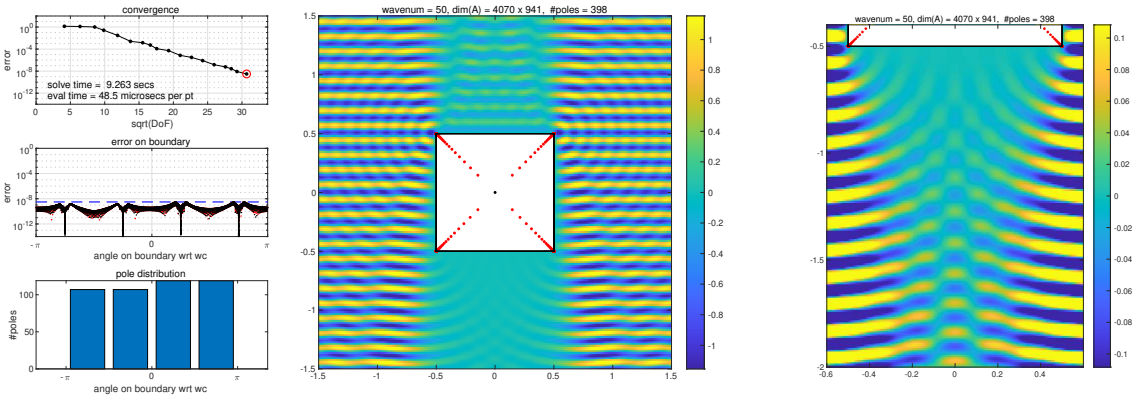

Figure 3.9: Demonstration of diffraction

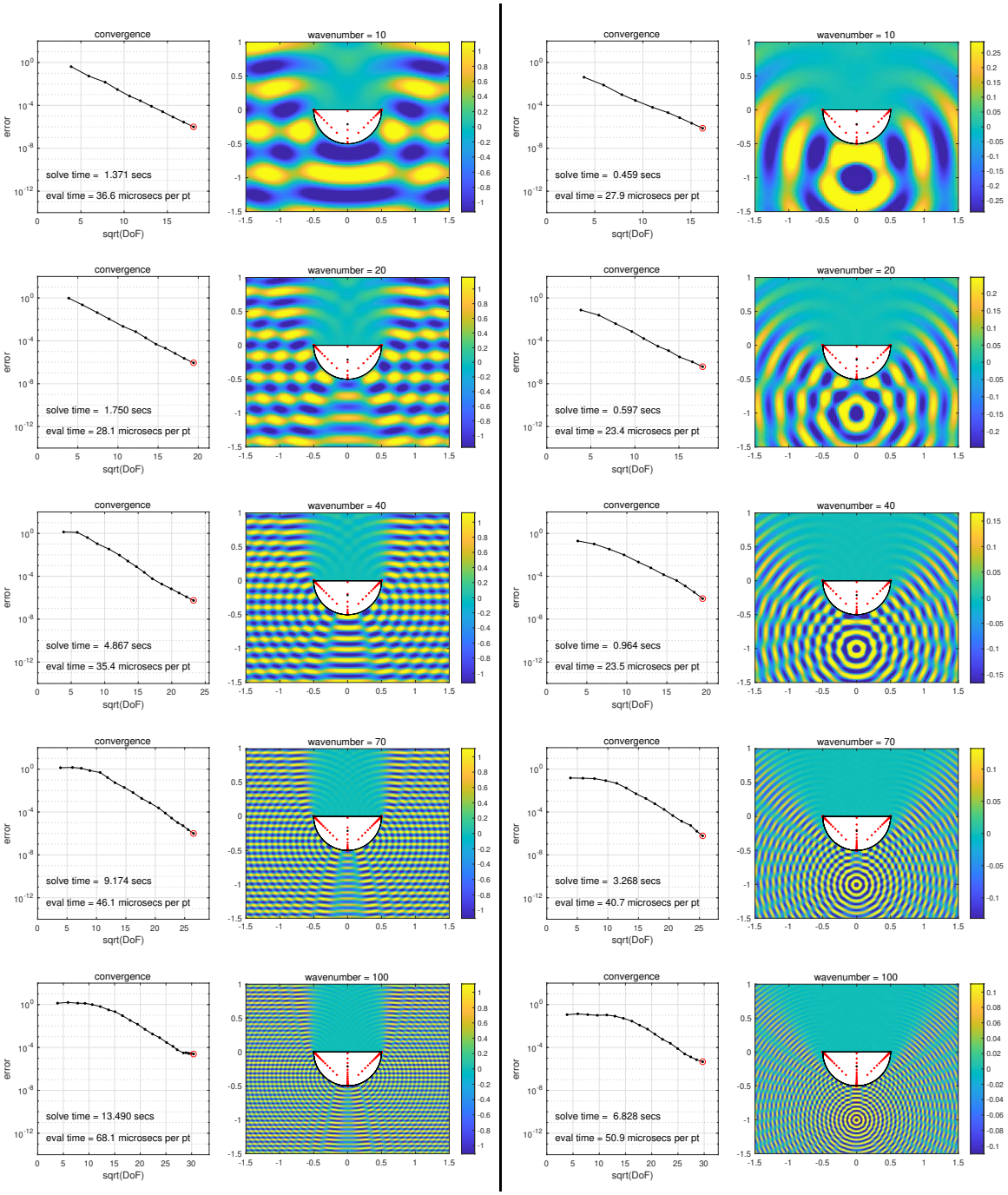

Figure 3.10: Effect of wavenumber on shadow region

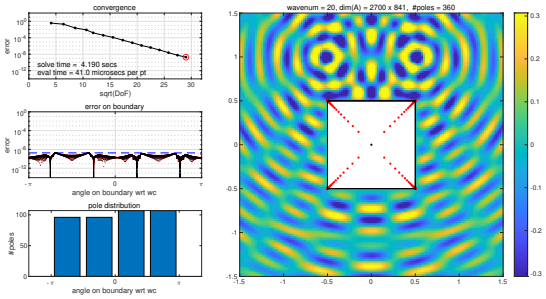

Figure 3.11: Two point sources

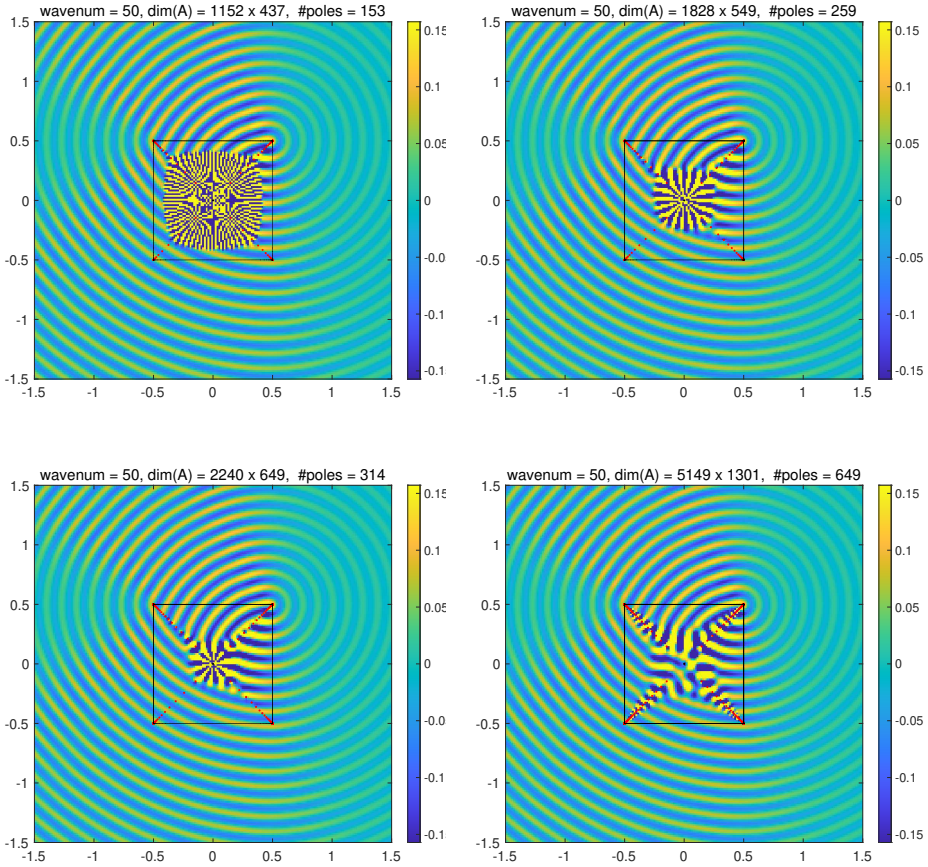

Figure 3.12: Behind the region

Appendix A

helmholtz.m

```
function [u, maxerr, tsolve, nkv, Z, Zplot, pol, A] = ...
    helmholtz(wavenum, P, varargin)
%HELMHOLTZ  Lightning Helmholtz solver.
%         U = HELMHOLTZ(wavenum,P,G) solves the Helmholtz ...
    equation with
%         Dirichlet boundary data on the simply-connected region ...
    Omega
%         bounded by P, which may be a polygon or circular polygon,
%         v1, University of Oxford, April 2022.
%
%  Inputs:
%      P = vector of corners as complex numbers z = x+iy in ...
    counterclockwise
%              order to specify a polygon
%           or cell array of corners v and pairs [v r] to specify ...
    a circular
%              polygon: r = radius of curvature of arc from this ...
    v to the next
%           or one of the following specified strings
%              'sqr'[square], 'rec'[tangle], 'snow'[flake], ...
    pent[agaon],
%               'hex'[agon], 'L', 'circleL', or 'C'
%           or integer ≥ 3, the number of corners of a random polygon
%           or integer ≤ -3, -1 x no. of corners of a random ...
    circular polygon]
%
%      g = function handle for Dirichlet boundary data that ...
    satisfies helm(g)
%           or cell array of function handles for sides P1-P2, ...
    P2-P3,...
%           (default @(z) exp(-1i*real(wavenum*exp(-1i*z0ang)*z))) ...
    for wavenumber>0
%                  @(z) @(z) besselh(0,-wavenum*abs(z-(z0_pt))) ...
    for wavenumber<0)
%
%  Further optional inputs:
```

```
25 %    'tol', tol = tolerance for maximal absolute error (default 1e-6)
26 %    'z0' to specify the point/angle of incidence for defulat ...
       incident source
27 %    'noplots' to suppress plotting
28 %    'noplot3d' to suppress 3D surface plotting
29 %    'steps' for step-by-step plots of errors on boundary and poles
30 %    'scat' to plot only the scattered field
31 %    'slow' to turn off adaptive mode for cleaner root-exp ...
       convergence curves
32 %    'fs' to set font size for plots
33 %
34 %  Outputs for [U,MAXERR,TSOLVE,NKV,Z,ZPLOT,POL,A] = helmholtz(k,P,G)
35 %       u = function handle for solution u(z) of helm(u) = 0, u = ...
       g on boundary
36 %  maxerr = estimated upper bound on maximal error,
37 %              even near singularities at corners
38 %       tsolve = runtime for calculcating solution
39 %       nkv = vector containing number of poles at each corner in ...
       solution
40 %       pol = set of poles
41 %       Z = sample points on boundary
42 %       Zplot = set of points for plotting boundary
43 %       A = final rectangular matrix used for least-squares problem
44 %
45 % Examples:
46 %
47 %   helmholtz(50,'sqr');                                  % plane ...
       with square
48 %   helmholtz(-50,'sqr');                                 % point ...
       with square
49 %   helmholtz(-20,'pent','tol',1e-10);                    % pentagon
50 %   helmholtz(-20,'circleL','z0',2+3i);                   % circular ...
       L-shape
51 %   helmholtz(20,'bullet','z0',1);                        % bullet
52 %   helmholtz(-30,'snow','steps')                         % snowflake
53 %   helmholtz(50,[1/2*exp(2i*pi*([1:3])/3)],'z0',1i)      % triangle
54 %
55 % two point sources:
56 %   wavenum = -30; z0_pt = .5+1i;
57 %   g = @(z) besselh(0,-wavenum*abs(z-(z0_pt))) + ...
58 %       besselh(0,-wavenum*abs(z-(-z0_pt')));
59 %   helmholtz(wavenum,'sqr',g,'noplot3d');
60
61 %% Set up the problem
62 [g, P, w, ww, pt, dw, tol, steps, scat, ...      % parse inputs
63    plots, plot3d, fs, slow] = ...
64    parseinputs(wavenum, P, varargin{:});
65 wavenum = abs(wavenum);
66 Zplot = ww;
67 nw = length(w);                                  % number of corners
68 x = real(ww); y = imag(ww);
69 wr = sort(x); wr = wr([1 end]);
70 wi = sort(y); wi = wi([1 end]);
```

```
scl = max([diff(wr),diff(wi)]);                    % characteristic ...
    length scale
q = .5; if slow == 1, q = 0; end                   % sets which ...
    corners get more poles
inpolygonc = @(z,w) inpolygon(real(z), ...         % complex ...
    variant of "inpolygon"
             imag(z),real(w),imag(w));

A = x(1:end).*y([2:end,1])-x([2:end,1]).*y(1:end);
As = sum(A)/2;
x_bar = (sum((x([2:end,1])+x(1:end)).*A)*1/6)/As;
y_bar = (sum((y([2:end,1])+y(1:end)).*A)*1/6)/As;
wc = x_bar+1i*y_bar;                                % centroid of ...
    polygon
zs = wc;
while ¬inpolygonc(zs,ww)
    zs = input('centroid not in polygon, please manually enter ...
        runge point: ');
end
for k = 1:nw
    forward = pt{k}(.01*dw(k)) - w(k);             % small step ...
        toward next corner
    j = mod(k-2,nw)+1;
    backward = pt{j}(.99*dw(j)) - w(k);            % small step ...
        toward last corner
    tmp = -1i*backward*sqrt(-forward/backward);
    inward(k) = tmp/abs(tmp);                      % inward ...
        direction from corner
end
warn = warning('off','MATLAB:rankDeficientMatrix');  % matrices ...
    are ill-conditioned

%% Set up for plots
if plots
    LW = 'linewidth'; MS = 'markersize'; FS = 'fontsize';
    PO = 'position'; FW = 'fontweight'; NO = 'normal';
    npts = max(120,min(wavenum*8,250));
    sx = linspace(wr(1)-scl,wr(2)+scl,npts); sy = ...
        linspace(wi(1)-scl,wi(2)+scl,npts);
    [xx,yy] = meshgrid(sx,sy); zz = xx + 1i*yy;
    ax = [wr(1:2); wi(1:2)] + scl*[-1 1 -1 1]';
    axwide = [wr(1:2); wi(1:2)] + 1.1*scl*[-1 1 -1 1]';
    axnarrow = [wr(1:2); wi(1:2)] + .5*scl*[-1 1 -1 1]';
end
if steps
    figure, subplot(1,2,1), shg, plot(w([1:end 1]),'k',LW,1)
    grid on, axis equal, axis(axnarrow)
end

%% Main loop: increase number of poles until convergence ...
    ============================
Nvec = []; errvec = []; tic
errk = ones(nw,1);                                 % max error near ...
    each corner
```

```
nkv = ones(nw,1);                                   % no. of poles ...
    at each corner
maxstepno = 30; degfmax = 35^2; err0 = Inf;

for stepno = 1:maxstepno
   % Fix poles and sample pts on bndry.  Side k means side from ...
      corner k to k+1.
   Z = [];           % col vector of sample points on boundary
   G = [];           % col vector of boundary values at these points
   pol = [];         % row vector of poles of the rational ...
      approximation
   J = [];           % row vector of indices of which corner each ...
      pole belongs to
   d = [];           % row vector of distances from poles to ...
      their corners
   tt = cell(nw,1);  % cell array of distances of sample points ...
      along each side
   for k = 1:nw
      nk = nkv(k);                                  % no. of ...
         poles at this corner
%       normal clustering
%            sk = sqrt(1:nk) - sqrt(nk);
%            dk1 = exp(4*sk); dk = scl*dk1;             % ...
    stronger clustering near corner
%            dk = dk(dk>1e-15*scl);                     % ...
    remove poles too close to corner

      % stronger clustering (greased lightning)
      sk = sqrt(1:nk) - sqrt(nk);
      dk1 = exp(4*sk);
      dk = .5*scl*dk1/max(dk1);                     % ...
         stronger clustering near corner
      dk = dk(dk>1e-15*scl);                        % remove ...
         poles too close to corner
      polk = w(k) + inward(k)*dk;                   % poles ...
         near this corner
      ii = find(¬inpolygonc(polk(dk>1e-12*scl),ww),1);  % work ...
         around inaccuracy
      if length(ii)>0                               % don't ...
         allow poles in Omega
         dk = dk(1:ii-2); polk = polk(1:ii-2);
      end
      pol = [pol polk]; d = [d dk];
      dvec = [(1/3)*dk (2/3)*dk dk];                % finer ...
         pts for bndry sampling
      nsidepts = sqrt(dw(k))*max(30,10*sqrt(nk));   % numbr ...
         additional pts along side
      if length(P{k}) == 2                          % add ...
         more bndry pts if side curved
         factor = 1+dw(k)/abs(P{k}(2));
         nsidepts = 1.5*nsidepts*factor;
      end
      tt{k} = [tt{k} dvec(dvec<dw(k)) ...           % add ...
         clustered pts near corner
```

```
149           linspace(0,dw(k),nsidepts)];
150        j = mod(k-2,nw)+1;                                  % index ...
              of last corner
151        tt{j} = [tt{j} dw(j)-dvec(dvec<dw(j)) ...           % ...
              likewise in other direction
152           dw(j)-linspace(0,dw(j),nsidepts)];
153     end
154     for k = 1:nw
155        tt{k} = sort(unique(tt{k}(:)));
156        tk = tt{k}; pk = pt{k};                             % ...
              abbrevations
157        Z = [Z; pk(tk)];                                    % sample ...
              pts on side k
158        G = [G; g{k}(pk(tk))];                              % ...
              boundary data at these pts
159     end
160
161     % Solve the Helmholtz problem
162     n = 4*stepno;                                          % (degree ...
           of polynomial term) i.e number of runge terms
163     Np = length(pol);                                      % number ...
           of newman terms
164
165     M = size(Z,1);                                         % number ...
           of boundary samples
166     Gn = G;
167     A = cauchy(Z,zs,pol,n,wavenum);
168     J = [zeros(1,2*n+1) J J];                              % corner ...
           for each col
169     N = size(A,2);                                         % no. of ...
           cols = 2n+1+2Np CHECK WHY THIS MAY NOT BE WORKING
170     Kj = zeros(M,1);
171     for j = 1:M
172        dd = abs(Z(j)-w);
173        Kj(j) = find(dd==min(dd),1);                        % nearest ...
              corner to Zj
174     end
175     D = 1./vecnorm(A); nD = length(D); D = spdiags(D',0,nD,nD);
176     c = D*((A*D)\Gn);
177     ftemp = @(zz) cauchy(zz,zs,pol,n,wavenum)*c;
178     u = @(z) reshape(ftemp(z(:)),size(z));                 % to u ...
           and f both allowed
179     for k = 1:nw
180        Kk = find(Kj==k);
181        errk(k) = norm(A(Kk,:)*c-Gn(Kk),inf);               % error ...
              near corner k
182     end
183     err = norm(A*c-Gn,inf);                                % global ...
           error
184     polmax = 120;
185     for k = 1:nw
186        if (errk(k) > q*err) & (nkv(k) < polmax)
187           nkv(k) = nkv(k)+ceil(1+sqrt(nkv(k)));            % ...
                 increase no. poles
```

```
188         else
189            nkv(k) = max(nkv(k),ceil(stepno/2));
190            %nkv(k) = min(polmax,nkv(k)+1);
191         end
192      end
193      if steps                                                % plot ...
            error on bndry
194         subplot(1,2,1), plot(ww,'k',LW,1), grid on
195         title(sprintf('step %d',stepno))
196         axis equal, axis(axnarrow), hold on
197         plot(pol,'.r',MS,7), hold off
198         subplot(1,2,2), semilogy([-pi pi],err*[1 1],'--b',LW,1), ...
               axis square
199         hold on, axis([-pi pi 1e-16 100]), grid on
200         semilogy(angle(Z-wc),abs(u(Z)-G),'.k',MS,4)
201         semilogy(angle(pol-wc),d,'.r',MS,7), hold off
202         set(gca,'ytick',10.^(-16:4:0))
203         set(gca,'xtick',pi*(-1:1),'xticklabel',{'-\pi','0','\pi'})
204         set(gca,FS,fs-1), xlabel('angle on boundary wrt wc',FS,fs)
205         title('bndry err (black) & poles (red)',FS,fs)
206         disp('Press <enter> for next plot'); pause
207      end
208      errvec = [errvec err]; Nvec = [Nvec; N];
209      if err < tol, break, end                                % ...
            convergence success
210      if err < err0                                           % save ...
            the best so far
211         u0 = u; Z0 = Z; G0 = G; A0 = A; M0 = M;
212         N0 = N; err0 = err; pol0 = pol;
213      end
214      if (N > degfmax) || (stepno == maxstepno) || (Np ≥ polmax*nw)  ...
            % failure
215         u = u0; Z = Z0; G = G0; A = A0; M = M0;
216         N = N0; err = err0; pol = pol0;
217         warning('HELMHOLTZ failure.  Loosen tolerance or add corners?')
218         break
219      end
220   end
221   warning(warn.state,'MATLAB:rankDeficientMatrix')          % back to ...
         original state
222   tsolve = toc;  % ========== end of main loop ...
         ========================================
223
224   %% Finer mesh for a posteriori error check
225   Z2 = []; G2 = [];
226   for k = 1:nw
227      newtt = mean([tt{k}(1:end-1) tt{k}(2:end)],2);
228      newpts = pt{k}(newtt);
229      Z2 = [Z2; newpts];
230      G2 = [G2; g{k}(newpts)];
231   end
232   err2 = norm((G2-u(Z2)),inf);
233   maxerr = max(err,err2);                                   % ...
         estimated max error
```

```
234
235 %% Convergence curve plot
236 if plots
237    fig = figure; fig.Position = [500 100 700 400]; shg
238    axes(PO,[.07 .7 .3 .2])
239    semilogy(sqrt(Nvec),errvec,'.-k',LW,0.7,MS,10), grid on, hold on
240    semilogy(sqrt(N),maxerr,'or',MS,7,LW,1), hold off
241    errmin = .01*tol; axis([0 1.1*max(sqrt(Nvec)) 1e-14 100])
242
243    set(gca,FS,fs-1), title('convergence',FS,fs,FW,NO)
244    xlabel('sqrt(DoF)',FS,fs), ylabel('error',FS,fs)
245    set(gca,'ytick',10.^(-16:4:0))
246    ax2 = axis; x1 = ax2(1) + .05*diff(ax2(1:2));
247    s = sprintf('solve time = %6.3f secs',tsolve);
248
249    if ¬steps, text(x1,4e-11,s,FS,fs), end
250    z = randn(1000,1)+1i*randn(1000,1); z = z/10;
251    tic, u(z); teval = 1e3*toc;
252    s = sprintf('eval time = %4.1f microsecs per pt',teval);
253    text(x1,4e-13,s,FS,fs)
254 end
255
256 %% Error plot along boundary
257 if plots
258    axes(PO,[.07 .4 .3 .2])
259    semilogy([-pi pi],maxerr*[1 1],'--b',LW,1), hold on
260    semilogy(angle(Z2-wc),abs(u(Z2)-G2),'.r',MS,4)
261    axis([-pi pi .0001*errmin 1]), grid on
262    semilogy(angle(Z-wc),abs(u(Z)-G),'.k',MS,4), hold off
263    set(gca,'ytick',10.^(-16:4:0))
264    set(gca,'xtick',pi*(-1:1),'xticklabel',{'-\pi','0','\pi'})
265    set(gca,FS,fs-1), xlabel('angle on boundary wrt wc',FS,fs)
266    ylabel('error',FS,fs)
267    title('error on boundary',FS,fs,FW,NO)
268    %    axis square
269 end
270
271 %% Displaying number of poles per corner
272 if plots
273    axes(PO,[.07 .1 .3 .2])
274    bar(angle(w-wc),nkv)
275    set(gca,'xtick',pi*(-1:1),'xticklabel',{'-\pi','0','\pi'})
276    set(gca,FS,fs-1)
277    xlabel('angle on boundary wrt wc',FS,fs)
278    ylabel('#poles',FS,fs)
279    xlim([-pi,pi])
280    ylim([0 round(max(nkv),-1)+10])
281    title('pole distribution',FS,fs,FW,NO)
282 end
283
284 %% Plot solution of total field
285 if plots
286    tic,
287    uu = u(zz(:));
```

```
288     if scat == 0, f = @(x) -g{1}(x);
289     else f = @(x) 0; end
290     ind = inpolygonc(zz,ww);
291     mxcol = .5*max(real(uu(¬ind))); mncol = min(real(uu(¬ind)));
292     uu = reshape(uu,size(zz));
293     axes(PO,[.40 .1 .6 .8])
294     imagesc(sx,sy,real(exp(-1i*0)*(f(zz) + uu)),[mncol mxcol]), ...
            hold on
295     fill(real(ww),imag(ww),[1 1 1]), hold on, colorbar, set(gca,FS,8)
296     set(gca,'YDir','normal')
297     mxabscol = max(abs([mxcol,mncol]));
298     caxis([-mxabscol,mxabscol])
299     axis equal, axis(ax)
300
301     plot(pol,'.r',MS,8), plot(real(zs),imag(zs),'.k',MS,8), ...
            plot(Z,'.k',MS,2), hold off
302     ss = ['wavenum = ' int2str(wavenum) ', dim(A) = ' int2str(M) ' ...
            x ' int2str(N) ', ', ...
303        ' #poles = ' int2str(length(pol))];
304     title(ss,FS,fs,FW,NO)
305 end
306
307 %% 3D Plotting
308 if plots && plot3d
309     zz3d = zz; zz3d(ind) = nan;
310     sz = real(exp(1i*0)*(f(zz3d)+u(zz3d)));
311     figure, axis equal, surfc(sx,sy,sz)
312 end
313
314 end
315 %% Functions
316 %% Treat the inputs
317 function [g, P, w, ww, pt, dw, tol, steps, scat, ...
318     plots, plot3d, fs, slow] = parseinputs(wavenum, P, varargin)
319 %% Defaults
320 tol = 1e-6; steps = 0; scat = 0; plots = 1;
321 plot3d = 1; fs = 9; slow = 0;
322
323 z0_pt = .5+1i; z0ang = 5*pi/6;
324
325 %% First treat the domain, defined by P
326 randomcirc = 0;
327 % this is if P is not a cell i.e P!={[..]}
328 if ¬iscell(P)
329     if isnumeric(P)
330         if length(P) > 1, w = P;                    % vertices have ...
                been specified
331         else
332             if P < 0
333                 randomcirc = 1; P = -P;             % random ...
                        circular arcs
334             end
335             w = exp(2i*pi*(1:P)/P).*(.1+rand(1,P)); % random vertices
336         end
```

```
337     else
338        cellscale = @(P,s) cellfun(@times, P, ...
              num2cell(s*ones(1,length(P))), 'UniformOutput',false);
339        if strcmp(P,'sqr'), w = .5*[-1-1i 1-1i 1+1i -1+1i];
340        elseif strcmp(P,'rec'), w = .5*[-2-1i 2-1i 2+1i -2+1i];
341        elseif strcmp(P,'snow'), P = exp(2i*pi*(1:12)/12);
342        elseif strcmp(P,'pent'), w = .7*exp(pi*2i*(1:5)/5);
343        elseif strcmp(P,'hex'), w = .7*exp(pi*2i*(1:6)/6);
344        elseif strcmp(P,'kite'), w = .25*[0 2+4i 5i -2+4i];
345        elseif strcmp(P,'L'), w = .5*([2 2+1i 1+1i 1+2i 2i ...
              0]-.5*(1+1i));
346        elseif strcmp(P,'circleL'), w = {2 [2+1i -1] 1+2i 2i 0};
347        elseif strcmp(P,'C'), P = {-2-1i 2-1i 2+1i [1+1i -1.1] ...
              -1+1i -2+1i}; P = cellscale(P,.25);
348
349           w = P.*(1+.2*(-1).^(1:12)); w = w/2.8;
350        elseif strcmp(P, 'bullet'), P = {.5*[1-.5i .6] .5*(1+.5i) ...
              .5*(-1+.5i) .5*(-1-.5i)};
351        end
352     end
353     if ¬iscell(P), P = num2cell(w); end            % convert to ...
           cell array
354     if randomcirc
355        for k = 1:length(P)
356           r = .6/rand;
357           P{k} = [P{k} r*(-1)^double(randn>0)];
358        end
359     end
360  end
361
362  nw = length(P);
363  for k = 1:nw
364     w(k) = P{k}(1);
365  end
366  w = w(:);
367  ww = [];                                          % bndry pts ...
        for plotting
368  for k = 1:nw
369     kn = mod(k,nw)+1;                              % index of ...
           next corner
370     ww = [ww; w(k)];
371     if isnumeric(P{k})
372        if length(P{k}) == 1                        %     ...
              straight arc
373           dw(k) = abs(w(kn)-w(k));                 % distance to ...
                 next corner
374           pt{k} = @(t) w(k) + t*(w(kn)-w(k))/dw(k);  % ...
                 parametrization of arc
375        else                                        %     ...
              circular arc
376           r = P{k}(2);                             % radius of arc
377           a = w(k); b = w(kn); ab = abs(b-a);      % endpoints ...
                 of arc
```

```
378           theta = real(asin(ab/(2*r)));              % half-angle ...
                 of arc
379           c = a + r*exp(1i*(pi/2-theta))*(b-a)/ab;   % center of arc
380           dw(k) = 2*theta*r;                         % arc length ...
                 of arc
381           pt{k} = @(t) c - ...
382              r*exp(1i*(pi/2+t/r-theta))*(b-a)/ab;   % ...
                    parametrization of arc
383           ww = [ww; pt{k}(linspace(0,dw(k),50)')];
384        end
385     else
386        error('HELMHOLTZ:parseinputs','general boundary arcs not ...
              yet implemented')
387     end
388  end
389  ww = [ww; w(1)];
390  Zplot = ww;
391
392  %% Next treat the boundary conditions
393  for k = 1:nw
394     % defaults
395     if wavenum>0, g{k} = @(z) exp(-1i*real(wavenum*exp(-1i*z0ang)*z));
396     else g{k} = @(z) besselh(0,-wavenum*abs(z-(z0_pt))); end
397  end
398
399  j = 1;
400  while j < nargin-1
401     j = j+1;
402     v = varargin{j-1};
403
404     if ¬ischar(v)                  % This block specifies Dirichlet ...
          bndry data g.
405        if isa(v,'cell')            % if cell array, nothing to change
406           g = v;
407        elseif isa(v,'double')      % if vector, convert to cell ...
             array of fun. handles
408           for k = 1:nw
409              g{k} = @(z) v(k) + 0*z;
410           end
411        elseif isa(v,'function_handle')  % if fun. handle, convert ...
             to cell array
412           for k = 1:nw
413              g{k} = @(z) v(z);
414           end
415        else
416           error('HELMHOLTZ:parseinputs','boundary data g not in ...
                 correct form')
417        end
418
419     elseif strcmp(v,'z0'), j = j+1;  % user-specified z0 parameter ...
          for default bndry funs
420        z0param = varargin{j-1};
421        for k = 1:nw
422           if wavenum>0
```

```
423             if ¬isreal(z0param), z0ang = angle(z0param); else ...
                    z0ang = z0param; end
424             g{k} = @(z) exp(-1i*real(wavenum*exp(-1i*z0ang)*z));
425          else z0_pt = z0param; g{k} = @(z) ...
                besselh(0,-wavenum*abs(z-(z0_pt))); end
426       end
427
428    elseif strcmp(v,'tol'), j = j+1; tol = varargin{j-1};
429    elseif strcmp(v,'steps'), steps = 1; plots = 1;
430    elseif strcmp(v,'scat'), scat = 1;
431    elseif strcmp(v,'noplots'), plots = 0;
432    elseif strcmp(v,'noplot3d'), plot3d = 0;
433    elseif strcmp(v,'slow'), slow = 1;
434    elseif strcmp(v,'fs'), j = j+1; fs = varargin{j-1};
435    else error('HELMHOLTZ:parseinputs','Unrecognized string input')
436    end
437 end
438 wavenum = abs(wavenum);
439
440 end   % end of parseinputs
441
442 %% Cauchy matrix
443 function C = cauchy(Z,zs,pol,n,wavenum)  % Make Cauchy matrix. ...
       The math is here!
444 Np = length(pol);
445 M = length(Z);
446 C = zeros(M,2*(Np+n)+1);
447 if Np>0                                                          ...
       % Newman parts
448    D = Z-pol; absD = abs(D);
449    vals = [real(D./absD) imag(D./absD)];
450    bessvals = besselh(1,wavenum*absD);
451    C(:,1:2*Np) = repmat(bessvals,[1,2]).*vals;
452 end
453
454 s = Z-zs; abss = abs(s);
455 for j = 1:n ...
                                                                 % ...
       Runge parts
456    bessval = besselh(j,wavenum*abss);
457    C(:,2*Np+j) = bessval.*real((s./abss).^j);
458    C(:,2*Np+n+j) = bessval.*imag((s./abss).^j);
459 end
460 C(:,2*(Np+n)+1) = besselh(0,wavenum*abss);
461
462 end
463
464 % Note: this script has been deliberately made to look similar to ...
       laplace.m
```

Bibliography

[1] Pedro RS Antunes. A well-conditioned method of fundamental solutions for laplace equation. *Numerical Algorithms*, pages 1–25, 2022.

[2] Alex H. Barnett and Timo Betcke. Stability and convergence of the method of fundamental solutions for helmholtz problems on analytic domains. *J. Comput. Phys.*, 227:7003–7026, 2008.

[3] Alex H. Barnett and Timo Betcke. Stability and convergence of the method of fundamental solutions for helmholtz problems on analytic domains. *J. Comput. Phys.*, 227:7003–7026, 2008.

[4] Pablo D Brubeck and Lloyd N. Trefethen. Lightning stokes solver. *ArXiv*, abs/2107.01572, 2021.

[5] *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.1.5 of 2022-03-15. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.

[6] Graeme Fairweather and Andreas Karageorghis. The method of fundamental solutions for elliptic boundary value problems. *Advances in Computational Mathematics*, 9:69–95, 1998.

[7] Abinand Gopal and Lloyd N. Trefethen. New laplace and helmholtz solvers. *Proceedings of the National Academy of Sciences of the United States of America*, 116:10223 – 10225, 2019.

[8] Abinand Gopal and Lloyd N Trefethen. Solving laplace problems with corner singularities via rational functions. *SIAM Journal on Numerical Analysis*, 57(5):2074–2094, 2019.

[9] Timothy Gowers, June Barrow-Green, and Imre Leader. *The Princeton companion to mathematics*. Princeton University Press, 2008.

[10] Peter Henrici. Applied and computational complex analysis. 1974.

[11] Dave Hewett. Multipole expansions for the helmholtz equation. Unpublished note, December 2021.

[12] V. D. Kupradze and Merab Aleksandrovich Aleksidze. The method of functional equations for the approximate solution of certain boundary value problems. *Ussr Computational Mathematics and Mathematical Physics*, 4:82–126, 1964.

[13] Zi-Cai Li, Yi min Wei, Yunkun Chen, and Hung-Tsai Huang. The method of fundamental solutions for the helmholtz equation. *Applied Numerical Mathematics*, 2019.

[14] Francesco Mainardi. *Fractional calculus and waves in linear viscoelasticity: an introduction to mathematical models*. World Scientific, 2010.

[15] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 9.12.0.1884302 (R2022a)*, 2021.

[16] Donald J Newman. Rational approximation to $|x|$. *Michigan Mathematical Journal*, 11(1):11–14, 1964.

[17] M. L. Rapún. On the solution of direct and inverse multiple scattering problems for mixed sound-soft, sound-hard and penetrable objects. *Inverse Problems*, 36:095014, 2020.

[18] Arnold Sommerfeld. Die greensche funktion der schwingungsgleichung. *J.-Ber. Deutsch Math.-Verein*, 21:309–353, 1912.

[19] Lloyd N. Trefethen. The definition of numerical analysis. *SIAM News*, November 1992.

[20] Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.