The Born approximation, multiple scattering, and the butterfly algorithm

Alejandro F. Martinez
University of Texas-Pan American

THE BORN APPROXIMATION, MULTIPLE SCATTERING, AND THE BUTTERFLY ALGORITHM

A Thesis

by

ALEJANDRO F. MARTINEZ

Submitted to the Graduate School of
The University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

July 2014

Major Subject: Mathematics

THE BORN APPROXIMATION, MULTIPLE SCATTERING, AND THE BUTTERFLY ALGORITHM

A Thesis
by
ALEJANDRO F. MARTINEZ

COMMITTEE MEMBERS

Dr. Zhijun Qiao
Chair of Committee

Dr. Andras Balogh
Committee Member

Dr. Tim Huber
Committee Member

Dr. Karen Yagdjian
Committee Member

July 2014

ABSTRACT

Martinez, Alejandro F., THE BORN APPROXIMATION, MULTIPLE SCATTERING, AND THE BUTTERFLY ALGORITHM. Master of Science (MS), July, 2014, 57 pages, 4 figures, 46 references, 41 titles.

Radar works by focusing a beam of light and seeing how long it takes to reflect. To see a large region the beam is pointed in different directions. The focus of the beam depends on the size of the antenna (called an aperture). Synthetic aperture radar (SAR) works by moving the antenna through some region of space. A fundamental assumption in SAR is that waves only bounce once. Several imaging algorithms have been designed using that assumption. The scattering process can be described by iterations of a badly behaving integral. Recently a method for efficiently evaluating these types of integrals has been developed. We will give a detailed implementation of this algorithm and apply it to study the multiple scattering effects in SAR using target estimates from single scattering algorithms.

DEDICATION

This is dedicated to my family and Tina for their love, support, and confidence in me.

ACKNOWLEDGMENTS

This work would have not been possible without the help of my advisor and mentor Dr. Zhijun Qiao and his imaging group for their support during my acadmenic career. I would also like to thank the many teachers in my life for their patience and ability to show the beauty of mathematics. This work was partially supported by the US Department of Defense Army Research Office under grant number W911NF-08-1-0511, the Norman Hackerman Advanced Research Program under grant number 003599- 0001-2009, and the U.S. Department of Education UTPA-MATH GAANN program under grant number P200A120256.

TABLE OF CONTENTS

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

Sensing the world is fundamental to understanding our universe. Without input from reality, we cannot hope to come up with a description of reality rooted in experiment. Without input we would never be able to know if a particular description is accurate at all. Without senses there could be no scientist, only philosophers. Vision is a very powerful sense of several living organisms. It is beneficial to a variety of animals and even micro organisms. It is also a very complicated process that has many subtle points to it. In particular, humans can see colors, track objects, approximate the distance to objects, and even identify objects. It is not a trivial topic for a machine to tell a car is actually a car, a person is a person, or a tree is a tree. Human auditory capabilities include direction and range finding. The human senses are very useful, but have several limitations. Seeing in the dark is not terribly easy for instance. During a fire, smoke occludes vision and during a storm rain lowers visibility while driving. To understand the limitations of our senses we must understand the physical process that provide them. Vision is a product of scattered light.

In order for vision to work, there must be a source that generates the light. The light travels through free space until it encounters an object which scatters some of the light. This scattered light keeps getting scattered by various objects, and whatever lands in our eye is processed by our brains to form a visual representation of the world. Each time the light scatters it loses a certain amount of energy. Intensity is also diminished as light travels through free space. Light is a particular type of Electromagnetic (EM) radiation and is governed by Maxwell's equations as long as the objects we are considering are not terribly small or moving near the speed of light. Light occupies a small region of the possible frequencies which are possible for EM radiation.

The senses we have described are remote. That is, we do not need to be terribly close to an object to get information about it. This is a good thing. Life would be very difficult if we had to lick a lion to know it was there. Some faults of human vision occur due to the interaction of visible light and small particles. This happens because the frequency band that light occupies is attenuated strongly by small objects. If we point a flashlight through smoke, the light will interact with the smoke. Another fault of human vision is the inability to see when saturated with light. If there is too much light present then we may not be able to see anything. If the sun it too bright we may be blinded.

It is possible to choose frequency bands of radiation that minimize the interaction with small particles and still promote scattering by large objects. If we control some portion of the generated radiation, we can pick out what we are interested in even when there is non negligible noise present. This is not the case with human vision. We cannot simply send a sequence of flashes and pick out the pieces of light in our eye that were scattered when there is a bright external source. This is what Radar does. Radar is the use of controlled microwave range radiation to "see" regions of interest often in low visibility scenarios.

RADAR(Radio Detection and Ranging) is an application of electromagnetism based on three principles: EM radiation travels at a finite constant speed while in a vacuum; radiation emitted from an antenna is contained in some localized region of space; targets scatter radiation. In RADAR we need to be able to measure, and often transmit radiation. RADAR is an active system for the most part meaning that we can generate radiation. There are some [40] who do study the passive (no generated radiation) capabilities of it. We will focus on active RADAR. That is to say, we know what the radiation looks like before it scatters(up to noise). To detect is the simplest task. When we listen to the radiation present we get basically random fluctuations due to noise. If there is no target present, then the energy due to the noise will not be terribly large. One could acknowledge detection of a target when measured radiation is above some noise threshold. This requires knowledge of background and system noise, but a system can be calibrated [35, 34]. More sophisticated systems are based on the fact that the time it takes for radiation to get from the antenna to a target at range R and back to the antenna is given by $t = 2R/c$ where c is the speed of light in air. This means that that each return we get corresponds to a target at range $R = ct/2$. The resolution of an imaging system is its ability to distinguish between nearby object. High resolution means we can distinguish very close targets. Radar is great for long distance high resolution ranging. The limits of the operational range of a radar system are discussed in literature and called radar equations [35]. Localization of the generated radiation can be used to resolve targets to only the region illuminated. Most of the energy generated will fall into a cone emanating from the antenna. How width of the beam of radiation is unsurprisingly called its beam-width. By pointing the beam in different directions.

There are several manufacturing problems involved in making such an idea reality. A difficult task early on was the generation of high frequency microwave radiation. The system is tied very heavily to the particular application. A system does not need to detect every type of object to perform its task. Moreover, designing a perfect system may take unreasonable long. Time is not a cheap commodity, particular during wartime. During the early days of radar a big limiting factor was the ability to generate the necessary radiation. Creating even a simple RADAR system is not terribly simple, but can be done. There are plans available for creating a rudimentary RADAR system for a few thousand dollars [5]. Plenty of research goes into the more practical aspects of creating the physical device and there are many interesting aspects in hardware that must be considered when developing an imaging device. This is the work of many engineers. For a brief history of RADAR and an introduction to the topic from an engineering standpoint see [35, 34]. We will not dwell too much on hardware aspects.

Radar is a very developed technology but does not mimic human vision at all. The human eye has several densely packed sensors which listen for light from any source. The definition of "see" must be broadened to capture breadth of tasks that radar systems are used for. A conventional Radar system has one antenna that acts as both a source of radiation and sensor. Some systems include several antennas, but not nearly as many photo-receptors as in the human eye. Radar systems can pick out our targets even in the presence of considerable noise. If we the sun is very bright, we may not be able to see objects. Vision is largely a passive scheme. Cameras are more accurate replicas of human vision and indeed there are microwave and infrared cameras being developed [17].

The beam-width of an antenna gives a sense of angular resolution. We know that any returns we get must be from inside the antenna beam and so targets that are sufficiently far apart in angle can be distinguished. Finer beams give us better angular resolution. Unfortunately beam-width is tied directly to antenna size and therefore to cost. If we want high angular resolution we need a very large and expensive antenna. This limits angular resolution in RADAR systems. Even if we had the funding for a large antenna, maneuvering it would severely limit its usefulness. These problems were evident by the early 1950s [7].

Two closely related systems were created to resolve these issues. By using multiple transmitting antennas delayed appropriately one can synthetically form and point beams. Such systems are called phased arrays [10]. These systems could electronically steer the beam of radiation as well as find the direction that the return came from. The other system developed is called Synthetic Aperture Radar (SAR).

In SAR a small antenna is moved over a region of space while sending a series of pulses. The beams from different pulses will overlap certain regions as we the antenna moves. We can use this fact to form high resolution images of the region of interest. The name is suiting since we attain angular resolution comparable to that of a much larger antenna. We essentially create a synthetic antenna out of the antenna path. These systems are what we will focus on.

A significant difference between SAR and traditional RADAR is the necessary processing time. Not much is free in life and angular resolution does not come cheap. If we don't want to pay for better hardware then we have to pay with algorithmic complexity. The scattered data of a SAR system in the frequency domain takes the form of an oscillatory integral. In estimating target information using SAR one must often evaluate these time consuming integrals. This makes SAR a very interesting topic mathematically and practically.

The history of radar is very closely connected with wartime efforts but has seen peaceful uses [35, **?**]. Civilian uses of SAR include space research. There is a sister technology which is slightly less morally ambiguous called Computed Tomography (CT). CT is similar to SAR in that both exploit spatial diversity of the antenna to produce high resolution images. In the simplest modalities of CT, one shoots a beam of photons through a body and measures the intensity on exit. This is done from several angles and lines along those angles. The theoretical solution to CT is then the inverse Radon transform. Limited data leads to artifacts that must be studied. CT is much more mathematically mature than SAR and there are several famous books on the mathematics of the topic [23, 13, 28]. There are many fast algorithms for imaging in CT that are being applied to SAR [33]. CT and SAR work at very different ranges. SAR is a long, to medium range imaging modality while CT is near range.

Light can bounce multiple times before it enters our eye. Microwave radiation can also reflect several times before it reaches the antenna. If two objects are near each other, then some radiation scattered from one of the objects will be scattered again by the other object. This is a problem. Traditional radar based on time delay will create false targets. RADAR and SAR typically rely on the Born approximation where one assumes that the recorded data only bounced once. This linearizes the imaging problem making it much more tractable. Many SAR imaging algorithms have been designed for the linearized scattering model. In the absence of multiple scattering they are efficient, and accurate. The approximation can be justified if targets are well separated, and not terribly reflective. Many targets are not well separated. Imaging of targets on the ground leads to a layover effect caused by multiple scattering. The closer the targets lie to each other, the more multiple scattering effects become apparent in image formation. When one wants to test a reconstruction algorithm they might use synthetic data in early phases of development. This data should include multi-scattering effects as well.

We have two main goals in this thesis: to give a straight forward serial implementation of the butterfly algorithm for Fourier integral operators [30, 2, 11]; to use this implementation recover image estimates to estimate multiple scattering effects with. We use the scalar wave model as the setting for electromagnetic propagation. In order to generate data, one must evaluate the scattering operator many times. Not only that, we must evaluate this operator near our targets, so far field estimates fail. This is a continuation of previous work [27].

Remote sensing in general is intimately connected to a probabilistic world. While this is an important aspect of reality that must be considered at some point we will not explore it. Since its invention radar has been very concerned with sensing moving targets. This adds difficulty to modeling multiply scattering data. Typically one wants to derive a Doppler style shift in frequency of the transmitted radiation. This is done for simple wave forms in [39]. We do not include moving targets in our model. Attaining high resolution space and velocity images simultaneously is a very active area of research and is an interesting direction to study in the future. We will not discuss regularization methods at this point. One may also consider several transmitting and receiving antennae. We previously discussed a basic image formation algorithm for MIMO RADAR based on the discrete Fourier transform and the wave equation in [4]. Recently there has been much work in the topic of compressive sensing by some of the same authors that developed the butterfly algorithm we use. This is concerned with solving under determined linear systems for the sparsest solution via convex optimization. MIMO RADAR is an area that has seen considerable application of this tool as discussed in [24]. If we can separate the waveforms generated by the system, then we are simply collecting data much more efficiently with multi-static SAR [4]. Another area related to SAR is Inverse SAR (ISAR), and turntable SAR [26] in which the antenna is stationary and the targets are moving.

1.1 Thesis Chapter Outline

The rest of this thesis is as follows:

Chapter 2:

In this chapter we model synthetic aperture radar and multiple scattering.

Chapter 3:

We consider numerical aspects of our problem. These include discretization, computation of the scattering series. The imaging solution will be posed in the form the butterfly algorithm is designed to evaluate.

Chapter 4:

We discuss low rank interactions and the butterfly algorithm. We give a detailed description of an implementation of the algorithm.

Chapter 5:

We conclude this thesis with a discussion of work that has been done to model and mitigate multiple scattering. We will discuss the results from our simulation and discuss future work.

Appendix:

Here we provide an implementation of the butterfly algorithm in C++.

CHAPTER II

SYNTHETIC APERTURE RADAR AND MULTIPLE SCATTERING

2.1 Introduction

These equations can be difficult to work with for imaging. One can show under certain conditions(no polarization) [7, 26, 18, 22] that we can work with an inhomogeneous wave equation instead. Following the literature we adopt this as our model. The goal in SAR imaging is to get detailed information about a region of interest by combining different antenna positions and pulses. One must model the data that we receive from this process. This model is then used to infer information about a region of interest. There are different models that are applicable to this problem depending on the scale are working at. In the macroscopic regime the appropriate model for electromagnetic radiation is Maxwell's equations. A classic treatment of the full equations is given in [37]. In this setting the types of materials we are looking at become very important since they dictate how our model behaves. A much simpler model that is appropriate for certain types of materials is the scalar wave equation. This is what we work with. We will give some reasoning why we use this. We will briefly review Maxwell's equations and show that they contain the scalar wave when dealing with certain types of object. Then we will consider the inhomogeneous scalar wave equation as the model for wave propagation. The source term which we control will correspond to the generated pulses. The solution to the equation evaluated along our flight path is what we measure. The radiation propogates information from our antenna to some targets. The targets change this information and propagate some back to the antenna. The goal is then to make the changed information useful. We will derive this solution and its multiple scattering components. We discuss the Born approximation in this context and simple theoretical image formation algorithms based on it. A big difference between theory and practice is the discretization of this algorithms. This will be discussed in chapter 3.

2.2 Synthetic Aperture Radar Overview

In our setup we will assume that the antenna is point like (no directionality to the antenna) and travels through a path $\gamma(t) : [0,1] \to \mathscr{R}^3$. We call γ the flight path of our antenna even if it is not mounted on a plane. This can be extended to more practical antenna configurations but we will focus on this simple setup to ease exposition. The antenna will generate a sequence of pulses of radiation as it moves through its path. To simplify matters even more we use the common start stop approximation where we treat the antenna as stationary in between pulses. That is, we think of the antenna as moving to a point, stopping, generating radiation, listening for returns, and then moving and repeating the process until it has completed its path. This is like walking to a spot, turning on a laser, then turning it off and looking for reflections, then moving to a new spot. The motion that does occur while the pulse is being transmitted causes error. This motion can be estimated by Global Positioning System (GPS) or other methods and be used to compensate for the approximation. The radiation generated will interact with materials and some will be returned to the antenna. This constitutes the data. We need to describe how our data is related to the generated radiation and objects interacting with them. We will assume that the objects we are interested in and the radiation we generate are of appropriate size that we can ignore quantum and relativistic effects. The appropriate model is then Maxwell's equations. We must also consider the particular types of targets we are looking for. We cannot consider all possible functions since there are very pathological examples. Targets that do not vary much locally are what is considered.

2.3 Electromagnetic Radiation

The phenomena that radar is based in is electromagnetism. Following [37] we assume the existence of time and space dependent fields satisfying

$$\nabla \times E + \partial_t B = 0 \tag{2.1}$$

$$\nabla \times H(B) - \partial_t D(E) = J. \tag{2.2}$$

for some given current density J at all points inside a domain $D \subset \mathscr{R}^3$ and all times. These are called Maxwell's equations.

The analytic properties of the fields and current must be considered. We want to be able to do Fourier analysis of electromagnetic radiation in time. Namely we want to deal with time harmonic fields. To this end we assume that each component of the fields are square integrable. We also assume that that the second partials of the fields and densities are all continuous so we can interchange derivatives. On the boundaries of different media we assume material properties vary nicely(continuously) but very quickly.

Fundamental to this model is charge. The average charge in an arbitrarily small volume around a point is represented by a charge density function p. Care must be taken here since charges are not arbitrarily small. We must keep in mind that there is no such thing as charge at a point, but average charge over a suitably large volume.

The current density is related to charge by the continuity equation

$$\nabla \cdot J = -\partial_t p \tag{2.3}$$

which can be derived using some physical hand waving, the divergence theorem, and conservation of charge. This represents the analogy of current being the flow of charged particles.

Recall that the divergence of a curl is zero. Taking divergence of each of Maxwell's equations we get $\nabla \cdot \partial_t B = 0$ and $\nabla \cdot J + \nabla \cdot \partial_t D = 0$. Then $\partial_t \nabla \cdot B = 0$ and $\nabla \cdot J + \partial_t \nabla \cdot D = 0$. The first equation implies that $\nabla \cdot B$ is time independent and so if the magnetic field ever vanishes then the divergence must be zero for all time

$$\nabla \cdot B = 0. \tag{2.4}$$

The second tells us that $\nabla \cdot J + \partial_t \nabla \cdot D = \partial_t (\nabla \cdot D - p) = 0$ so that $\nabla \cdot D - p$ is time independent. One can argue(while waving their hands) that it must be zero so that

$$\nabla \cdot D = p. \tag{2.5}$$

The two curl and two divergence equations are sometimes called Maxwell's equations, but they are not independent if we assume the continuity equation(conservation of charge).

For completeness we should mention the relationship between current and the field vectors.The current density J is dependent on E and H but in many cases media the dependence on H is negligible. In many cases the relationship between J and E is given by

$$J(x,t) = \sigma(x,t)E(x,t) \tag{2.6}$$

where σ is called the conductivity of the medium.This equation is called Ohm's law.Conductivity is a material property and this can actually be seen as a definition of it.

The equations we have governing electromagnetism are all first order partial differential equations.These are not enough to determine the fields uniquely since there are 12 unknowns and only 9 not necessarily independent equations.To even hope of having a unique solution to this model we need more equations or less unknowns.

We need to introduce constitutive relationships that describe the dependence of H and D on their arguments. We want the equations to only involve material parameters and the fields H and E.The media that fields are permeating dictates these relationships so we must discuss the type of materials we are willing to work with.

The relationships may be nonlinear, but this is not typical. At very high field values above the Schwinger limit, nonlinearity may show up in vacuum [19]. Such effects are not producible on this planet. If we encountered such fields we would have more pressing problems than imaging. Many materials that are nonlinear are only weakly so. This is not true of all materials. Meta materials are very advanced materials which have a strong nonlinear response. These are the basis of the current work into invisibility cloaks and cloaking. No doubt they will be vital in the future of stealth. That said, applications of nonlinear materials in practice are still quite limited. We will not address nonlinear materials, but for those interested a recent survey on their uses in cloaking and invisibility is [15].

Another common type of material is called anisotropic. These are materials where the relationship is some matrix multiplying the electric and magnetic fields. Such materials are common in military aircraft. Though we may run into these in practice, they pose significant difficulty in modeling. We want to focus on multiple scattering effects so we want a simpler model.Namely we consider isotropic media.

We restrict ourselves to isotropic media. This is media where the relationships are given by

$$D = \varepsilon(x,t)E(x,t) \tag{2.7}$$

$$H = \frac{B(x,t)}{\mu(x,t)}. \tag{2.8}$$

By material properties we mean ε and μ. In homogeneous media the material properties have approximately zero gradient. Free space is a homogeneous media that is also independent of time with the relationships given by

$$D = \varepsilon_0 E(x,t) \tag{2.9}$$

and

$$H = \frac{B}{\mu_0}. \tag{2.10}$$

We want to rework Maxwell's equations to get rid of two of the fields using the relations we just introduced. We also would like to to convert these into second order partial differential equations that are hopefully easier to work with. This will require vector calculus. We list the identities that we will use for completeness. Let A be any $3-d$ vector valued function, and ψ be a scalar valued function. The product rules for vector derivatives are given by

$$\nabla \cdot \nabla \times A = 0 \tag{2.11}$$

$$\nabla \times \nabla \psi = 0 \tag{2.12}$$

$$\nabla \cdot \nabla \psi = \nabla^2 \psi \tag{2.13}$$

$$\nabla \times \nabla \times A = \nabla(\nabla \cdot A) - \nabla^2 A \tag{2.14}$$

$$\nabla \cdot (\psi A) = \psi \nabla \cdot A + A \cdot \nabla \psi \tag{2.15}$$

$$\nabla \times (\psi A) = \psi \nabla \times A + \nabla \psi \times A \tag{2.16}$$

and if B is also a vector then

$$\nabla \times (A \times B) = A(\nabla \cdot B) - B(\nabla \cdot A) + (B \cdot \nabla)A - (A \cdot \nabla)B. \tag{2.17}$$

Since we are dealing with homogeneous media, the gradients of the material parameters are zero. Since we are also assuming that all objects are stations while the radiation is propagating, the time derivatives of the material parameters must be zero. The constitutive relations and the product rules let us rewrite Maxwell's and the divergence equations as

$$\nabla \times E + \mu \partial_t H = 0 \tag{2.18}$$

$$\nabla \times H - \varepsilon \partial_t E = J \tag{2.19}$$

$$\mu \nabla \cdot H = 0 \tag{2.20}$$

$$\varepsilon \nabla \cdot E = p \tag{2.21}$$

where the equations now only involve material properties, H, and E. Taking the curl of the first of Maxwell's equations and applying the product rules we get $\nabla(\nabla \cdot E) - \nabla^2 E + \mu \nabla \times H + \nabla \mu \times H = \frac{\nabla p}{\varepsilon} - \nabla^2 E + \mu(\partial_t J + \varepsilon \partial_t^2 E) = 0$ which can be written as

$$\nabla^2 E - \mu\varepsilon \partial_t^2 E = \frac{\nabla p}{\varepsilon} + \mu \partial_t J \tag{2.22}$$

using the continuity of the derivatives and the curl of curl identity. This is the inhomogeneous vector wave equation. Taking the curl of the second of Maxwell's equations gives us $\nabla(\nabla \cdot H) - \nabla^2 H - \varepsilon \partial_t \nabla \times E = -\nabla^2 H + \varepsilon\mu \partial_t^2 H = \nabla \times J$ which becomes

$$\nabla^2 H - \varepsilon\mu \partial_t^2 H = -\nabla \times J \tag{2.23}$$

Again this is the vector wave equation for the magnetic field with the source term J. When we look at H and E in rectangular coordinates the vector Laplacian turns into taking the scalar Laplacian of each component. We assume that our targets have no current present. In both wave equations the source terms depend on current and charge density. If there is no charge or current outside of our antenna, then we have complete control over the source term in each of these wave equations. If this is not the case the source term will have a random component. The goal of imaging is to derive information about the material parameters given measurements of H and E. As we will see, each wave equation(there are 6) can be used to recover information about these parameters. We will denote $\frac{1}{\varepsilon\mu}$ by c. This is the speed of propagation in the media. The assumption that the media is homogeneous and isotropic may seem fairly strong. Most materials look constant if we are at a fine enough scale. This is what the assumption amounts too. As long as the scale necessary to make materials look piecewise constant is not small enough to induce quantum effects our model should be reasonable.

2.4 Wave Propagation: Continuous Model

Our focus will be on image reconstruction using only one component of the field equations. There are downsides to this. Much information is contained in the other 5 wave equations. Working with one component ignores useful effects such as polarization but captures the wave nature of electromagnetism. There are several more simplifications to the model that will be made.

Simplifying the model leads to more efficient algorithms at the cost of accuracy. That said, the simplifications capture several important features of the scattering process and are more manageable than the full Maxwell's equations. We must keep in mind, that SAR is tied very heavily to its particular application, and this controls what approximations are reasonable. We have assumed that there are no moving targets. This corresponds to a short flight path. The amount of time we can assume that targets are stationary, or more generally that the model is maintained is sometimes called the coherent processing time (CPI) in the literature. There is motion present in almost any scenario and this induces a Doppler shift in the return frequency. Antenna motion can be compensated for since we know what the motion looks like. Target motion is more difficult to model. Simple(along a line) motion can be incorporated into the wave model as seen in [39]. This assumed that target motion was a diffeomorphisms since it should be reversible. The authors then derive Doppler effects due to antenna and Doppler motion. The effects are what should be expected, but the wave model itself seems to be based on stationary media. We will not deal with these effects. There are features of electromagnetic radiation that we will not exploit. For one we make no use of the rotation (polarization) of the wave as it propagates toward a target. We have also assumed a very restricted set of targets that we can consider.

The wave model and its solutions have been tested and worked fairly well under a variety of real world circumstances. There are several different viewpoints taken in the literature of the imaging problem. In more practical(engineering) literature one typically models what the return from a point like object is and then treats the as superpositions of returns from point like targets. The incident wave is often assumed to be planar as well [12]. The strength of these lie in their ease of discretization. The problem is already discrete since we model targets as a finite superposition of point like objects. Several authors begin with the wave equation and derive algorithms and analysis of these algorithms based on linearizing the inverse problem in this wave model [7, 40]. Other authors seek indirect information about targets. Particularly [8] seeks the support of targets using the frequency domain wave model with the nonlinear inverse problem. Several algorithms that are used in practice for detecting stationary objects in synthetic aperture radar can be derived from the wave equation. Recently there has been a lot of interest in applying sparse solution methods to imaging problems. Fields medalist do that [3]. The theoretical implications are quite nice. The idea is that if we know what much of the space we are trying to image is empty, then we can efficiently recover the non empty part even if we don't where it is. The idea is that since one usually takes a signal(some high dimensional vector), compresses it(represents it by a much lower dimensional vector), and throws away a large amount of data, then why should they need the high dimensional signal in the first place. Shouldn't they be able to represent the signal without taking so many measurements.

The answer depends on the model for the signal. The theory is based on linear sampling schemes. If the matrix representing the transformation from true vector to the samples(an under determined system) satisfies certain properties then the answer is probably. We will not exploit sparsity but this is a very active research field.

The wave nature of radio waves give very important information about objects we want to see. The inhomogeneous wave equation or its solutions are the typical starting point for SAR image reconstruction [11, 6, 7, 4, 16, 32, 29, 26, 25, 36, 40, 9, 12]. This model describes a linear scattering problem, but an ill-posed nonlinear inverse problem. Under certain assumptions we can develop very efficient imaging algorithms. More general materials could make the scattering problem terribly non-linear and difficult. Even handling anisotropic media gets fairly involved. The problem we consider is reconstruction of the wave speed from limited measurements of the field by controlling the source.

The homogeneous wave equation in 3 dimensions is given by

$$\left(\nabla^2 - \frac{\partial_t^2}{c(x)^2}\right) u = -j(x,t) \tag{2.24}$$

where $c(x)$ is the speed of light at x. The source term is what we get to play with to see how the field at the antenna is affected by the wave speed of our targets. The antenna transmits N_s pulses and is stationary during each transmission and measurement cycle. The wave equation should have a source term containing to all the pulses and this would generate one field that we take measurements from. If each cycle is long enough that any effects u feels from previous pulses will by gone by the next pulse then we can treat each pulse independently. We parametrize the current pulse by $s \in \{1, \ldots, N_s\}$ which we call the slow time variable. For each pulse s there is a field $u_s(x,t)$, an antenna position $\gamma(s)$, and a source $j_s(x,t)$ that satisfy

$$\left(\nabla^2 - \frac{\partial_t^2}{c(x)^2}\right) u_s(x,t) = -j_s(x,t). \tag{2.25}$$

The data from the s-th pulse with an isotropic antenna is simply $u_s(\gamma(s),t)$. To incorporate more general antennas we would need to take a weighted average of $u_s(x,t)$ over the antenna. In this ideal scenario was stated earlier we are assuming that we have complete control over $j_s(x,t)$ and that our antenna is point like. That is, $j_s(x,t)$ is zero everywhere except at $\gamma(s)$ and we are there is no noise. There is still ambiguity here. There is not necessarily a unique solution to this problem. Both $u_s(x,t)$ and $j_s(x,t)$ are required to have well defined Fourier transforms in time for each point in space. To give meaning to a point like source that generates a non zero field, we need to treat $-j(x,t)$ as a distribution. This is a sloppy notion since we are assuming the solution corresponds to a point wise function even though the equation is distributional. By a point source we located at $\gamma(s)$ we mean $j_s(x,t) = \delta_\gamma(s)p_s(t)$ where $p_s(t)$ is the input signal and is only non zero for some finite interval. We assume that $p_s(t) \in \mathbf{L}^2[0,1]$ since this is a natural space for analysis of compactly supported signals. This means that $p_s(t)$ has a meaningful Fourier transform $\hat{p}_s(\omega)$. Since $p_s(t)$ has compact support, its Fourier transform will never stay identically zero. That said, if $p_s(t)$ is chosen properly we can center the effective support of its Fourier transform in a compact set around a point. Assuming that the Fourier time transform of any solution to 2.25 exist and so do the transforms of its time and space derivatives, then taking taking Fourier transform of both sides we get

$$\left(\nabla^2 + k^2\right)\hat{u}_s(x,\omega) = -\hat{j}_s(x,\omega) \tag{2.26}$$

where $k(x) = \frac{\omega^2}{c^2(x)}$.

We define the reflectivity [7, 6, 9, 26, 4] function by

$$v(x) = \frac{1}{c_0^2} - \frac{1}{c(x)^2}. \tag{2.27}$$

The reflectivity function gives us an idea of what our target distribution looks like. It is zero in the absence of targets, characterizes the brightness of an object. The goal of RADAR imaging in this wave model is to solve for (2.27). The reflectivity function must be discontinuous in almost any real setting, but we assume it can be modeled as a continuous function that varies very quickly in thin transition regions. The reflectivity function is very closely tied to the field $u_s(x,t)$. In a weak scattering setting v can be determined from u_s. The reflectivity is also related to the index of refraction of a material by

$$v(x) = 1 - n(x)^2. \tag{2.28}$$

The largest refractive index of a material produced on earth as of 2011 was 38.6, so it is reasonable to say v is bounded.It is also piecewise constant since $c(x)$ is. We will assume that v is compactly supported since the incident radiation can only interact with a compact region before dissipating. This means that $v(x)$ is square integrable and has a meaningful Fourier transform. To simplify things let us suppose that our targets are contained in the unit cube $[0,1]^3$ so that $v(x) \in \mathbf{L}^2([0,1]^3)$. In free space the wave speed is given by $c_0 = 299792458 m/s$. When a wave encounters a media the energy may either reflect, be absorbed, or pass through. Absorption corresponds to complex valued reflectivity. We will not consider this and assume $c(x)$ us real valued. Since we have embedded the argument of v in the unit cube, the units we are measuring in will be in terms of the size of the support of v. For example, if the targets are contained in a 100 meter length cube, then the space variable must be in $100m$ units(i.e. Hectometers).

Using the reflectivity function we can rewrite the 2.26 equation as

$$(\nabla^2 + k_0^2)\hat{u_s}(x,\omega) = -(j(x,\omega) - \omega^2 v(x)\hat{u_s}(x,\omega)) \tag{2.29}$$

where $k_0 = \frac{\omega^2}{c_0^2}$. When $v(x) = 0$ this is called the Helmholtz equation.

We require that the solution $\hat{u_s}$ to 2.29 satisfies the Sommerfeld radiation condition

$$\lim_{|x|\to\infty} |x| \left(\frac{\partial}{\partial |x|} - ik \right) \hat{u_s} = 0. \tag{2.30}$$

This ensures that $\hat{u_s}$ away from the target support is unique [9]. Such solutions are said to be radiating, move away from objects that scatter them, and decay at a rate $\frac{1}{r}$ where r is the distance traveled.

The radiation condition implies that the solution to the Helmholtz equation is given by

$$u_s(x,\omega) = G*_x j(x,\omega) - \omega^2 G*_x v(x)\hat{u_s}(x,\omega) \tag{2.31}$$

where

$$G(x,\omega) = \frac{e^{-ik_0||x||}}{4\pi||x||} \tag{2.32}$$

is the Green's function for the Helmholtz equation and $*_x$ denotes convolution over space. This is the Lippmann Schwinger equation. We call $G*_x\hat{j}_s(x,\omega)$ the incident field and denote it by $\hat{u_s}^{in}(x,\omega)$. We control the incident by varying the source term. That is, we think of the incident field as the portion of u that satisfies the Helmholtz equation in free space. This can be found explicitly as

$$u_s^{in}(x,\omega) = \hat{p}_s(w)G(x-\gamma(s),\omega). \tag{2.33}$$

Let use define the scattering operator L which depends on both v and a given field u_s by

$$L_v(\hat{u_s}) = -\omega^2 G*_x v(x)\hat{u_s}(x,\omega). \tag{2.34}$$

This is linear in both v and u_s so can be viewed as a bilinear operator. The Lippmann Schwinger equation can be rewritten as

$$(I-L_v)\,\hat{u_s} = \hat{u_s^{in}} \tag{2.35}$$

where I is the identity operator. This can solve by Neumann series provided the operator L_v behaves nicely for every v. One condition [7] is is that the norm of L_v be less than 1 similar to the geometric series. This is fairly restrictive and is not a necessary condition for the series acting on a particular incident field to converge. Since we have control over the set of incident fields we need not consider convergences for all possible incident fields. Letting $L^0 = I$ and $L_v^n(\hat{u_s}^{in}) = L_v^{n-1}(\hat{u_s}^{in})$ a less restrictive condition for the convergence of

$$\hat{u_s} = \sum_{n=0}^{\infty} L_v^n(\hat{u_s}^{in}) \tag{2.36}$$

to a solution of 2.35 is that $L_v^n(\hat{u_s}^{in})$ goes to 0 strongly as $n \to \infty$ and L is completely continuous [38]. We are not terribly concerned with the exact conditions when this series converges. This is typical of this subject in the engineering literature. Convergence properties of this series are fairly difficult results to get and this is still a fairly open area of research. We assume that the series is convergent, This means that the effective frequency support of our incident field cannot be too high, or our targets cannot be terribly large and reflective. We call 2.36 the scattering series and consider it the solution to 2.29 that we are measuring.

Each term in the series corresponds to a scattering event. The first term in the series corresponds to incident field interacting with the targets and getting scattered. The second term corresponds to this first scattered field again interacting with targets and getting scattered again. In general the n-th term is the $n-1$-th term being scattered. The field $\hat{u_s}$ is then the sum of waves that were scattered n times.

As time progresses the incident wave moves away from the source in a spherical manner spreading out energy over this surface. The portion of the sphere that overlaps an object gets scattered. Some of the energy scatters in the direction of our antenna, and some in the direction of other objects. The wave then continues and may scatter many times in this manner. We measure the sum of these scattering events at our antenna.

The difference $\hat{u_s} - \hat{u_s^{in}}$ is often called the scattered field. This consist of only the part of $\hat{u_s}$ that was scattered by the targets. The multiple scattering components(powers of L with $n = 2, 3, \ldots$) of $\hat{u_s}$ make the imaging problem nonlinear. This is hard to work with. The most common method for recovering v is by performing some type of error minimization over the space of possible functions v. This is not very efficient yet. The target support can be estimated using methods developed by [9]. These methods require large amounts of data and require that the incident field has no curvature to it. They are efficient and handle multiple scattering.

To summarize, our data should be $\hat{u_s}(\gamma(s), \omega)$ sampled in frequency. There is a bit more thought that needs to go into this when we talk about sampling in ω. This will be saved for the next chapter. For now we will pretend we have $\hat{u_s}(\gamma(s), \omega)$ for all ω. Note that $\hat{u_s}(\gamma(s), \omega) = 0$ if $p_s(\omega) = 0$. Similarly if we consider frequencies outside of the effective support of $p_s(\omega)$, then $\hat{u_s}(\gamma(s), \omega)$ should be negligible.

The purpose of the Born approximation is to remove the non-linearity in the dependence of

v on j by approximating the total field by the incident field. Polynomial operator equations are hard to solve just as regular polynomials are. The waves induced lose energy as they travel due to the radiation condition. This means each scattering event must be weaker and weaker since the wave must travel more to scatter several times. Whenever we try to solve for the reflectivity we will need to truncate the scattering series. Taking only the first two terms is called the Born approximation and is typically done to linearize the problem thereby simplifying it considerably.

2.5 Image Formation From Single Scattered Data

Let us focus on imaging in the absence of multiple scattered data. To simplify things further we will assume that our targets live on the ground and are very thin so that we can consider v as two variable function. We will use the same v to represent the function of 2 variables, and the function of 3 variables. Then assuming that we have removed the incident field already, the data is given by

$$\hat{d}(s,\omega) = -\omega^2 p(\omega) \int_{[0,1]^2} G(\gamma(s)-z,\omega)^2 v(z)\, dz. \tag{2.37}$$

Assuming that $\|\gamma(s)\| >> \|z\|$ for any $z \in [0,1]^2$, one may make the approximation $\|\gamma(s)-z\| \approx \|\gamma(s)\| - \hat{\gamma}(s)\cdot z$, and $\frac{1}{\|\gamma(s)-z\|} \approx \frac{1}{\|\gamma(s)\|}$ where $\hat{\gamma}(s) = \frac{\gamma(s)}{\|\gamma(s)\|}$ which leads to

$$G(\gamma(s)-z,\omega) \approx \frac{e^{-ik_0\|\gamma(s)\|}e^{ik_0\hat{\gamma}(s)\cdot z}}{4\pi\|\gamma(s)\|}. \tag{2.38}$$

This is called the far field approximation. The far field data approximately contains Fourier transform samples of v

$$d(s,\omega) \approx -\omega^2 \hat{p}_s(\omega)\frac{e^{-i2k_0\|\gamma(s)\|}}{(4\pi\|\gamma(s)\|)^2}\int_{\mathscr{R}^2} e^{i2k_0\hat{\gamma}(s)\cdot z} v(z)\, dz = -\omega^2 \hat{p}_s(\omega)\frac{e^{-i2k_0\|\gamma(s)\|}\hat{v}(2k_0\hat{\gamma}(s))}{(4\pi\|\gamma(s)\|)^2}. \tag{2.39}$$

We have used the positive sign in the Fourier space transform. In particular

$$\hat{v}(2k_0\hat{\gamma}(s)) \approx -\frac{d(s,\omega)e^{i2k_0\|\gamma(s)\|}(4\pi\|\gamma(s)\|)^2\overline{\hat{p}_s}(\omega)}{\omega^2|\hat{p}_S(\omega)|^2} \tag{2.40}$$

where $\overline{\hat{p}_s}$ is the complex conjugate. This holds so long as $\hat{p}_S(\omega)$ is not zero. If it is zero, then there is no data on that frequency. Since v is square integrable its Fourier transform is invertible so

$$v(x) = \left(\frac{1}{2\pi}\right)^2 \int_{\mathscr{R}^2} e^{-ix\cdot\xi}\hat{v}(\xi)\, d\xi. \tag{2.41}$$

If the flightpath is a circle(i.e. $\gamma = r(\cos(2\pi t), \sin(2\pi t))$ with $t \in [0,1]$, and $r > 0$) on the ground then the samples of the Fourier transform of v are in polar coordinates. It is not centered on the middle of targets, but rather at $(0,0)$. This means that the distance to the antenna is fixed at r. The radius must be chosen so that the flight path does not pass through the target support. If we have access to all frequencies from $[0,\infty)$, then we can exactly recover $v(x)$ by taking an inverse Fourier transform. Usually we will send the same type of signal during each pulse since otherwise we would need to design complicated hardware. We will deal with this setting and drop the dependence of $\hat{p}_s$ on s and simply write $\hat{p}$. Such a system is also called coherent in literature since we are sending the same pulse. Consider the change of variables $\xi = \frac{2\omega\hat{\gamma}(t)}{c_0}$ from ξ to (ω,t). Then the Jacobian is

$$\frac{\partial \xi}{\partial(\omega,t)} = \det(\begin{pmatrix} \frac{2\cos(2\pi t)}{c_0} & -\frac{4\pi\omega\sin(2\pi t)}{c_0} \\ \frac{2\sin(2\pi t)}{c_0} & \frac{4\pi\omega\cos(2\pi t)}{c_0} \end{pmatrix}) = \frac{8\pi\omega}{c_0^2}. \tag{2.42}$$

Then we can integrate the Fourier transform in terms of t and k_0 by

$$v(x) = \frac{2}{c_0^2\pi}\int_0^\infty\int_0^1 e^{-ix\cdot\frac{2\omega\hat{\gamma}(t)}{c_0}}\hat{v}(\frac{2\omega\hat{\gamma}(t)}{c_0})\omega\, dt\, d\omega. \tag{2.43}$$

We re parameterize the flight path with $s \in [0,1]$ instead of an integer by dividing by the number of pulses. Then we have a reconstruction formula in the form of

$$v(x) = -\frac{32\pi r^2}{c_0^2}\int_0^1\int_0^\infty \frac{e^{-i\omega(\frac{2(x\cdot\hat{\gamma}(s)-r)}{c_0})}\overline{\hat{p}}(\omega)d(s,\omega)}{\omega|\hat{p}(\omega)|^2}\, d\omega\, ds. \tag{2.44}$$

Image formation then amounts to averaging the Fourier inverses of $\frac{\overline{\hat{p}}(\omega)d(s,\omega)}{\omega|\hat{p}(\omega)|^2}$ at different points in the slow time(s) domain. We could go ahead and just perform a bunch of 1 dimensional inverse Fourier transforms using the fast inverse Fourier transform (IFFT) but these algorithms are designed for evaluation onto a rectangular grid. We would need the points along $2(x\cdot\hat{\gamma}(s)-r)$ for all x and s. This is not terribly efficient. Rather we will use the butterfly algorithm described in chapter 4 to evaluate the oscillatory integral. It must be kept in mind that the generated signal will typically have finite effective bandwidth. That is, $\hat{p}(\omega)$ is very near zero outside of some interval $[\omega_0 - B/2, \omega_0 + B/2]$ where B is the bandwidth and ω_0 is called the carrier(or center) frequency.

Then we the image reconstruction formula becomes

$$v(x) = -\frac{32\pi r^2}{c_0^2} \int_0^1 \int_{\omega_0 - B/2}^{\omega_0 + B/2} e^{-i\omega\left(\frac{2(x\cdot\hat{\gamma}(s)-r)}{c_0}\right)} D(s,\omega)\, d\omega\, ds. \tag{2.45}$$

We would like to normalize the limits of integration to the unit square so we perform the change of variables $\omega = \omega_0 - \frac{B}{2} + qB$. Then $d\omega = dqB$

$$v(x) = -\frac{32B\pi r^2}{c_0^2} \int_0^1 \int_0^1 e^{-i(\omega_0 - \frac{B}{2} + qB)\left(\frac{2(x\cdot\hat{\gamma}(s)-r)}{c_0}\right)} D(s,\omega_0 - \frac{B}{2} + qB)\, dq\, ds. \tag{2.46}$$

To test this the problem needs to be made discrete. This is what we do next.

2.6 Conclusions

We established a data model and an image formation algorithm based on single scattering. We have shown that in the absence of noise and multiple scattering effects this algorithm can be used to reconstruct reasonable image estimates. One can take into account more general scenarios. Multiple antennas can be handled and their usefulness for static targets rely on being able to separate the piece of field from a particular antenna [4]. In the next chapter we will discuss how to to discretize the forward problem to exploit the butterfly algorithm for data generation.

CHAPTER III

DISCRETIZATION AND COMPUTATION

Models that exploit analysis are generally continuous(in that they exploit the continuum). Digital processing has made many problems more tractable financially since by moving focus on software test before experimentation. We must convert the problem into a discrete form. There are two goals of this chapter. We need to discretize the generation of data with multiple scattering effects. We also discretize image formation. The imaging problem will be written in the form of a discrete FIO to exploit the butterfly algorithm.

3.1 Data Generation

To construct data we need to be able to solve the inhomogeneous wave equation. The solution was give by Neumann series and consist of repeated application of what we called the scattering operator. This series must be truncated to perform computation. The truncation to K terms is denoted $\hat{u_s}^K(\omega)$. The problem is to compute

$$\hat{u_s}^K(x,\omega) = \sum_{k=0}^{K} L_v^n(\hat{u_s}^{in})(x,\omega) \tag{3.1}$$

for some large N. Fast computation corresponds to finding an efficient algorithm for computing a truncated geometric series with a very expensive multiply operation. The series can be factored to speed up computation. Consider the geometric series to $2^k - 1$ terms

$$s_k = 1 + x + x^2 + x^3 + \cdots + x^{2^k-1}. \tag{3.2}$$

Horner's [20, 31] method may be used to evaluate a general $2^k - 1$ degree polynomial in $2^k - 1$ multiplications and additions by writing the sum as

$$s_k = 1 + x(1 + x(1 + \cdots (1 + x) \cdots)) \tag{3.3}$$

and evaluating it recursively as

$$s_j = 1 + x s_{j-1}. \tag{3.4}$$

The series can also be evaluated in the form

$$s_k = (1 + x^{2^{k-1}}) \cdots (1 + x^2) \cdot (1 + x) \tag{3.5}$$

very efficiently if we can find $x^{2^{j+1}}$ by squaring x^{2^j}.

$$\hat{u_s}^N(x,\omega) = \hat{u_s}^{in}(x,\omega) + L(\hat{u_s}^{N-1}(x,\omega)) \tag{3.6}$$

with

$$\hat{u_s}^0(x,\omega) = \hat{u_s}^{in}(x,\omega). \tag{3.7}$$

The grunt of the work is then repeated application of the scattering operator. The scattering operator is give by

$$L_v(\hat{u_s})(\omega) = -\int_0^1\int_0^1 \omega^2 \frac{e^{-ik_0\|x-z\|}}{4\pi\|x-z\|} v(z)\hat{u_s}(z,\omega)\, dz \tag{3.8}$$

for $\omega \in [\omega_0 - B/2, \omega_0 + B/2]$.

The integral has a singularity when $x = z$. In spherical coordinates with $r = \|x - z\|$ the singularity goes away as long as $x \neq z$. The weight of this point is negligible so we can interpret the integral as a limit that ignores the diagonal by integrating in the complement of a small ball centered at x. That is, whenever $x - z$ is sufficiently small we will ignore contributions in the scattering operator.

The discretization of the integral depends on how we want to discretize v. The easiest thing we can do is use a Riemann sum and partition the unit square into N^2 boxes of size $1/N \times 1/N$ and say that v is piecewise constant in each region. Of course v is probably not of this form. We could pick any value in the box and the Riemann sum would still work. We could approximate v inside that box using interpolation or some other scheme. We will stick with the simple uniform partition. The i, j-th box has center $(i/N + 1/2N, j/N + 1/2N)$ where $i, j \in \{0, 1, 2, \ldots, N-1\}$. These will be the points that we know v at when generating data. The field evaluated on the flight path has two variables ω, and s. The field inside the support of v has three variables ω, s, and x.

The scattering operator is then approximately

$$L_v(\hat{u_s})(x,\omega) = -\frac{\omega^2}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\frac{e^{-i\frac{\omega\|x-z_{nm}\|}{c_0}}v(z)\hat{u_s}(z,\omega)}{4\pi\|x-z_{mn}\|} \tag{3.9}$$

where $z \neq x$.

The number of slow time(s) and frequency ω samples we need depends on the resolving power we expect to see in image reconstruction. They will be denoted by N_s and N_ω respectively. The samples will be uniform so that $s_i = i/N_s + 1/2N_s$, and $\omega_j = \omega_0 - B/2 + jB/N_\omega + 1/2N_\omega$ where $i = 0,1,2,\ldots N_s - 1$ and $j = 0,1,2,\ldots N_\omega - 1$. The difficulty in evaluating this is that to generate data we need to evaluate the field due to each pulse along the flight path and on the target support. This means for each point in the support of v we need to compute the field for all ω and all s. This is not cheap at all and is not capable of exploiting the butterfly algorithm. For this reason we will assume that the support of v is a fairly small set, and just evaluate directly. There are two problems then:

1. Compute $L_v(\hat{u_s})(\gamma(s_i),\omega_j)$ for $\omega_j = \omega_0 - B/2 + jB/N_\omega + 1/2N_\omega$ to generate data contribution due to the current scattering event

$$L_v(\hat{u_{s_i}})(\gamma(s_i),\omega_j) = -\frac{\omega_j^2}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\frac{e^{-i\frac{\omega_j\|\gamma(s_i)-z_{mn}\|}{c_0}}v(z_{mn})\hat{u_{s_i}}(z_{mn},\omega_j)}{4\pi\|\gamma(s_i)-z_{mn}\|}. \tag{3.10}$$

   We only add the terms where $v(z_{mn) \neq 0}$.

2. Compute $L_v(\hat{u_s})(x_{pq},\omega_j)$ for $\omega_j = \omega_0 - B/2 + jB/N_\omega + 1/2N_\omega$ to generate field needed to compute the next scattering event.

$$L_v(\hat{u_{s_i}})(x_{pq},\omega_j) = -\frac{\omega_j^2}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\frac{e^{-i\frac{\omega_j\|x_{pq}-z_{nm}\|}{c_0}}v(z_{nm})\hat{u_{s_i}}(z_{nm},\omega_j)}{4\pi\|x_{pq}-z_{mn}\|} \tag{3.11}$$

   where we do not add the term if $x_{kl} = z_{nm}$. This is very expensive to do since there are N^4 numbers to evaluate in the worst case. If $v(x_{pq}) = 0$, then we do not compute $L_v(\hat{u_{s_i}})(x_{pq},\omega_j)$ since it will not be needed in the sum for the next step.

To generate the data with K scattering events we need to generate the field along the flight path with k scattering we use Horner's method. The field with K scattering events $\hat{u}_s^K(x,\omega)$ can be found recursively

$$\hat{u}_s^K(x,\omega) = \hat{u}_s^{in}(x,\omega) + L_v(\hat{u}_s^{K-1})(x,\omega). \tag{3.12}$$

Inside of the operator we need $\hat{u}_s^{K-1}(x,\omega)$ for $x \in \operatorname{supp}(v)$. The data can be computed bottom up for $k = 0, 1, \dots K-1$:

1. Compute $\hat{u}_{s_i}^k(x_{pq},\omega_j)$ for $x_{pq} \in \operatorname{supp}(v)$. When $k = 0$, this is the incident field. Otherwise we need to have $\hat{u}_{s_i}^{k-1}(x_{pq},\omega_j)$ from the previous step.

2. Compute $\hat{u}_{s_i}^k(\gamma(s_i),\omega_j)$ and add it to the data.

After this we only need to compute the last scattering event $\hat{u}_{s_i}^K(\gamma(s_i),\omega_j) = \hat{u}_s^{in}(\gamma(s_i),\omega_j) + L_v(\hat{u}_s^{K-1})(\gamma(s_i),\omega_j)$ at the antenna and add it to the data. Since we plan on removing the incident field from the data, we may as well add it in this last step. Thus, we need only add $L_v(\hat{u}_s^{K-1})(\gamma(s_i),\omega_j)$ to the data.

3.2 Image Recovery

We will assume that we have the data measured at uniform samples as in the previous section. The integrals are approximated with a Riemann sum as

$$v(x) = -\frac{32B\pi r^2}{N_s N_\omega c_0^2} \sum_{l=1}^{N_s} \sum_{j=1}^{N_\omega} e^{-i\omega_j\left(\frac{2(x\cdot\hat{\gamma}(s_l)-r)}{c_0}\right)} D(s_l,\omega_j) \tag{3.13}$$

where N_s and N_ω are the number of data samples. We recover v on uniform samples located at $(m/N + 1/2N, n/N + 1/2N)$ for $m,n \in \{0,1,2,\dots,N-1\}$. That is, we are recovering $v(x_{ij})$ where $x_{mn} = (m/N + 1/2N, n/N + 1/2N)$.

To perform simulation choose a flight path that is circular, and centered at $(h,k) = (1/2,1/2)$ with radius $r = 10$. Then

$$v(x_{mn}) = -\frac{3200B\pi}{N_s N_\omega c_0^2} \sum_{l=1}^{N_s} \sum_{j=1}^{N_\omega} e^{-i\omega_j\left(\frac{2(x_{mn}\cdot\hat{\gamma}(s_l)-10)}{c_0}\right)} D(s_l,\omega_j) \tag{3.14}$$

where $q_j = j/N_\omega + 1/2N_\omega$. $\omega_j = \omega_0 - B/2 + (q+1/2)B/N$

This is what we call the direct method. Since we want to use the butterfly algorithm it will be useful to have this in a normalized form as well.

$$v(x_{mn}) = -\frac{3200B\pi}{N_s N_\omega c_0^2} \sum_{l=1}^{N_s} \sum_{j=1}^{N_\omega} e^{-i\omega_j(\frac{2(x_{mn}\cdot\hat{\gamma}(s_l)-10)}{c_0})} D(s_l, \omega_j) \tag{3.15}$$

3.3 Summary

To recap, the computational problems we need to are:

1. Evaluate field in target support,
2. Evaluate Field along the flight path, and
3. Reconstruct a target estimate.

These sums can be evaluated directly as they are but very slowly. In the recover problem if the number of samples in dimension are equal to N, then we would need to sum N^2 points for each of the N^2 points we want to evaluate at yielding $O(N^4)$ steps which is pretty terrible for even small N(if $N = 10^3$, then $N^4 = 10^{12}$, which would take about 15 minutes on a standard laptop). If we are willing to allow a certain amount of approximate error, then we significantly reduce this time via the butterfly algorithm. In the data generation problem, by using only the non zero points, we will gain significant speed if the target support is sparse. Let us finally describe the butterfly algorithm.

CHAPTER IV

LOW RANK APPROXIMATIONS AND THE BUTTERFLY ALGORITHM

In this section we will discuss the ideas behind the algorithm used to efficiently numerically compute the oscillatory integrals that have shown up. We call this the butterfly algorithm for short. Numerical integration is a type of matrix product(tensor products in higher dimensions). The butterfly algorithm is an algorithm that exploits multi scale pairing of regions to construct low rank approximations each row of the matrix product in an efficient manner. There are many variants of these algorithms [41]. Hence in the literature there are butterfly algorithms. Since we will only be working with the particular butterfly algorithm developed in [11]. In the first section we will discuss low rank separated approximations. In the second we will define the butterfly algorithm in the abstract. In the third we will define the matrix product we are evaluating and in the fourth section we will give the butterfly algorithm for FIOs that we will use.

4.1 Low Rank Approximations

First some motivation. In any linear problem, we must be able to perform matrix operations efficiently. Consider multiplying an $N \times N$ matrix A with an N dimensional vector x. How can we do this fast? For a general matrix, computing the action on a vector takes $O(N^2)$ steps since we must multiply each row of A with x. If A is diagonal we only need to perform $O(n)$ multiplications, but most matrices are not diagonal. If the matrix is low rank with rank r, then it can be factored into a product $Ax = BCx = B(Cx)$ where B is $n \times r$ and Cx is $r \times 1$. This means that the product Ax can be found in terms of a low rank matrix times a low rank vector. This means that we can perform a matrix multiplication in $2rn$ steps by multiplying Cx first to get a r vector, then $B(Cx)$ to get an n vector. A being rank r means that there is a $\delta \in \mathscr{R}^r$, and a $n \times r$ matrix A' such that $Ax = A'\delta$. The action of our matrix on a vector can be approximated by the action of a low rank matrix acting on a low rank vector. The elements in the vector δ are called equivalent weights. Finding the matrix and vector is not trivial but it can be done if we exploit analytic properties of the entires of the matrix.

This would be nice, but most matrices that are not full rank are not necessarily low rank. The next relaxation we can consider is that A be approximately low rank. By this we mean for any $\varepsilon > 0$ there exist a matrix A' with rank r_{eps} such that $\|A - A'\| < \varepsilon$ where r_ε is much less than N. The rank depends on the amount of error we are willing to allow in our approximation. Even this is often too much to ask for. Generally there does not exist a low rank approximation for an entire matrix. The point is that global approximation is often quite difficult. Since global does not work local is the next step which immediately requires analysis. Consider any sub matrix contained in A. If it is low rank, then we may perform block matrix multiplication for a more efficient multiplication. It is difficult to see if a general matrix large has low rank sub matrices. This requires analysis of where the matrix entries come from. In the hierarchal matrix scheme [1], one store matrices in a hierarchy of factored low rank matrices. This method is concerned with formating a matrix so that we can do efficient matrix algebra. The matrices that are done for are typically the result of elliptic PDEs. Once the hierarchical structure is built one can perform several matrix vectors efficiently. We will not make use of hierarchical matrices, but they are closely related to the butterfly algorithm. The butterfly algorithm is concerned with evaluating the matrix on a given vector.

Consider evaluating the integral operator $\int_0^1 k(x,y) f(y)\, dy$ numerically for several values of x. This is a fairy difficult problem since we would need to perform this integral for every x. If we use a Riemann sum to approximate the integral this is the same as the matrix problem above. If $k(x,y) = \sin(x+y)$, then we know that $k(x,y) = \sin(x)\cos(y) + \cos(x)\sin(y)$, so that $\int_0^1 k(x,y) f(y)\, dy = \sin(x)\int_0^1 \cos(y) f(y)\, dy + \cos(x)\int_0^1 \sin(y)\, dy$. That is, the kernel k can be decomposed as a sum of products of single variable functions separating the integration. We say that k has a separated representation. This greatly speeds things up. Now we only need to compute two integrals, store those numbers and for different values of x evaluate sin and cos. The key feature of this kernel is that it has a low rank(i.e the number of terms in the sum is small) separated representation allowing us to evaluate the integral very efficiently. The representation corresponds to a matrix factorization low rank matrix(tensor) factorization. In general it is not possible to represent most functions in this manner, allowing a small amount of controlled error widens the space of functions that have such approximations. Having one approximation that is valid for all x and y is still rare. Restricting the domain of the kernel to a suitably small region often guarantees the existence of a low rank separated approximation. Such a restriction is called an admissibility condition. A pair of sets whose product satisfies the admissibility condition is said to be an admissible pair. This means we can perform the partial integral over the range of the y variable. The admissibility condition is exploited by the butterfly algorithm by pairing subsets of X and Y whose product satisfies the admissibility condition evaluate the integral.

4.2 The Butterfly Algorithm

The butterfly algorithm is a method for evaluating

$$F(x_m) = \sum_{n=1}^{N} k(x_m, y_n) f(y_n) \tag{4.1}$$

for all $m = 1, \ldots N$ where $x_m = m/N$ and $y_n = n/N$(approximate integral over $[0,1]$). Consider two partitions X_l of $[0,1]$ with cell size $\frac{1}{2^l}$ and Y_l with cell size $\frac{1}{2^{L-l}}$ for some L and $l = 0, \ldots, L$. The partition is a set of equivalent classes of points in $[0,1]$ where points are identified if they are in the same box. Each class is called a cell. The scale of the partition refers to the number of cells it has. These partitions are on opposite scale since they are inversely proportional. The first partition has 2^l cells, and the second 2^{L-l} cells. The number of pairs of cells from the two partitions is 2^L which is independent of the scale l. Let $A \in X_l$ and $B \in Y_l$. Then the product of A and B has area $1/2^L$ independent of l. Moreover if L is sufficiently large, the area becomes an admissibility condition for certain kernels. For each pairing at each scale, there exist a low rank separated approximation

$$k(x,y) \approx \sum_{t=1}^{q} \alpha_t^{AB}(x) \beta_t^{AB}(y) \tag{4.2}$$

for all $x \in A$ and $y \in B$. This is the natural extension of separation of variables, and is closely related to tensor decompositions [21]. Denote by $F_B(x)$ the sum restricted to points that are in B. Then

$$F_B(x_m) \approx \sum_{y \in B} \sum_{t=1}^{q} \alpha_t^{AB}(x) \beta_t^{AB}(y) f(y_n) = \sum_{t=1}^{q} \alpha_t^{AB}(x) \delta_t^{AB} \tag{4.3}$$

for all $x_m \in A$ where $\delta_t^{AB} = \sum\limits_{y \in B} \beta_t^{AB}(y) f(y_n)$. That is, for any pairing, the restricted integral becomes a low rank matrix multiplying a low rank vector. So choosing L large enough guarantees that existence of an equivalent weight and matrix. When $l = L$, $Y_L = [0,1]$, so for each pair $F_B(x) = F(x)$ for all $x \in A$(i.e the sum is over all of Y). There exist a low rank separated for the integral for each of these pairs at this scale but the weights are not easy to construct. On the other hand, when $l = 0$ we have for all $x \in [0,1] \in X_0$ that $\delta_t^{AB} = \sum\limits_{y \in B} \beta_t^{AB}(y) f(y_n)$. Since B is very fine, the sum will be very cheap to compute. At scale $l = 1, 2, \dots L-1$, each $A \in X_l$ has a parent $A_p \in X_{l-1}$ and each $B \in Y_l$ has children $B_c \in Y_{l-1}$. Each parent child pair satisfies the admissibility condition so

$$F_{B_c}(x_m) \approx \sum_{t_c}^{q} \alpha_{t_c}^{A_p B_c}(x_m) \delta_{t_c}^{A_p B_c} \tag{4.4}$$

for all $x_m \in A \subset A_p$. We also know that the partial sum over a cell B is equal to the sum of the partial sums over its children

$$F_B(x_m) = \sum_{c=1}^{2} F_{B_c}(x_m) \approx \sum_{c=1}^{2} \sum_{t_c}^{q} \alpha_{t_c}^{A_p B_c}(x_m) \delta_{t_c}^{A_p B_c}. \tag{4.5}$$

Then we have the approximate equation

$$\sum_{t=1}^{q} \alpha_t^{AB}(x_m) \delta_t^{AB} \approx \sum_{c=1}^{2} \sum_{t_c}^{q} \alpha_{t_c}^{A_p B_c}(x_m) \delta_{t_c}^{A_p B_c}. \tag{4.6}$$

This is an over determined linear system for δ_t^{AB} in terms of $\delta_{t_c}^{A_p B_c}$. The butterfly algorithm is the process of computing the weights for X_L and Y_L bottom up starting from X_0, Y_0. The butterfly algorithm requires that the separation rank is constant for different pairs. It also requires a depth L. The algorithm is then to compute the weights, and matrices $\alpha_t^{AB}(x_m)$ for all $A \in X_L$ and $B \in Y_L$ bottom up by finding approximate solutions to 4.6 and using the definition for $l = 0$. For the FIO case the matrices $\alpha_t^{AB}(x_m)$ will be constructed via Chebyshev interpolation in either x or y depending on if $l < L/2$, $l = L/2$, or $l > L/2$so will be known. This is because the low rank separation can be shown to be in monomials of the respective variable, and the Chebyshev interpolant is almost proportional to the best approximating monomial. The admissibility condition for the FIO case is that $dim(A)dim(B) < \frac{1}{N}$ This is for integration over one dimension. The same idea works in higher dimensions. The work we are following focuses on 2-d due to its ease of visualization and practicality. This is exactly the method we need.

To summarize, an admissibility condition on the size of the support can be used with a sequence of partitions X_l, Y_l such that the product of any pair is admissible, to find equivalent weights bottom up for the entire sum starting by computing the equivalent weights by definition for $l = 0$. These final pairs correspond to evaluating the entire sum and we can now perform this very quickly since we have the weights. We have described the algorithm for integrating over one dimension. This is easily extendible to any dimension d. In this case each cell has 2^d children. We will work with 2 dimensions for the FIO algorithm.

4.3 Butterfly Algorithm For Fourier Integral Operators

In this section we give the butterfly algorithm from [11]. This algorithm was developed in [2] as well. We define a discrete Fourier integral Operator(DFIO) to be

$$F(x_{kl}) = \sum_{m=1}^{N} \sum_{n=1}^{N} a(x_{kl}, y_{mn}) e^{i\phi(x_{kl}, y_{mn})} f(y_{mn}) \tag{4.7}$$

where $x_{kl} = (k/N - 1/2N, j/N - 1/2N)$ and $y_{mn} = (m/N - 1/2N, n/N - 1/2N)$ where $k, l = 1, \ldots N$. This is an integral over and evaluated on the unit square. As we saw in the previous section the goal is to reconstruct equivalent weights to approximate the integral. To do this we need the partitions X_l and Y_l as before. The unit square is naturally partitioned. For example, with $[0, 1]$ partitioned into $[0, 1/2)$ and $[1/2, 1]$, then we know which points are in either set by looking at the first bit after the decimal point in the binary representation of the number. The number $1/4$ is given by $(.01)_2$ in binary, so it belongs to the first class. since its first bit after the decimal is 0. On the other hand

$1/2 = (.1)_2$ in binary so it belongs to the second class since its bit is 1. This is also true for the unit box since we can look at each numbers bits after the decimal. At level l, the point x belongs to the box $A \in X_l$ if the first l bits of its first and second component correspond to A. In the algorithm itself we will need to know the center and dimensions of each cell in a given partition. This can be found analytically. This means we do not actually have construct any partition since we can figure out the parameters of the cells directly, and know membership. In the paper by Demanet the first step in the algorithm is to construct two quad trees(the sets of partitions). This is a booking tool and is not necessary since points in $[0,1]^2$ are naturally partitioned. We will use this natural partitioning.

Recall that the one dimensional Chebyshev grid adapted to $[-1/2, 1/2]$ is given by

$$\{z_j = \frac{1}{2}\cos(j\pi/(q-1)) | 0 \leq j \leq q-1\}. \tag{4.8}$$

This can be adapted to any interval by appropriate scaling and shifting. The Lagrange basis polynomial for a given grid are given by

$$L_j(z) = \prod_{0 \leq k \leq q-1, k \neq j} \frac{z - z_k}{z_j - z_k}. \tag{4.9}$$

The 2-d Chebyshev grid is defined by taking tensor products of the 1-d grids $\{(z_{t_1}, z_{t_2})\}$. The Chebyshev basis functions are

$$L_t(z) = L_{t_1}(z_1) L_{t_2}(z_2) \tag{4.10}$$

where $z = (z_1, z_2)$ and $t = (t_1, t_2)$. The Chebyshev grid adapted for box B is denoted $\{y_t^B\}$. The basis polynomials for these grids are denoted $\{L_t^B\}$. Let $x_0(A)$ be the center of A, and $y_0(B)$ be the center of B. We need to choose a number of partitions L to use so that the admissibility condition is satisfied. Choose $L \geq 2\log_2(N)$ since we want the size of the finest boxes to be at most $\frac{1}{N} \times \frac{1}{N}$. Finally we need to choose the number q of Chebyshev points to use. This is not so trivial on what q should be. One thing we can do is run the algorithm several times with q being squared each time to see if we get noticeable improvements. We will just choose q to be some reasonably large number. The algorithm is then:

1. For $l = 0$, initialize

$$\delta_t^{AB} = e^{-i\phi(x_0(A), y_t^B)} \sum_{y \in B} L_t^B(y) e^{i\phi(x_0(A), y)} f(y) \tag{4.11}$$

   for all $A \in X_0$, $B \in Y_0$, and $t = 1, \dots q$. There are qN^2 weights to compute.

2. For $l = 1, \dots, L/2$ compute the weights for all $A \in X_l$ and $B \in Y_l$ by

$$\delta_t^{AB} = e^{-i\phi(x_0(A), y_t^B)} \sum_c \sum_{t_c} L_t^B(y_{t_c}^{B_c}) e^{i\phi(x_0(A), y_{t_c}^{B_c})} \delta_{t_c}^{A_p B_c} \tag{4.12}$$

3. Switch representation. Compute weights by

$$\delta_t^{AB} = \sum_s a(x_t^A, y_s^B) e^{i\phi(x_t^A, y_s^B)} \delta_s^{AB}. \tag{4.13}$$

   Note that $l = L/2$ is seen twice in this algorithm.

4. For $l = L/2 + 1, \dots L$, compute weights by

$$\delta_t^{AB} = \sum_c e^{i\phi(x_t^A, y_0(B_c))} \sum_{t_p} L_{t_p}^{A_p}(x_t^A) e^{-i\phi(x_{t_p}^{A_p}, y_0(B_c))} \delta_{t_p}^{A_p B_c} \tag{4.14}$$

5. Compute the sum using the equivalent weights from last step

$$F(x_{kl}) = e^{i\phi(x_{kl}, y_0(B))} \sum_t L_t^A(x_{kl}) e^{-i\phi(x_t^A, y_0(B))} \delta_t^{AB} \tag{4.15}$$

4.4 Imaging With The Butterfly Algorithm

Suppose that $N_s = N_w = N$. The imaging algorithm given data $d(i, j)$ on $\gamma(s_j)$ is

$$v(x_{mn}) = -\frac{3200B\pi}{N^2 c_0^2} \sum_{l=1}^{N} \sum_{j=1}^{N} \frac{e^{-i(\omega_0 - \frac{B}{2} + q_j B)(\frac{2(x_{mn} \cdot \hat{\gamma}(s_l) - 10)}{c_0})} \overline{\hat{p}}(\omega_0 - \frac{B}{2} + q_j B) d(s_l, \omega_0 - \frac{B}{2} + q_j B)}{(\omega_0 - \frac{B}{2} + q_j B)|\hat{p}(\omega_0 - \frac{B}{2} + q_j B)|^2} \tag{4.16}$$

for all $x_{mn} = (m/N + 1/2N, n/N + 1/2N)$, $n, m = 0, 1, \dots N-1$. Setting $\phi(x_{mn}, s_l, q_j) = -(\omega_0 - \frac{B}{2} + q_j B)(\frac{2(x_{mn} \cdot \hat{\gamma}(s_l) - 10)}{c_0})$, $a(x_{mn}, s_l, q_j) = 1$, and $f(s_l, q_j) = \frac{\overline{\hat{p}}(\omega_0 - \frac{B}{2} + q_j B) d(s_l, \omega_0 - \frac{B}{2} + q_j B)}{(\omega_0 - \frac{B}{2} + q_j B)|\hat{p}(\omega_0 - \frac{B}{2} + q_j B)|^2}$ then the problem is exactly in the form of the butterfly algorithm. We will ignore the amplitude for now since it is constant. The algorithm is then

1. For $l = 0$, initialize

$$\delta_t^{AB} = e^{i\phi(x_0(A),y_t^B)} \sum_{y \in B} L_t^B(y) e^{i\phi(x_0(A),y)} f(y) \tag{4.17}$$

   for all $A \in X_0$, $B \in Y_0$, and $t = 1, \dots q$. There are q^2N^2 weights to compute.

2. For $l = 1, \dots, L/2$ compute the weights for all $A \in X_l$ and $B \in Y_l$ by

$$\delta_t^{AB} = e^{i\phi(x_0(A),y_t^B)} \sum_c \sum_{t_c} L_t^B(y_{t_c}^{B_c}) e^{i\phi(x_0(A),y_{t_c}^{B_c})} \delta_{t_c}^{A_pB_c} \tag{4.18}$$

3. Switch representation. Compute weights by

$$\delta_t^{AB} = \sum_s e^{i\phi(x_t^A,y_s^B)} \delta_s^{AB}. \tag{4.19}$$

   Note that $l = L/2$ is seen twice in this algorithm.

4. For $l = L/2+1, \dots L$, compute weights by

$$\delta_t^{AB} = \sum_c e^{i\phi(x_t^A,y_0(B_c))} \sum_{t_p} L_{t_p}^{A_p}(x_t^A) e^{-i\phi(x_{t_p}^{A_p},y_0(B_c))} \delta_{t_p}^{A_pB_c} \tag{4.20}$$

5. Compute the sum using the equivalent weights from last step

$$v(x_{mn}) = e^{i\phi(x_{kl},y_0(B))} \sum_t L_t^A(x_{kl}) e^{-i\phi(x_t^A,y_0(B))} \delta_t^{AB} \tag{4.21}$$

4.5 Conclusions

It is worth pointing out a few different methods for similar problems. The problem of efficiently storing and using matrices has applications in nearly all facets of applied mathematics. One method is to recursively break a matrix down into sub matrices until each sub matrix is approximately low rank and store the factored form of each low rank matrix. This matrix representation is called a Hierarchal matrix [1]. Similar to the butterfly algorithm, the existence of a low rank approximation is guaranteed by an admissibility condition and depends on the product of the dimensions of the sub matrix. The focus of the butterfly algorithm is for a single matrix vector multiply, while, once we have formated the matrix hierarchically we can perform several matrix vector multiplies very fast. The C++ simulation implementing the butterfly algorithm developed in [11] is in the appendix.

Chapter V

Simulation, Current Methods For Mitigating Multiple Scattering, and Conclusions

In this chapter we will run several simulations. First we will directly evaluate the integral defining the data to simulate single scattered data due to synthetic targets. Then we will use the image recovery formula to recover the test image using the direct formula. This is a test of the recovery method. It will be slow. The point is that we can use the butterfly algorithm to speed things up significantly if we allow controlled error. We will then generate data using the direct sum, and evaluate the imaging method using the butterfly algorithm. Finally we will use the target estimate to estimate multiple scattering effects in data. That is, we will compute the multiply scattered field from an image estimate, and compare its magnitude to that of the data.

5.1 Simulation of Multiple Scattering and Its Effects on Imaging

We simulate the iterated scattering series with various target setups. The data may be difficult to interpret so we only plot the original target, followed by the reconstructed target using the direct method. The point here is to see degradation of images using more scattering data. The size of the images is limited by the image reconstruction algorithm.

The butterfly algorithm is not directly applicable to data generation since we need to compute the scattering operator along the flight path and in the image support. It may be possible to use it.

5.2 Current Methods For Compensating for Multiple Scattering

Mitigating multiple scattering effects is still an active area of research. In [14], the author avoids the use of the Born approximation in compressive sensing MIMO RADAR. This is based on

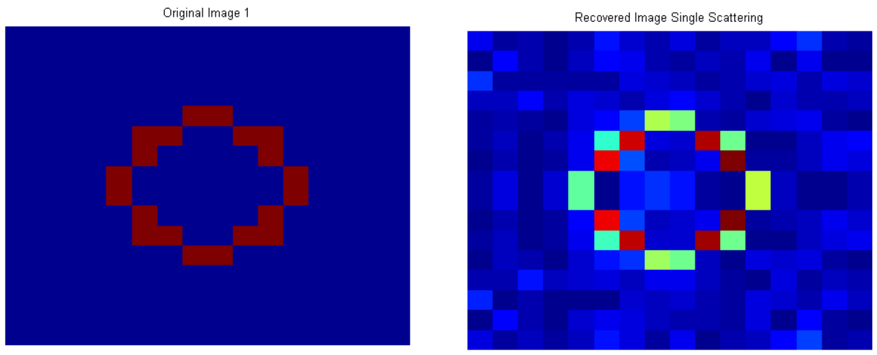

Figure 5.1: Reconstruction using single scattered data.

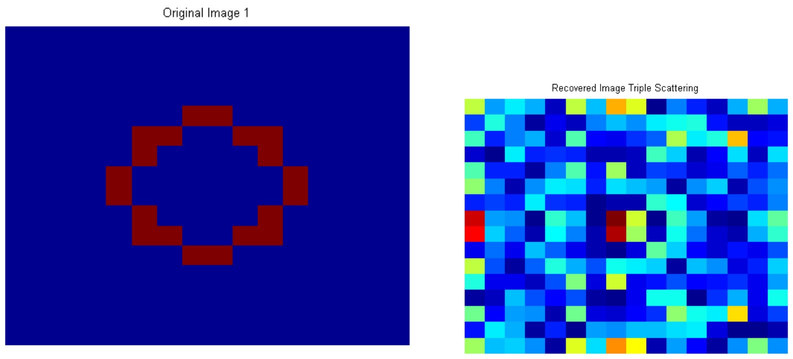

Figure 5.2: Multiple scattering effects on linear image formation.

point like targets. There a few common ways to handle multiple scattering effects in imaging. One may pose the imaging problem as a non linear optimization problem. That is, we use our model for the data and find a function that minimizes the error between the data collected, and the data that the function generates. This slow. Another way is to treat multiple scattering components as coherent noise and try to filter it out of the data. A completely different approach which is still an active research area is to look for less direct information about targets. The linear sampling method [8] seeks the support of the targets. The method works, but requires a small scene so that we may assume that the wave front of the incident field is planar and a large amount of data. In the engineering community there has been interest in model based imaging. This relies on know what we are trying to see, so if we can keep a catalog of targets we are interested and try to fit them in the data. There are other approximations used to find a linear data model such as the Kirchoff approximation, or the paraxial approximation. These still ignore multiple scattering effects.

5.3 Conclusions And Summary

We discussed the wave model for synthetic aperture imaging from the continuous to the digital implementation. We have discussed multiple scattering in the context of the wave equation. Recent work in computational harmonic analysis has made it possible to efficiently evaluate a large class of oscillatory integrals. These are applicable to the imaging problem. The multiple scattering effects can be interpreted as the powers of the scattering operator so that the field is a geometric series in the scattering operator. To speed things up we perform a few different factorizations of this series for an appropriate number of scattering events. One is based on Horner's method, and the other on a product(composition) of binomials which contain powers of the scattering operator. Horner's method allows us to apply the butterfly algorithm repeatedly to compute k scattering events in k compositions. On the other hand, if we represent the scattering operator by a square matrix, we can compute $2^k - 1$ scattering events can be computed using k^2 matrix multiplications. We used the butterfly algorithm in image formation to test the accuracy of estimating multiple scattering effects in data by forming an estimate of the target and using it in the data model. The multiple scattering problem is still very much open. New ideas are needed to attack it. Multi scale analysis is a powerful tool for reducing the rank of linear operators for restricted regions. It has applications in imaging and data creation. Synthetic Aperture Radar is a powerful technology that has been developed over the last 50 years. It is ripe with interesting mathematical problems and sees application of very recent mathematics. This is a bridge between abstract mathematics and pure application and an exciting field to study. There is still much work to be done in this field. The model we have used was for very simple materials. In this setting the imaging problem is described by a polynomial operator in v. Linear operators have a very powerful theory behind them making the linearized schemes much easier to study mathematically. Multiple scattering effects are inherently nonlinear in nature and make this problem very difficult and deep.

BIBLIOGRAPHY

[1] S. BÖRM, L. GRASEDYCK, AND W. HACKBUSCH, *Hierarchical matrices*, Lecture notes, 21 (2003), p. 2003.

[2] E. J. CANDÈS, L. DEMANET, AND L. YING, *A fast butterfly algorithm for the computation of fourier integral operators*, Multiscale Modeling & Simulation, 7 (2009), pp. 1727–1750.

[3] E. J. CANDES AND T. TAO, *Near-optimal signal recovery from random projections: Universal encoding strategies?*, Information Theory, IEEE Transactions on, 52 (2006), pp. 5406–5425.

[4] Y. CAO, J. F. LOPEZ, A. MARTINEZ, AND Z. QIAO, *A mathematical model for mimo imaging*, (2012), pp. 839308–839308–11.

[5] G. L. CHARVAT, A. J. FENN, AND B. T. PERRY, *The mit iap radar course: build a small radar system capable of sensing range, doppler, and synthetic aperture (sar) imaging*, in Radar Conference (RADAR), 2012 IEEE, IEEE, 2012, pp. 0138–0144.

[6] M. CHENEY, *A mathematical tutorial on synthetic aperture radar*, SIAM review, 43 (2001), pp. 301–312.

[7] M. CHENEY AND B. BORDEN, *Fundamentals of radar imaging*, CBMS-NSF regional conference series in applied mathematics, Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009.

[8] D. COLTON, H. HADDAR, AND M. PIANA, *The linear sampling method in inverse electromagnetic scattering theory*, Inverse problems, 19 (2003), p. S105.

[9] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, Applied Mathematical Sciences, Springer, 1998.

[10] D. F. DeLong, *Multiple signal direction finding with thinned linear arrays*, tech. report, DTIC Document, 1983.

[11] L. Demanet, M. Ferrara, N. Maxwell, J. Poulson, and L. Ying, *A butterfly algorithm for synthetic aperture radar imaging*, SIAM Journal on Imaging Sciences, 5 (2012), pp. 203–243.

[12] A. W. Doerry, *Wavefront curvature limitations and compensation to polar format processing for synthetic aperture radar images*, United States. Department of Energy, 2006.

[13] C. L. Epstein, *Introduction to the mathematics of medical imaging*, Siam, 2008.

[14] A. C. Fannjiang, *Compressive inverse scattering I. High frequency SIMO/MISO and MIMO measurements*.

[15] R. Fleury and A. Alù, *Cloaking and invisibility: A review*, in Forum for Electromagnetic Research Methods and Application Technologies (FERMAT), vol. 1, 2014, pp. 1–24.

[16] G. Garza and Z. Qiao, *Resolution analysis of bistatic sar*, (2011), pp. 80211V–80211V–6.

[17] M. Ghasr, M. Abou-Khousa, S. Kharkovsky, R. Zoughi, and D. Pommerenke, *Portable real-time microwave camera at 24 ghz*, Antennas and Propagation, IEEE Transactions on, 60 (2012), pp. 1114–1125.

[18] D. J. Griffiths and R. College, *Introduction to electrodynamics*, vol. 3, prentice Hall Upper Saddle River, NJ, 1999.

[19] W. Heisenberg and H. Euler, *Consequences of dirac theory of the positron*, arXiv preprint physics/0605038, (2006).

[20] G. KAISER, *A friendly guide to wavelets*, Modern Birkhèauser classics, Birkhäuser Boston, 2011.

[21] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM review, 51 (2009), pp. 455–500.

[22] J. A. KONG, *Theory of electromagnetic waves*, New York, Wiley-Interscience, 1975. 348 p., 1 (1975).

[23] P. KUCHMENT, *The Radon Transform and Medical Imaging*, vol. 85, SIAM, 2014.

[24] J. LOPEZ AND Z. QIAO, *Array geometries, signal type, and sampling conditions for the application of compressed sensing in mimo radar*, in SPIE Defense, Security, and Sensing, International Society for Optics and Photonics, 2013, pp. 871702–871702.

[25] ——, *Array geometries, signal type, and sampling conditions for the application of compressed sensing in mimo radar*, in SPIE Defense, Security, and Sensing, International Society for Optics and Photonics, 2013, pp. 871702–871702.

[26] J. X. LOPEZ AND Z. QIAO, *Filtered back projection inversion of turntable isar data*, (2011), pp. 805109–805109–9.

[27] A. MARTINEZ AND Z. QIAO, *Iteratively compensating for multiple scattering in sar imaging*, in SPIE Defense, Security, and Sensing, International Society for Optics and Photonics, 2013, pp. 874603–874603.

[28] F. NATTERER, *Regularization techniques in medical imaging*, in Mathematics and Computer Science in Medical imaging, Springer, 1988, pp. 127–141.

[29] N. PENA, G. GARZA, AND Z. QIAO, *Filtered back projection type direct edge detection of real synthetic aperture radar images*, (2012), pp. 83940N–83940N–7.

[30] J. POULSON, L. DEMANET, N. MAXWELL, AND L. YING, *A parallel butterfly algorithm*, arXiv preprint arXiv:1305.4650, (2013).

[31] W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND W. T. VETTERLING, *Numerical Recipes: the art of scientific computing*, Cambridge Univ. Press, New York, 1986.

[32] T. RAY, Y. CAO, Z. QIAO, AND G. CHEN, *2d and 3d isar image reconstruction through filtered back projection*, (2012), pp. 836107–836107–11.

[33] A. REIGBER AND A. MOREIRA, *First demonstration of airborne sar tomography using multibaseline l-band data*, Geoscience and Remote Sensing, IEEE Transactions on, 38 (2000), pp. 2142–2152.

[34] M. I. SKOLNIK, *Radar handbook*, (1970).

[35] M. L. SKOLNIK, *Introduction to Radar Systems*, McGraw-Hill Education (India) Pvt Limited, 2003.

[36] M. SOUMEKH, *Synthetic Aperture Radar Signal Processing with MATLAB Algorithms*, Wiley-Interscience publication, Wiley, 1999.

[37] J. A. STRATTON, *Electromagnetic theory*, vol. 33, John Wiley & Sons, 2007.

[38] N. SUZUKI, *On the convergence of neumann series in banach space*, Mathematische Annalen, 220 (1976), pp. 143–146.

[39] L. WANG AND B. YAZICI, *Ground moving target imaging using ultranarrowband continuous wave synthetic aperture radar*, (2013).

[40] C. E. YARMAN, B. YAZICI, AND M. CHENEY, *Bistatic synthetic aperture hitchhiker imaging*, in Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on, vol. 1, IEEE, 2007, pp. I–537.

[41] L. YING, *Sparse fourier transform via butterfly algorithm*, SIAM Journal on Scientific Computing, 31 (2009), pp. 1678–1694.

APPENDIX A

APPENDIX A

BUTTERFLY ALGORITHM IMPLEMENTATION

1.1 C++ Imaging Code

```
#include "input.h"

//for 1d grid

inline int Grid_Index(int Box,int t, int l){

return ((int) pow(2.0,((double) l)))*q+Box*q+t;

}



inline point Grid_Point(double * grids, int Box, int t1, int t2, int l){

int B1=Box/N;

int B2=Box%N;

        return point(grids[Grid_Index(B1,t1,l)] ,grids[Grid_Index(B2,t2,l)]);


}


//give the chebyshev grid for [0,1] with q points, sorted from left to right.

inline void Proto_Grid(double *fillme){

for(int t=0;t<q;t++){


fillme[t]=.5*cos((q-1-t)*pi/(q-1))+.5;

// cout<<"fill "<<fillme[t]<<endl;

}

}


//fill in the array with the chebyshev grid adapted to box  at level l.

//Assumes that protogrid is already allocated.  b runs from 0 to 2^level-1

inline void Grid(double *grid, double *proto_grid, int Box, int l){

double scale=pow(2,(double)-l);
```

```
for(int t=0;t<q;t++){
//shift protogrid by scale, to  grid from 0 to b.  Then shift by b*scale for the
//corresponding box.
grid[t]=scale*(proto_grid[t]+Box);
}
}

//Fills grids with all the 1d grids for [0,1] we will need. There are 2N-1 grids, and each has q entries.
//The function Grid_Index returns the appropriate index.  A tensor grid is used for higher dimensions
inline void Build_Grids(double* &grids){
grids= new double[(2*N-1)*q];
//make the first q entries the chebyshev grid adapted to [0,1].
double * current_grid=grids+q;
Proto_Grid(grids);
//go through each scale
for( int l=1;l<=L;l++){
//go through each offset at that scale.
for(int Box=0;Box<(int)(pow(2.0,l));Box++){
Grid(current_grid,grids,Box,l);
current_grid=grids+q;
}
}
}

//The t-th Lagrange basis polynomial for the grid evaluated at the point x
inline double L1d(double *grids, int l, int Box, int t,double x){
double product=1;
int grid=Grid_Index(Box,q,l);
cout<<"T!!"<<t;
for(int j=0;(j<q)&&(j!=t);j++){
product*=(x-grids[grid+t])/(grids[grid+t]-grids[grid+j]);
}
return product;
}

//Tensor interpolation L_t1(y.x)*L_t2(y.y).
//t goes up to q^2. B goes to N^2, so that t1, t2 are between 0, and q-1,
//and B1, and B2 are between 0, and M-1
inline double L2d(double *grids, int t1, int t2, int Box, int l,point y){
        int B1=Box/N;
int B2=Box%M;
```

```
        return L1d(grids,l,B1,t1,y.x)*L1d(grids,l,B2,t2,y.y);
}

inline void Swap(complex<double>* &current, complex<double>* &previous){
complex<double>* temp;
temp=previous;
previous=current;
current=temp;
}

inline void Zero(complex<double> * weights ){
for(int n=0;n<M*r_eps;n++)weights[n]=0;
}

//return the center of the box from the left at level l.
inline double Center1d(int l, int box){
double scale=pow(2,(double)-l);
return (int)(scale*((double)box+0.5));
}

inline point Center(int l, int box){
int b1=box/N;
int b2=box%N;
return point(Center1d(l,b1),Center1d(l,b2));
}

//We need to enumerate the boxes, and pairings we are working with.
//Consistency is key.  In X_l there are 4^l cells.  In Y_l there are
//4^{L-l}.  The number of pairs is 4^L=N^2 for any level.
//Each pair (A,B) at a scale corresponds to a number n=0,...N^2-1.
//A=0,...,4^l-1, and B=0,...,4^{L-l}-1.  A simple map
//is to take the first 2l bits of n to be A, and the last 2(L-l)
//bits to be B n->(n/4^l-1,4n^l)=(A,B).
//We always count from left to right, bottom up, so that we would start around (0,0), and end around
//(1,1)
inline int A(int n, int l){
return (n/(int)pow(2.0,2*(L-l)));
}

//shift n to the left, so that only the last 2*(L-l) digits are non Zero,
//then shift back so that the first 2l bits are Zero.
inline int B(int n, int l){
return n%(int)(pow(2.0,2*(L-l)));
}

//Parent of A is just A with its last two bits dropped
inline int Parent(int A){
return A>>2;
}

//Children of B are numbers where we append two bits
inline int Child(int B,int c){
    return (B<<2)+c;
}
//There are N*N*r_eps weights to index.  Here, B needs to have at most 2l bits
inline int Weight_Index(int A, int B, int l, int t1, int t2){
return ((A<<(2*(L-l)))+B)*r_eps+q*t1+t2;
}

///Find the box that p is in  at level l. l=0 means only one cell.
//The unit box is naturally partitioned. The box that a point is in is completely determined by
//the first 2l bits of its components. First l (from left) bits are where y is, next l are x.
inline int Box(point p, int l){
//shifts x, and y left by l bits by multiplying by 2^l.
int column =(int) (p.x*pow(2.0,(double) l));
int row    =(int) (p.y*pow(2.0,(double) l));
return row*pow(2.0,(double) l)+column;
}
inline int Image_Index(int m, int n){
return N*n+m;
}
inline int Data_Index(int s,int w){
return N*s+w;
}
inline double X(int m){
return m/(double)N+ 1.0/(2.0*N);
}
inline double Y(int n){
return n/(double)N+ 1.0/(2.0*N);
}
inline double Gamma_D_X(double sl){
```

```
return Gamma_X(sl)/sqrt(Gamma_X(sl)*Gamma_X(sl)+Gamma_Y(sl)*Gamma_Y(sl));
}

inline double Gamma_D_Y(double sl){
return Gamma_Y(sl)/sqrt(Gamma_X(sl)*Gamma_X(sl)
+Gamma_Y(sl)*Gamma_Y(sl));
}
inline double W(int w){
return w_0-Band/2+w*Band/(double)N+1.0/(2.0*N);
}

inline double Q(int e){
  return e/N+1.0/(2*N);
}
inline double S(int s){
return s/(double)N+1.0/(2*N);
}

inline void Print_Weights(complex<double>* weights,int l)
{
for(int n=0;n<M;n++){
//go through the children of B paired with the parent of A to compute weights
for(int t1=0;t1<q;t1++){
for(int t2=0;t2<q;t2++){

cout<<weights[Weight_Index(A(n,l),B(n,l),l,t1,t2)]<<" ";
}
}
}
}


inline void Print_Grids(double *grids)
{
for( int l=0;l<=L;l++){
//go through each offset at that scale.
for(int Box=0;Box<(int)(pow(2.0,l));Box++){
for( int t=0;t<q;t++)
{
cout<<grids[Grid_Index(Box,t,l)]<<" ";
}
cout<<endl;
}
}
}

//Perform image recovery using butterfly algorithm
//Parameters are defined in input.h. Conventions are handled by functions above.
//We follow the algorithm developed in: "A butterfly algorithm for synthetic aperture radar imaging"
//At each level there are N^2q^2 weights to compute. We need weights from the previous level
//to compute weights at current level.
void Image_Recovery_Butterfly(double*& recovered_reflectivity, complex<double>* data){
int l=0;
//Constants outside the sum
double C=-3200*Band*pi/(M*c_0*c_0),qj=0,sl=0;

point x,y;
//complex valued sum. We will need to take modulus before
//it is reflectivity estimate.
complex<double> *SUM=new complex<double>[M];
recovered_reflectivity=new double[M];

for(int n=0;n<N;n++){
for(int m=0;m<N;m++){
SUM[Image_Index(m,n)]=0;
recovered_reflectivity[Image_Index(m,n)]=0;
}
}

complex<double> *previous_weights=new complex<double>[M*r_eps];
complex<double> *weights = new complex<double>[M*r_eps];

cout<<"Made it here"<<endl;
//Compute all the 1-d grids we will need.
double *grids;
Build_Grids(grids);
Print_Grids(grids);
cout<<"Well the grids were built"<<endl;
//Step 1: Initialize the weights. Done different than the paper to not need tree.
//go through all the points that we are integrating over and add them to the weight
//for the box they are in. This is so we do not need to have an actual tree structure
//The exponential is moved inside the sum.
```

```
   Zero(weights);
Zero(previous_weights);
  for(int e=0;e<N;e++){
for(int o=0;o<N;o++){
y=point(S(o),Q(e));
if(abs(f(y,data))!=0){
for(int t1=0;t1<q;t1++){
for(int t2=0;t2<q;t2++){
weights[Weight_Index(0,Box(y,L),l,t1,t2)]+=
exp(-i*(phi(Center(0,0),Grid_Point(grids,Box(y,L),t1,t2))
-phi(Center(0,0),y)))
*L2d(grids,t1,t2,Box(y,L),L,y)
*f(y,data);
}
}
}
}
}



cout<<"First Level is done"<<endl;
Swap(previous_weights,weights);
Zero(weights);
//Step 2: for l=1,..., L/2
//Compute the next weights from previous weights using the fact when diam(B)<1/N^2
for(l=1;l<=L/2;l++){
std::cout<<"Current level is"<<l<<endl;
//go through all the pairs of boxes.  There are N^2.
for(int n=0;n<M;n++){
//go through the children of B paired with the parent of A to compute weights
for(int t1=0;t1<q;t1++){
for(int t2=0;t2<q;t2++){
for(int c=0;c<4;c++){
//for each child parent pair
for(int tc1=0;tc1<q;tc1++){
for(int tc2=0;tc2<q;tc2++){
if(abs(previous_weights[Weight_Index(Parent(A(n,l)),Child(B(n,l),c),l-1,tc1,tc2)])!=0){
weights[Weight_Index(A(n,l),B(n,l),l,t1,t2)]+=
  L2d(grids, tc1,tc2, B(n,l),L-l,Grid_Point(grids,Child(B(n,l),c),tc1,tc2))
*exp(i*phi(Center(A(n,l),l),Grid_Point(grids,Child(B(n,l),c),tc1,tc2)))
*previous_weights[Weight_Index(Parent(A(n,l)),Child(B(n,l),c),l-1,tc1,tc2)];
}
}
}
}
weights[Weight_Index(A(n,l),B(n,l),l,t1,t2)]
  *=exp(-i*phi(Center(A(n,l),l),Grid_Point(grids,B(n,l),t1,t2)));
}
}
}
Swap(previous_weights,weights);
Zero(weights);
}


cout<<"Step 2 done!!"<<endl;
//Step 3: for l=L/2 Do switch
for(int n=0;n<M;n++){
for(int t1=0;t1<q;t1++){
for(int t2=0;t2<q;t2++){
for(int s1=0;s1<q;s1++){
for(int s2=0;s2<q;s2++){
if(abs(previous_weights[Weight_Index(A(n,L/2),B(n,L/2),L/2,s1,s2)])!=0){
weights[Weight_Index(A(n,L/2),B(n,L/2),L/2,t1,t2)]
+=exp(i*phi(Grid_Point(grids,A(n,l),t1,t2),Grid_Point(grids,B(n,l),s1,s2)))
*previous_weights[Weight_Index(A(n,L/2),B(n,L/2),L/2,t1,t2)];
}
}

}
}
}
}
cout<<"Step 3 is done!!"<<endl;
Swap(previous_weights,weights);
Zero(weights);
//Step 4: for l=L/2,...,L Compute the next weights from previous
//weights using the fact that diam(A)<1/M handle l=L/2+1,... L
   for(;l<=L;l++){
std::cout<<"Current level is"<<l<<endl;
for(int n=0;n<M;n++){
```

```
//set grid for current box b.
for(int t1=0;t1<q;t1++){
for(int t2=0;t2<q;t2++){
for(int c=0;c<4;c++){
//for each child parent pair
for(int tp1=0;tp1<q;tp1++){
for(int tp2=0;tp2<q;tp2++){
//The indices for the pairings are given by
if(abs(previous_weights[Weight_Index(Parent(A(n,l)),Child(B(n,l),c),l-1,tp1,tp2)])!=0){
weights[Weight_Index(A(n,l),B(n,l),l,t1,t2)]+=L2d(grids,tp1,tp2,Parent(A(n,l)),l+1,Grid_Point(grids,A(n,l),t1,t2))
*exp(-i*phi(Grid_Point(grids,Parent(A(n,l)),tp1,tp2),Center(Child(B(n,l),c),L-l+1)))
*previous_weights[Weight_Index(Parent(A(n,l)),Child(B(n,l),c),l-1,tp1,tp2)];
}
}
}
weights[Weight_Index(A(n,l),B(n,l),l,t1,t2)]
=exp(i*phi(Grid_Point(grids,A(n,l),t1,t2),Center(l,Child(B(n,l),c))))
*previous_weights[Weight_Index(Parent(A(n,l)),Child(B(n,l),c),l,t1,t2)];
}
}
}
}
Swap(previous_weights,weights);
Zero(weights);
}
cout<<"Step 4 is done!!!!!!!!!!!!!!!!"<<endl;
//undo the last Swap since we need the latest computed weights.
Swap(previous_weights,weights);

//Step 5: use equivalent weights to compute full sum. B=0 since we are doing the entire integral
for(int n=0;n<N;n++){
for(int m=0;m<N;m++){
x=point(X(m),Y(n));
//Get the box A that x is in
for(int t1=0;t1<q;t1++){
for(int t2=0;t2<q;t2++){
SUM[Image_Index(m,n)]+=L2d(grids,t1,t2,Box(x,L),L,x)
  *exp(-i*phi(Grid_Point(grids,0,t1,t2),Center(0,0)))
  *weights[Weight_Index(Box(x,L),0,L,t1,t2)];
}
}
```

```
SUM[Image_Index(m,n)]=C*exp(i*phi(x,Center(0,0)))*SUM[Image_Index(m,n)];
recovered_reflectivity[Image_Index(m,n)]=abs(SUM[Image_Index(m,n)]);
}
}
//clean up memory
delete[] weights;
   delete[] previous_weights;
delete[] SUM;
}


void Image_Recovery_Direct(double * recovered_reflectivity, complex<double>* data){
double A=-3200*Band*pi/(M*c_0*c_0);
complex<double> * SUM= new complex<double>[M];
for(int m=0;m<N;m++){
for(int n=0;n<N;n++){
SUM[Image_Index(m,n)]=0;
}
}
std::cout<<"Initialized sum"<<endl;
for(int m=0;m<N;m++){
std::cout<<"Working on Row "<<m<<endl;
for(int n=0;n<N;n++){

for(int s=0;s<N;s++){
for(int w=0;w<N;w++){
SUM[Image_Index(m,n)]+=exp(-i*W(w)*(2*(X(m)*Gamma_D_X(S(s))+Y(n)*Gamma_D_Y(S(s))-r))/c_0)
  *conj(p(W(w)))*data[Data_Index(s,w)]/(W(w)*abs(p(W(w)))*abs(p(W(w))));
}
}
}
}

std::cout<<"Sum  was computed"<<endl;
for(int m=0;m<N;m++){
for(int n=0;n<N;n++){
cout<<"Computed sum is:"<<SUM[Image_Index(m,n)]<<endl;
recovered_reflectivity[Image_Index(m,n)]=A*abs(SUM[Image_Index(m,n)]);
}
}
std::cout<<"Reflectivity was built"<<endl;
```

```
}


//Output reconstruction for viewing in Matlab.
void Output_Reflectivity(double * reflectivity,string file){
ofstream output;
output.open(file.c_str());
for(int n=0;n<N;n++){
for(int m=0;m<N;m++){
   output<<reflectivity[Image_Index(m,n)]<<" ";
}
output<<endl;
}
output.close();
}
```

BIOGRAPHICAL SKETCH

Alejandro(Alex) Federico Martinez was born in Weslaco, TX on November 22, 1989. His childhood was spent primarily in San Benito and Elsa Texas. After graduating from San Benito High School he attended the University of Texas Pan American from 2008 until graduation in 2012. He majored in computer science and mathematics. His interest include algorithms, DNA self-assembly, electromagnetics, physics, imaging, game development, and all sorts of mathematics. An elegant data structure is something to be excited about. Occasionally he will learn some history and philosophy. From 2012-2014 he worked on his masters degree in mathematics. As an undergraduate he participated in a seminar on DNA self-assembly. From 2009-2014 he worked in the "DoD Project on PDE Analysis and Radar Image Reconstruction" program. Alex has presented work in imaging at the 2012, 2013, and 2014 SPIE Defense, Security, and Sensing Conferences. He can be contacted at mnight08@yahoo.com or afmartinez4@broncs.utpa.edu if anyone wants to talk about code or mathematics.