THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

ENGINEERING SCIENCE AND MECHANICS

IMPLEMENTING AN ACOUSTIC PROPAGATION AND SCATTERING MODEL FOR SIMULATING SYNTHETIC APERTURE SONAR DATA

JOHN R. BROWNSTEAD
SPRING 2025

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Engineering Science
with honors in Engineering Science

Reviewed and approved* by the following:

Daniel C. Brown
Associate Research Professor
Thesis Supervisor

Andrea P. Argüelles
Associate Professor of Engineering Science and Mechanics
Thesis Supervisor

Gary L. Gray
Associate Professor of Engineering Science and Mechanics
Honors Advisor

*Signatures are on file in the Schreyer Honors College

Abstract

The objective of this thesis is to develop an acoustic scattering and propagation model. This model includes fundamental physical effects, can be used to create synthetic sonar data, and allows users to identify the optimal strategy for introducing new physical effects into existing models. This model specifically produces synthetic aperture sonar data using a seafloor that consists only of a user-defined planar scene. The benefit of this model compared to the already existing and more sophisticated sonar models is its simplicity and ease of modification. This allows the user to implement new physical effects using various methods in a low risk environment to determine which method optimizes image quality and the required number of computations. Throughout this thesis, the theory behind many acoustic propagation and scattering models is be explained. This thesis should assist readers unfamiliar with acoustics in understanding acoustic propagation and scattering models and in comparing different models.

Nontechnical Abstract

This thesis serves as a guide to the development of a model that simulates the propagation and scattering of sound waves. The model developed can be used to evaluate the tradeoff in accuracy and run time for different model customizations. This model is generalizable, but it was specifically developed to model a sonar system with a single transmitter and receiver that travels in a straight path and insonifies a seafloor which was constructed using a .png file. This model, along with existing and more sophisticated models, can be used for synthetic data production. This synthetic data can be used to train operators, to train automatic target recognition systems, and to inform the design of sonar hardware. Synthetic data is often preferred to real-world data because real-world data can be costly, difficult to find naturally, time-consuming, and often does not have a known ground truth.

Table of Contents

List of Figures

Acknowledgments

Thank you, Dr. Brown, Dr. Argüelles, Dr. Gray, the Department of Engineering Science and Mechanics at Penn State, and the Applied Research Laboratory at Penn State for providing both an enjoyable and enriching college experience. Thank you to my family and friends for all of your support.

Chapter 1
Introduction

1.1 Overview of Engineering Problem

Sonar systems and algorithms are constantly being developed and data is needed for testing and training. Acquiring real-world sonar data to test and train these new technologies can be expensive, time-consuming, and inconvenient. Additionally, specific underwater environments in which it may be desirable to test a sonar system may not be common in nature or the environment may be changing over the course of an experiment. Furthermore, knowledge of precisely what the real-world environment consists of and how it is organized is unknown and having a known ground truth is valuable. Simulated sonar data is an alternative to real-world testing that can be designed to overcome some of these drawbacks. Sonar data has many uses, some of which are to train operators, to train sonar systems that rely on large amounts of data like automatic target recognition, to optimize the design of sonar sensors for object detection [2], to develop and evaluate acoustic navigation techniques [3], or to test models for geothermal activity [4]. The development of propagation and scattering models and simulated sonar data will improve testing, training, and modeling, which will support critical technologies. Navigation and hazard-detecting technologies benefit the defense, transportation, and fishing industries. Sonar seafloor imaging technology is used to study oceans and environments and is largely done using autonomous underwater vehicles. Producing rigorous models and large amounts of high quality simulated sonar data ultimately improves technologies in these critical industries and benefits national security, environmental studies, supply chains, and autonomous vehicle capabilities. This

thesis will provide an introduction to propagation and scattering models and to synthetic sonar data which will assist readers in learning about acoustics and model development.

1.2 Design Needs and Responsibilities

The objective of this thesis is to develop an acoustic propagation and scattering model which can simulate raw time series data for a simple synthetic aperture sonar system captured as a sonar moves past a seafloor which consists of any image. This model can be used to optimize the inclusion of new physical effects prior to their implementation into existing models. This model itself will have a minimal effect on public health and welfare and will also have a minimal effect on global, cultural, social, environmental, and economic factors. This model could have a marginal economic benefit if synthetic data is more widely used in lieu of real-world data. This model could also have an effect on public safety if it is used to improve existing synthetic data producing models. If those improved models are used to train autonomous systems or to train operators which are active in the real-world and they rely on synthetic data, the quality of the synthetic data could ultimately have real-world implications. Additionally, any lack of understanding of model functionality could produce synthetic data that results in poorly trained or biased systems and operators. Without proper testing, these biases could remain unidentified and could lead to errors.

Simulated acoustic data can be used to improve many technologies that are critical to modern society. This also means that errors in simulated acoustic data could go on to affect changes in these technologies. A false confidence in the capabilities of these models could have real-world consequences. The development engineers and authors of acoustic simulation literature have the ethical responsibility of communicating the true capabilities of these models and advocating only for appropriate applications. Many acoustic propagation and scattering models have already been developed and combined to produce simulations. The literature review of this thesis includes a review of many of those models and care will be taken to accurately attribute efforts to the appropriate authors to avoid plagiarism. This thesis will also be subjected to peer review to ensure its accuracy, originality, and responsible development.

Chapter 2
Literature Review

This section of the literature review provides background information before going into an overview of multiple acoustic scattering models. The background and models inform the experimental methods of this thesis. In section 2.1, a high-level overview of some common steps employed by acoustic propagation and scattering models to produce simulated data is provided. In section 2.2 an overview of some common acoustic propagation and scattering model concepts is presented. Then, sections 2.3 and 2.4 detail methods employed by acoustic scattering models when calculating scattering from the ocean environment and targets, respectively.

2.1 How Simulated Sonar Data is Generally Produced

The acoustic scattering models used to simulate sonar data can be broadly classified as either environment scattering models or target scattering models. Environment scattering models are used as a less computationally intensive option for simpler portions of the scene where minor details in real-world data would be lost. Target scattering models are generally more detailed and computationally intensive, so they are only used in specific portions of the scene where minor details are critical. Some steps which many scattering models follow are found in the list below.

1. A scene is discretized into some combination of points, line segments, shapes, and volumes. Points, line segments, and shapes are called scatterers and properties of these scatterers, such as the location, orientation, and material properties, are stored. Volumes are often associated with finite element methods, which are generally used for targets.

2. The movement of a sonar through the scene is defined.

3. Using the known positions and properties of the sonar and of the scatterers, geometrical relationships between the sonar and each scatterer, such as distances and angles, can be derived for use in implementing different physical effects.[1]

4. At each sonar position, these physical effects are used to modify the transmitted signal over a specific path and series of interactions with scatterers. This modified transmit signal becomes the simulated return signal.[2]

5. For a single transmit position, the return signals resulting from each individual scatterer or path in the scene are summed to produce a simulated return signal which is similar to real-world data. This process is repeated for each transmit signal in each position. This results in simulated sonar data.[3]

2.2 Theory-Based Overview of Acoustic Simulation Concepts

For sonar simulations, two of the most prevalent constituent models are acoustic propagation and scattering. Propagation models can be used independently or in combination with scattering models to propagate a wave to or from a transducer. Scattering models employ mathematical and statistical methods to implement physical models of acoustic scattering to produce simulated data. Sections 2.2.1 and 2.2.2 provide a review of theoretical concepts which are applied when implementing propagation and scattering models, respectively.

[1]Each model includes different physical effects which affect the accuracy and computational intensity of the model. Some complicated physical effects can be approximated using computationally efficient correction factors.

[2]For target models, finite element methods are generally used only to simulate the scatterer amplitude of a target and other methods are used to propagate that wave beyond a very small zone around the target to mitigate computational requirements.

[3]After the sonar data has been collected, one possible method of processing the data is to use replica correlation and beamforming to produce an image of the seafloor. This allows the user to easily visualize the results of the simulation and of including or excluding certain physical effects.

2.2.1 Wave Propagation

After a wave is transmitted, it propagates through a medium. In the far field of propagation, waves become easier to model and can be approximated as planar, while a source can be approximated as a point. The environments in which a wave propagates can be broadly classified as isovelocity or non-constant velocity. The different ways that this propagation can be modeled in these environments is the focus of this section.

- **Absorption**

  As a wave propagates through a medium, its energy is absorbed into the medium through two primary processes. First, there are frictional losses due to the viscosity of the medium as molecules collide with one another as the wave propagates. These viscous losses vary with the square root of temperature and the square of frequency. Second, largely in air, there are relaxation processes that cause the oxygen and nitrogen in air to vibrate and rotate due to the energy in the traveling wave [5]. Absorption is optional to be included in acoustic scattering models and is more critical to air-based models than it is to water-based models.

- **Isovelocity Propagation**

  Isovelocity propagation occurs in environments where the speed of sound throughout a medium is constant. For the ideal case of an isovelocity environment, waves travel in straight lines which results in either spherical or cylindrical spreading. Power is conserved as a wave propagates through a non-absorbing medium. Power is equal to the product of intensity and area. For spherical spreading, the area of the spherical shape in which a wave propagates increases and varies with the square of the distance traveled. Consequently, the intensity of the wave decreases and varies inversely with the square of the distance traveled. The result of spherical spreading appears in Green's equation. Cylindrical spreading can be used when the wave is confined to a two-dimensional space instead of a three-dimensional space. This may occur if the top and bottom boundaries of a medium produce perfect reflections. Cylindrical spreading causes the intensity of the wave to vary inversely with the distance traveled.

- **Non-constant Velocity Propagation**

  A more detailed model for the environment should be used when the material properties of the medium of propagation vary spatially and cause propagating waves to refract. For sonar models, the variation of properties can result from varying temperature, pressure, salinity, and more. To model the path of a wave in these non-constant velocity environments, ray tracing can be used. Ray tracing models consider the refraction of a wave as it travels through a medium which has varying material properties. For a non-constant velocity environment, a wave will refract and travel in a curved path. Ray tracing models, such as BELLHOP [6], consider this effect and can be incorporated into scattering models.

- **Wave, Helmholtz, and Green's Equations**

  The linear acoustic wave equation for inviscid fluids relates velocity potentials through time and space and can be derived from the Navier-Stokes equations in fluid mechanics. This equation has infinite solutions, so boundary conditions need to be applied to restrict the equation and lead to specific solutions. This equation can be found in (2.19) of [7].

  The Helmholtz equation, (2.25) in [7], can be derived by taking the Fourier transform of the linear acoustic wave equation. The Helmholtz equation provides the frequency domain representation of the velocity potentials. Fourier decomposition can separate the wave equation into many Helmholtz equations, one for each frequency.

  Green's function is a solution to the linear wave equation for an impulse or point source. The free space Green's function, which is used for homogeneous media, is (2.30) in [7] and clearly models propagation by including the phase shift and spherical spreading of a point source if no other sources are present. Green's function can be extended through superposition to find the solution to the linear wave equation for any arbitrary formation of sources.

2.2.2 Wave Scattering

When a wave reaches a change in medium, there can be a change to the wave's phase, amplitude, and direction. The methods used to model reflections, which are

the portion of the wave that returns into the original medium of travel, are the focus of this section.

- **Specular Scattering**

  If after scattering, the reflected wave has an angle of reflection equal to its angle of incidence and travels in the plane defined by the vector normal to the surface and the direction of travel of the incident wave, then the reflection of the wave is said to be specular.

  The reflection coefficient is the complex ratio of incident and reflected waves. This coefficient depends on the material properties of the two media at an interface and can impose a change in both amplitude and phase. If the phase structure of the reflected wave differs from the phase of the incident wave, the reflection is considered incoherent; otherwise, if the phase structure is unchanged through reflection, the reflection is considered coherent.

- **Diffuse Scattering**

  If after scattering, the reflected wave does not travel in the specular direction, this is called a diffuse wave. The wave scattered in the specular reflection direction will have the greatest amplitude and the scattered amplitude will decrease as the angle between the specular direction and the scattered direction increases.

- **Scatterer Strength, Scatterer Amplitude, Target Strength, and Cross Section**

  The scattering strength, also known as the scatterer amplitude, is the ratio of the intensity of the incident and scattered waves over a volume or area at a distance of 1 meter. The target strength has the same formulation but is applied to what is considered to be a target [8]. The scatterer strength can be defined differently depending on the acoustic model, but is still described as the ratio of the scattered and incident wave intensities [7]. The square root of the cross section is typically related directly to the scatterer amplitude, but other factors or relationships can be used.

- **Multiple Scattering**

The re-radiation of sound is called scattering and the sum total of all scattering is called reverberation [8]. After interacting with one scatterer, a wave may travel toward the receiver, or it may travel toward another scatterer before returning to the receiver. Wave paths which include multiple scatterer interactions can frequently be ignored because the amplitude of the reflected wave decreases with each scatterer interaction and with travel distance, and because the multiple scattered signal is returned much later in time than the initial return.

- **Beampattern**

  A beampattern describes how the amplitude of an emitted wave varies with direction. The main lobe of a beampattern is typically in the specular direction and there is a gradual drop off in amplitude for any other direction.

- **Aperture Function**

  An aperture is another term for a transducer or projector. The aperture function describes the shape of the aperture. Under the Fraunhoffer approximation for the scattered field, the Fourier transform of the aperture function is directly related to the scattered field through the beampattern [7].

- **Helmholtz-Kirchhoff Integral Equation**

  The Helmholtz-Kirchhoff Integral Equation (HKIE) relates velocity potentials at a certain position in a volume to velocity potentials and their normal derivatives over a closed surface. The HKIE, (2.35) in [7], is often used to model the reflected wave from the surface of a scatterer. Solutions to the HKIE are computationally intensive for most geometries and require knowledge of the incident field and Green's function for the boundary. Approximations such as the Kirchhoff approximation are used to making finding solutions easier.

- **Kirchhoff Approximation**

  The Kirchhoff approximation is used to simplify boundary conditions for rough interfaces for which the scattered field will be calculated via the HKIE. The Kirchhoff approximation treats the reflected field from each point on a surface equivalently to the field reflected from a plane at each point and

utilizes the reflection coefficient [9]. Additional simplifications can be realized by using the narrow band and small slope approximations. The narrow band approximation assumes that certain parameters are frequency-independent. The small slope approximation assumes that a region of the seafloor is flat and has minimal variations. The Kirchhoff approximation can be applied if the rough surface is sufficiently smooth so that multiple scattering and shadowing are avoided. Using the Kirchhoff approximation leads to a specific formulation of the scattering cross section, which can readily be incorporated into the T-matrix method, as shown in section L.1 of [10].

- **Finite Element Methods**

  Partial differential equations are used to describe many physical or mathematical concepts and they are difficult or impossible to solve analytically. The finite element method is a numerical solution to partial differential equations which reduces the domain from infinite to finite and utilizes known boundary conditions to solve systems of equations [11].

- **Boundary Element Method**

  The boundary element method, also known as the boundary integral method, is a numerical method used to solve partial differential equations such as the Helmholtz equation [12]. The boundary element method replaces an equation that operates on a three dimensional domain with an integral equation that only operates on a two dimensional boundary within the domain before solving the equation numerically. For example, in (1.1) of [12], this allows the velocity potential and its derivative to be directly related along a surface domain in an integral equation. For specific applications, the boundary element method is often preferred to finite element methods since the domain of the problem is reduced by one dimension, which reduces computations.

- **Doppler Shift**

  If a source, receiver, or target are moving relative to one another, the object's apparent frequency to an observer will be different from its actual frequency. The Doppler shift results from the compression or expansion of time scales rather than an actual change in frequency.

- **T-Matrix Method**

  The T-matrix method relates the amplitudes of incident and reflected plane waves, as shown in section J.3 of [10]. Vector waves with amplitude and phase can be expanded into a series format. This series format is scaled by the columns of the T-matrix. The T-matrix can be thought of as a reflection coefficient and can be extended to easily incorporate multiple scattering. The T-matrix can also be decomposed to separately consider both the specular and diffuse components of a wave.

2.3 Overview of Environment Scattering Models

2.3.1 TriKirch Model

This summary was created using Abawi's 2016 paper on modeling scattering from a triangularly faceted target [13]. These triangular facets are non-penetrable, which means that only the surface of the environment contributes toward the scattered return signal. This is a fair assumption for perfectly rigid materials. The environment is discretized into triangles which have a side length of at most $\lambda/5$, where λ is the wavelength of the incident sound wave. Using increasingly small triangles produces a more accurate approximation for the environment which leads to the analytic solution if a limit is taken. Triangles are used because they are more powerful than other shaped facets and can represent any arbitrary surface. Abawi's model relies on the Kirchhoff approximation. The Kirchhoff approximation is valid when the wavelength of the incident field is much smaller than the size of the facet and when the interaction takes place in the far-field. In Abawi's model, planar triangular facets make up a surface and reduce computations relative to other shapes without sacrificing significant accuracy, as long as the facets are sufficiently small. Abawi presents an expression for scattering from a planar triangular facet. Abawi's expression coherently sums the responses from each facet to calculate scattering. The expression for scattering from a single facet is found in (7) of [13]. Abawi's approximation depends on the facet edge vectors, the direction of the incident and reflected wave, and a position vector for one vertex of the facet. Abawi's model is referred to as the TriKirch approximation due to triangular facets being combined with the Kirchhoff approximation to model scattering.

2.3.2 Hunter's Model

This summary was created using Alan Hunter's extension of the Kirchhoff model which is described in his PhD dissertation [7]. Before Hunter's thesis, the typical method of discretizing the environment in a scene was to use a large number of points or smooth facets. Hunter recognized that a smaller number of rough facets could be used in place of many smooth facets, which reduces computational requirements. Additionally, with sufficiently large facets, edge effects can be ignored. These facets in Hunter's thesis are characterized by their far-field beampatterns. The beampattern statistics are derived from the shape and roughness statistics of each facet. These beampatterns account for both the specular and diffuse components of the reflected field.

Hunter is able to calculate scattering from a rough facet using two key functions. First, the aperture function describes the general shape of the facet. Second, the height function describes the deviations of the actual surface from the mean plane surface. These two functions are included in the facet's beampattern, which is a function of facet shape, facet orientation, the acoustic properties of the two media at a surface, and the statistics used to describe the facet roughness. The beampattern is described in (3.47) of [7]. The statistics of the beampattern are determined by the Gaussian statistics of the approximately stationary facet's height function. Hunter argues that an empirically valid approximation is to assume that the field scattered from a rough surface follows a Gaussian normal distribution for both its real and imaginary components. A Gaussian distribution isn't the most accurate known statistical representation, but due to its simplicity, it is used. Statistical representations can be used since many minor scatterer features are unresolvable.

Scattering is one portion of the overall model and Hunter calculates the scattered field by using the Kirchhoff approximation, Green's function, and the Kirchhoff-Helmholtz integral. The result for scattering from a smooth triangular facet is found in (3.28) of [7] and includes the beampattern as a scale factor. This smooth facet beampattern can be substituted with a rough facet beampattern. Hunter uses the Fraunhoffer approximation to determine a portion of the far-field beampattern of a facet by taking the Fourier transform of the aperture function.

2.3.3 Point-based Sonar Scattering Model (PoSSM)

This summary was created using the PoSSM paper from Brown, Johnson, and Olson [14]. PoSSM can be used to find the incoherent component of the field scattered from an individual point-based scatterer. This can be applied to a collection of points that make up a scene. PoSSM includes physical effects like directivity functions, spherical spreading, and time delays. One of PoSSM's notable features is the method it uses to ensure incoherence between any two points and generate surface roughness. PoSSM does this by scaling the transmitted linear spectrum by a non-physical stochastic scale factor.

In PoSSM, first, the deterministic component of the scatterer amplitude is calculated. The amplitude of each scatterer varies directly with the cross section per unit area per unit solid angle and inversely with the scatterer density (scatterers/m^2). The cross section accounts for the roughness of the seafloor and can be calculated using external models. The scatterer density is included to ensure that the reflection resulting from an area of scatterers is appropriate regardless of scatterer concentration. Next, the stochastic component of the scatterer amplitude is included. The deterministic amplitude is scaled by a complex circular Gaussian random variable. This non-physical scale factor associates a frequency-independent phase shift and random amplitude scalar with each point scatterer. This scale factor is included because it allows the spatial coherence of the scattered field to agree with the analytic solution, which is the van Cittert-Zernike theorem. This stochastic scale factor is appropriate as long as the surface correlation length of the scene attempting to be simulated is smaller than the spacing between scatterers and must be altered otherwise.

2.3.4 Personal Computer Shallow Water Acoustic Tool-set (PC-SWAT)

This summary was created using Sammelman's PC-SWAT overview [15] and detailed review [16]. To simulate rigid target scattering, PC-SWAT first discretizes an object into rectangular facets and uses the Kirchhoff approximation to find the first order approximation for the target strength. The scattered field is found in (2.2) of [15] and depends on the reflection coefficient of the surface, the source and facet locations, the vectors which define the facet's perimeter and surface normal,

and on frequency. The field scattered from individual facets can be coherently summed to determine the total scattered field. For improving accuracy, PC-SWAT considers the edges between those facets and uses Keller's theory of edge diffraction to calculate scattering from an edge. Keller's theory applies to edges of infinite length and this theory was extended to edges of finite length by Pierce (reference 2 in [15]) before it was modified for use in PC-SWAT. The modification weights Pierce's solution by the directivity function of the edge. The scattering from an edge is described in (2.37) of [15]. Additionally, for finer details in small protrusions of a target, point scatterers can be used with simple target strength approximations which are not specified.

PC-SWAT has different methods for calculating scattering from rigid targets, from solid elastic targets, from elastic targets, from rough interfaces, and from volumes. The general method used to determine these scattering relationships uses Green's function, the Helmholtz Integral Equation, and other theories that ultimately lead to a T-matrix. PC-SWAT can be used for either environment scattering or for target scattering and switching between the two can be easily done by changing assumptions and specific models used to calculate scattering.

2.3.5 HoloOcean

HoloOcean is an environment simulator intended to be used for autonomous underwater vehicles (AUVs). The model is based on UE4, which is a video game building engine that comes with some built-in underwater effects [17]. HoloOcean includes the ability to create sonar images. HoloOcean tracks emitted sound waves and uses reflection coefficients, geometry, and time delays to model return signals, which are used to construct sonar images.

To simulate these scatterer amplitudes, the surface of any target is discretized into clusters of points (clusters are points within a small range of transmit angles). For each point, if the surface normal is facing the potential receiver, the point is kept. After all these points are identified, points in shadows (which may be facing the receiver, but are obstructed by other points) are removed. Removing shadows is done by analyzing all points within a small cluster. Within this cluster, points are sorted by ascending range and after the difference between adjacent points exceeds a set value, those points are removed and considered shadows. The value of each

cluster is then calculated by averaging the angle between the incident wave and surface normal for each scatterer, scaling by noise that follows a normal distribution with mean 1, and adding noise that follows a Rayleigh distribution. This value, for each cluster, represents the scatterer amplitude of a pixel in the simulated sonar image.

HoloOcean was then revised in [18] to use a cluster-based ray-tracing algorithm to simulate multi-path noise, to use improved probabilistic noise models, and to use material dependence to better model underwater noise. In tracking the multi-path rays, scatterers are grouped into clusters and the central scatterer is used to compute ray tracing. These clusters must be within a certain range of the original cluster and have a normal vector that is within 15°of the central scatterer. HoloOcean assumes planar reflections for multi-path tracing. The algorithm presented includes reflections off of two surfaces prior to returning to the receiver. The scatterer amplitudes are dependent on ray geometries and the reflection coefficient for the two materials. The intensity returning from a cluster is found by averaging the results from individual points. These coefficients are defined in equations (4) and (5) of [18].

The stochastic component of the model in [18] accounts for the typical diffusion of time of flight arrivals. It does this by adding an exponential distribution to the location of each scatterer, $Exp(\lambda)$, to artificially increase the distances of scatterers from the receiver by a small amount. This also affects the scatterer intensity. The intensity of each scatterer is scaled by $exp(\frac{-Exp(\lambda)}{\lambda})$ to include the change in intensity. Finally, the additive noise was altered to influence the speckle noise and to be adjusted for range. The additive noise became a product of the normalized range squared and the original Rayleigh distribution.

2.4 Overview of Target Scattering Models

2.4.1 Abawi's Finite and Boundary Element Method

This summary was created using Abawi's target scattering model which uses finite and boundary element methods [19]. Any elastic target and its surroundings can be accurately modeled using finite element analysis. However, the extreme computational requirements associated with finite element methods encourage the

use of alternative methods when possible. Abawi's method uses finite element methods to model the motion of an elastic target as it pulses after being insonified. Section II of [19] details the specific derivation of the finite element method. Abawi then uses the boundary element method to model the propagating pressure field in the medium surrounding the target. The boundary element method and finite element methods are coupled by the normal pressure and velocity at the surface of the target, as calculated in the finite element method. Abawi's method uses the Helmholtz-Kirchhoff Integral Equation to propagate the scattered field as found in (33) of [19]. The boundary integral method combines Fourier series for the symmetric and antisymmetric components of the scattered field with Green's function before applying axially symmetry arguments to reduce the complexity of the equation. This equation leads to the pressure field, which is defined in (56) of [19]. The use of the boundary element method as opposed to a finite element method for calculating the pressure field in the surrounding medium produced valid results and reportedly led to a 36 times improvement in the speed of the model.

2.4.2 Sonar Simulation Toolset (SST)

The SST manual was authored by Robert Goddard within the Applied Physics Laboratory at the University of Washington [20]. Some key effects included in SST are diffuse scatterers (reverberation), discrete scatterers (which are considered to be active targets), diffuse sound sources (such as the environment and self), and finally discrete sound sources which emit noise from a compact region. SST uses point scatterers. SST also has a direct sound propagation model, where a wave travels directly from source to receiver. These direct propagation models are summarized in (47-52) of [20]. The primary focus of [20] is on explaining how reverberation is calculated. The other sources of sound rely on the same underlying methods and functions as reverberation, but use varied assumptions. SST can be used to calculate responses from both environment and target scatterers.

SST has two methods of using point scatterers with varying accuracy. SST treats a scatterer as a transducer which receives a signal, scales it, and then transmits the scaled version. The first scattering model, *PointTarget*, allows the user to define a list of the positions and velocities of point scatterers based on a target's coordinate system. The user can then choose between two options. One option is to define

scatterers around a target that share scattering features and are only differentiated by their arrival times. A second and more accurate option is to define scatterers around a target with distinct scattering features, which allows scatterers to be differentiated by arrival time and by arrival direction. Additionally, a Gaussian random component can be added to scatterer locations to reduce undesirable effects from perfectly organized scatterers. The second scattering model is *HighlightTarget*, which improves upon *PointTarget* in two ways. First, each scatterer can have a complex, frequency-dependent response. Second, each scatterer can experience a delay between reception and transmission.

When calculating reverberation from diffuse scatterers, SST considers eigenray propagation, multiple scattering, randomly distributed scatterers, and scatterer motion. This scatterer motion causes a distribution of Doppler shifts among groups of scatterers. The motion of the source or receiver also causes a Doppler shift. For reverberation, the scatterer amplitude follows the Poisson scattering assumption and is the product of multiple delta functions and a differential cross section per unit volume per unit Doppler shift, as seen in (76) of [20]. The Poisson distribution ensures that receive signals are unrelated with one another and that a Poisson distribution across one domain can be superimposed with a Poisson distribution on another domain to produce a valid overall Poisson distribution. According to the Poisson assumption, one Dirac delta function is for making scatterers uncorrelated regardless of physical proximity. A second Dirac delta function is for making scatterers at the same location with different Doppler shifts uncorrelated. Two Kronecker delta functions are included so that signals arriving from any two different eigenray pairs are uncorrelated. When calculating volume layer scattering, SST assumes that all scatterers lie on a plane in the middle of the layer, which is a reasonable approximation except at close ranges. Reverberation typically has Gaussian statistics since there are enough points in each resolution cell of the sonar. With some simplifying assumptions and grouping, those delta functions can be represented by an ordinary Gaussian distribution weighted by the size of the group of scatterers, as seen in (111) of [20]. To calculate the return signal due to reverberation, the source signal is multiplied by the scatterer amplitude, the transmit and scattering beam patterns, the path propagation losses, and time delays resulting from offsets in different channels from the center of the transducer array (if a group of points is represented by a single array).

2.4.3 Target In the Environment Response Model (TIER)

The Applied Physics Laboratory at the University of Washington's TIER model develops an efficient ray tracing algorithm which is intended to be combined with a finite element method [21], [22]. TIER is intended to be used for target scattering when a target lies above the interface of water and a sediment layer. Passive noise and reverberation should be calculated using other models. TIER utilizes images of the target resting above a water-sediment interface to consider four paths between a source, target, and receiver that contribute to the return signal in a sufficiently short amount of time. These paths include the direct path, two paths with a single bounce on the bottom, and a path with two bounces off of the bottom. Additionally, the TIER model can be expanded to simulate data resulting from buried targets and it can include multiple scattering effects within a sub-bottom layer.

TIER calculates scattering as the superposition of many free field scatters.[4] This calculation relies on the assumptions that the surface between the waveguide and the sediment has no effect on the scattered field and that multiple scattering of the target does not occur. These assumptions are empirically shown to have a negligible effect on results. This leads to a formulation for the scattered linear spectrum, which is equal to the product of the transmitted linear spectrum, the free field scatterer amplitude, a number of reflection coefficients for the upper and lower interfaces, time delays, and a spherical spreading term. Additional restrictions can be applied which lead to the final equation for the scattered spectrum, (2) in [22]. This final form assumes that reflections with the air-water interface can be time-gated out and that the source and receiver are co-located. The scattered field from an incident plane wave is described below in (2.1), which is (2) in [22].

$$P(\omega) = [f_1 \frac{exp(i\omega t_0)}{d_0^2} + 2L(\theta_g) f_2 \frac{exp(i\omega t_1)}{d_0 d_1} + L^2(\theta_g) f_3 \frac{exp(i\omega t_2)}{d_1^2}] r_0 P_{src}(\omega) \quad (2.1)$$

In (2.1), $P(\omega)$ is the scattered spectrum, f_k is the scatterer amplitude, ωt is the frequency-dependent time delay, d is the distance between a target and an image source or receiver, and L is the reflection coefficient for the seafloor. The direct, one-bounce, and two-bounce paths are represented by the first, second, and third

[4]The free field is an extension of the far field in which a wave travels from source to receiver without interacting with any scatterers.

terms contained within the brackets of equation (2.1).

In [22], two models are described which together create TIER. The first model utilized a ray-tracing technique that tracks time of flight for different wavepackets and includes scattering from targets. The scatterer amplitude used in this model can be calculated using a few methods including analytic solutions, empirical data, or finite element methods. TIER is intended to be used with a provided finite element method to calculate the scatterer strength which provides f_k in (2.1). An alternative model to TIER would use the Helmholtz Integral and Green's function to calculate the propagation of the pressure field.

2.4.4 Stanton's Model

Dalton implemented Stanton's scattering model and used it to simulate the response from an elastic target [23]. Unlike rigid materials, elastic materials are capable of supporting shear waves which lead to the re-radiation of sound after an initial return. This late-time energy disrupts conventional target recognition algorithms, so a model which contains appropriately modeled elastic targets would be valuable for training. Finite element methods can be used to simulate target strength with late-time energy included, but they often have large computational requirements. Dalton presented a less computationally intensive and sufficiently accurate formulation of the scattering amplitude for an elastic open-ended cylinder derived by Stanton, (1) in [23]. The scatterer amplitude depends on the length of the cylinder, frequency, and geometries. The late-time energy appears in the re-radiated time series through destructive interference. This interference produces minima in the re-radiated output which produces pulses. These pulses are indicative of an elastic target.

Dalton compared the results of this elastic target strength to an ideal case presented by Urick for a rigid cylinder, (2) in [8]. The results showed great agreement. In Urick's formulation, target strength is a function of the cylinder diameter and length, frequency, and geometries.

Chapter 3
Methodology

The primary component of this experiment is creating an acoustic scattering model using methods detailed in literature. The developed scattering model will be used to evaluate the change in data quality and consequential change in required computations due to both including certain physical phenomena and using specific parameters while running the model. The inclusion and explanation of the computational approach used to create this scattering model will provide the reader with an example scattering model implementation which relies on the theory described previously.

Very few materials are required to run this experiment. The required materials include a computer, access to a coding platform, and an image. The model derived in this thesis uses MATLAB. Using these materials and principles of acoustic signal analysis, physics, and statistics, this experiment can be performed and replicated.

The acoustic scattering model developed for this thesis is based on Brown, Johnson, and Olson's PoSSM [14]. The model considers spherical spreading, transmit and receive beampatterns, scatterer amplitude, single scattering, and time delays when calculating the interaction of the transmitted time series with each scatterer. It is a simpler model than many existing models which allows the effect of including or excluding new physical phenomena and using specific parameters to be seen clearly in the results. The acoustic scattering model described will be tested on a flat seafloor which consists of scatterers organized to represent a Penn State logo. The different colors of the logo can be treated as different materials. The scatterer cross section is calculated using a product of Lambert's cosine law and a Lambert parameter valued between 0 and 1. Various parameters and physical effects, such as scatterer density, statistical assumptions, and setup geometry can

be varied during experiments. The results of varying some of these parameters will be shown visually by creating a synthetic aperture sonar image, generated using beamforming, of the Penn State logo on the seafloor. Since the ground-truth is known and provided to the simulation, the accuracy of the output of the simulation can be qualitatively compared to its expected value. The quality of the image can also be evaluated using a quantitative metric which is found in both the sonar and radar communities [24].

Chapter 4
Experimental Procedure

4.1 Motivation

The experimental procedure followed during this thesis consists of developing and testing MATLAB code. The MATLAB code used in this project can be found in Appendix A. The goal of this thesis project was to develop code which can produce simulated synthetic aperture sonar (SAS) data using methods described in [14]. For this simulation, synthetic sonar data would be collected from a sonar moving through a scene and scanning a seafloor in which any image is embedded. Sonar data, both synthetic and real-world, can be easily visualized using beamforming. Beamforming is a method of using geometry, an expected parabolic return pattern from individual scatterers (the sonar smile), Fourier transforms, and inverse Stolt mapping to solve for the scatterer amplitude as a function of position [25]. Wavenumber beamforming was used during this project to visualize the simulated sonar data and compare it with expected results. Beamformers produce figures displaying scatterer amplitude as a function of position. Matthews, et al. developed a SAS system with the objective of relating automatic target recognition performance to image quality. During this research, real-world SAS data was collected and beamformed with varying testing conditions showing the resulting varying image qualities. A SAS beamformed image with moderate image quality can be seen below in Fig. 4.1, which is Fig. 9 in [1].

This SAS image, and many other examples which can be found online, serves as a reference for a goal end-product of this project. The steps taken during this thesis project to achieve this goal will now be discussed in detail.

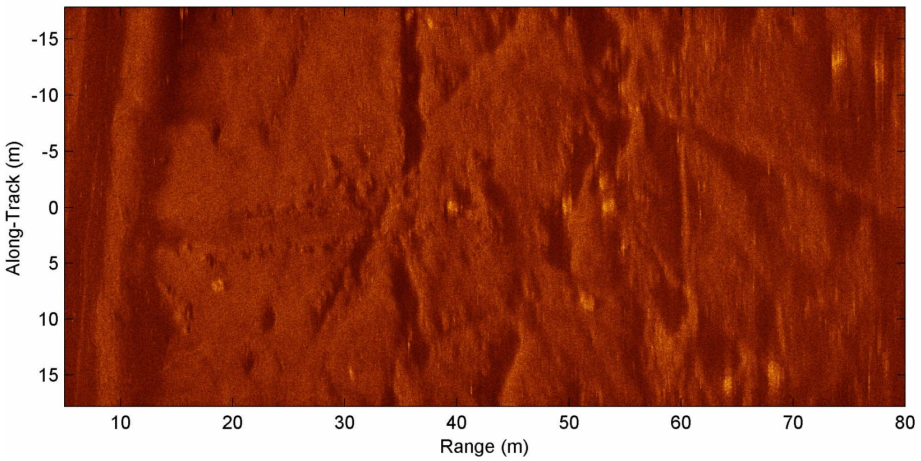

Figure 4.1. A real-world beamformed image of the seafloor constructed using synthetic aperture sonar data with a target roughly 70 cm in diameter. The bright portions of the image contains lobster or crab traps while the darker portions are the backside of small ridges in the sand on the seafloor. Data was collected in the Gulf of America near Panama City, Florida in January of 2006 [1].

4.2 Procedure

The first step in defining a scene to simulate sonar data is to define a coordinate system. The 3-D system that is being built uses a right-handed coordinate system in which the direction of movement of the sonar is the positive x direction, the range direction to the right of the sonar is the positive y direction, and the depth is the positive z direction. The next step in producing this SAS image is to develop a method of defining the seafloor and discretizing it into individual scatterers. To produce a simulated return signal from a single scatterer using the method described in [14], a few key pieces of information are needed: the transmitted signal, the transducer beampattern, the scatterer amplitude, and the locations and normal vectors for the sonar and for the scatterer. The first portion of the code produces these values using the information provided in the image and by the user. The inputs to a function developed for this project, *imageToScattererAmplitude.m*, are the cartesian coordinates of the center of the image on the seafloor, the height and width dimensions of the image on the seafloor, and the image itself. This function outputs arrays which contain the amplitude of each scatterer, the location of each scatterer, and the vector normal to each scatterer. Each pixel in the provided image is used to produce a scatterer. The steps completed in the first portion

of *imageToScattererAmplitude.m* can be visualized in Fig. 4.2. The image used to create scatterers consists of an M by N by 3 array of integers, where M is the number of pixels in height, N is the number of pixels in width, and there are 3 values (red, green, blue) for the color of each pixel. The provided image can be seen in subplot A of Fig. 4.2. The provided image is converted to grayscale to associate a single scalar value corresponding to brightness with each pixel. The grayscale image can be seen in subplot B of Fig. 4.2. Ideally, each scatterer will appear in the sonar simulation, so pixels with a grayscale value of 0 are altered to have a nonzero value which is roughly 4% of maximum. Next, during the grayscale process, lighter colors are associated with higher amplitudes. The grayscale image is inverted because by design, it is preferred that dark colored lettering stands out relative to a light background. The new grayscale value for each pixel is normalized to between 0 and 1. The adjusted grayscale image with these changes can be seen in subplot C of Fig. 4.2. A critical part of synthetic simulations is the division of a scene into scatterers. Infinitely dense facets or scatterers is ideal, but is computationally prohibitive. *imageToScattererAmplitude.m* allows the user to condense variable integer sized arrays of scatterers into a single scatterer to improve run time. For the results shown in this paper, pixels were not condensed from the original image resolution to preserve its clarity. An example condensed and adjusted grayscale image with non-overlapping 4 by 4 arrays of pixels condensed into a single pixel can be seen in subplot D of Fig. 4.2.

The next portion of *imageToScattererAmplitude.m* is converting pixels into scatterers. This involves two steps. First, a position and normal vector is associated with each pixel based on the physical image center and dimensions. After this step, pixels can be considered scatterers. Each scatterer has a normal vector and the normal vectors are perpendicular to the plane on which the scatterers are defined. For the figures in this thesis, the scatterer normal vectors are all in the negative z direction since the scatterers lie on a z-plane. These planar scatterers can be seen in subplot A of Fig. 4.3. Second, a stochastic scale factor is applied to most scatterers. This stochastic scale factor is designed to achieve reasonable experimental mean field, mean square field, and coherence length results [14]. The stochastic scale factor identified in PoSSM is a complex circular Gaussian random variable, $Z = X + iY$, where X and Y are both independently and identically distributed normal random variables. PoSSM performs best when used for environment scatterers with a

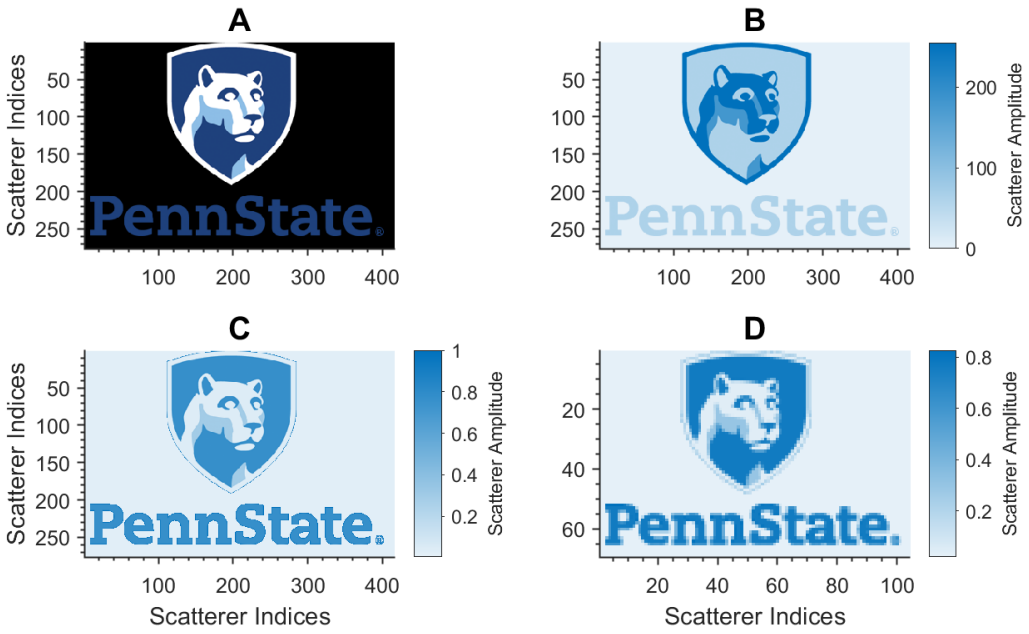

Figure 4.2. Four images displaying the progression of an image as pixels are converted to scatterers. Subplot A shows the user-provided image loaded into MATLAB. Subplot B shows the grayscale of the image. Subplot C shows the image after pixels with grayscale values of zero are removed and the image is inverted. Subplot D shows that pixels can be condensed or interpolated to alter the scatterer density.

coherence length that is less than the resolution of the SAS system. This stochastic scale factor ensures that each scatterer is incoherent with one another and has the effect of displacing the scatterers from their defined plane and gives them a range of amplitudes. For the figures presented in this paper, scatterers which make up the text and logo of the image were considered targets and a stochastic scale factor was not applied to them, while scatterers which made up the background of the image were scaled by a stochastic scale factor since they are considered to be environment scatterers. These new scatterer amplitudes and positions can be seen in subplot B of Fig. 4.3. An important component of ensuring incoherence between environment scatterers is in the phase content of the scatterer amplitude. Randomly distributed phases indicate incoherent scatterers and deterministic phase content indicates coherent scatterers. Subplot C of Fig. 4.3 displays the phase content in the complex plane. The scatterers with only real components and positive scatterer amplitude are the target scatterers, while the scatterers which are normally distributed about the point (0,0) and have a random phase are the environmental scatterers. A

portion of the dimensionless bistatic differential cross section per unit area per unit solid angle, found within the scatterer amplitude in (2) in [14], is defined using the scatterer amplitude developed in this section.

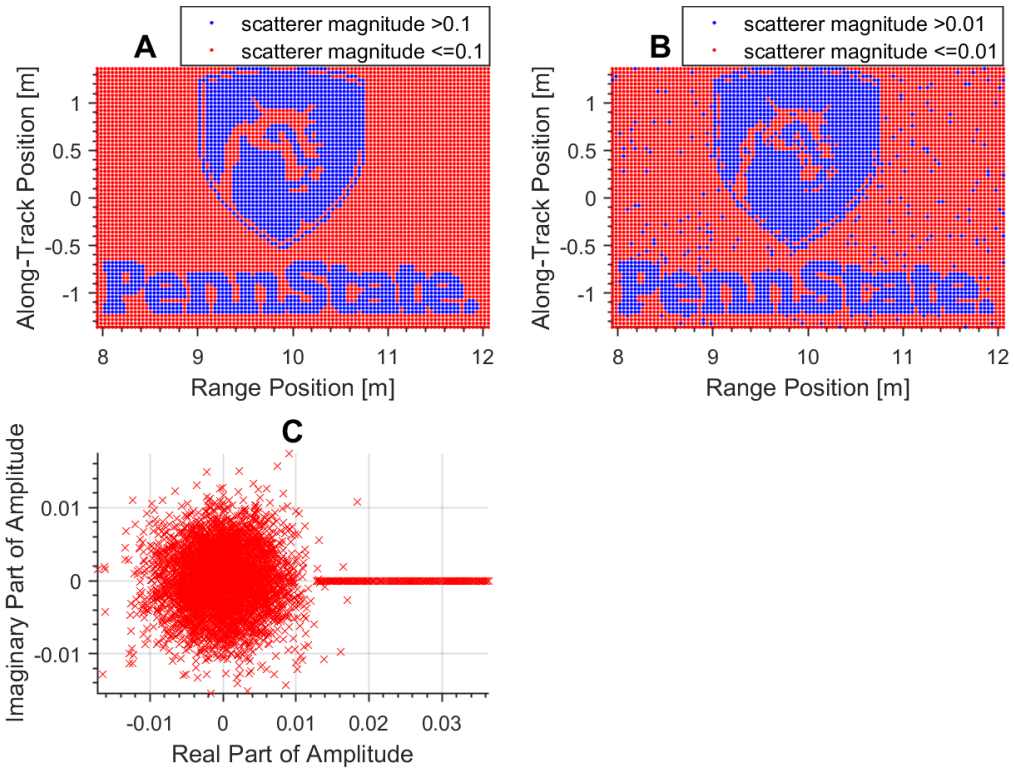

Figure 4.3. Subplot A displays a deterministic component of the scatterers and classifies the scatterers into high and low magnitude scatterers which represent either target (blue) or environment (red) scatterers. Subplot B shows the effect of the application of a stochastic scale factor to the environment scatterers which raises the magnitude of some environmental scatterers above the previous environmental-target magnitude threshold. Subplot C displays the phase content of individual scatterers showing the coherence and incoherence between target and environmental scatterers, respectively.

The second step in producing this SAS image is to define the vehicle's path, position, and orientation through the scene. For this relatively simple model, the vehicle is restricted to move in a straight line along the positive x axis, known as the along-track direction. In this simulation, the vehicle does not move continuously through the scene while transmitting and receiving, but instead transmits and receives at a single location, then travels to the next ping location. One component of defining the vehicle motion through the scene is the vehicle's orientation. The

transducer on the sonar is defined to point at an angle of $\approx 5.7°$ below the x,y-plane[1] and perpendicular to the along-track direction. The orientation of the sonar system does not change with consecutive pings and the vehicle normal vectors create a plane defined by the along-track direction and the vehicle normal. The center of the scatterers is defined to be in-line with this plane. A critical parameter in the setup is the distance between consecutive ping locations. The resolution in the along-track direction of sonar is defined by the distance between elements in an array. For synthetic aperture sonar, the sonar may have a single hydrophone, as it does in this simulation. Then, the distance the sonar and its hydrophone travel between pings is the distance between elements in the synthetic array. The resolution and distance between pings are equal. Minimizing the between-ping distance produces the highest quality simulated data, but increases computational expense. The starting position of the sonar is defined to ensure that the scatterers are entirely out of view during the first ping of the sonar, then appear in view on the following pings before they drift fully out of view on the last ping of the sonar. This setup will produce visible artifacts as scatterers are essentially toggled between 'on' and 'off.' Scatterers are considered to be out of view if the amplitude of the transducer beampattern is less than -3 dB, which creates a cone of influence centered around the normal direction of the vehicle. This setup can be seen in Fig. 4.4.

Additionally, a transmitted signal is defined at each ping location. In this simulation, the transmission signal is the same at each location. A linear frequency modulated (LFM) signal is used for transmission due to its benefits in replica correlation. The transmit signal used in these simulations and its frequency domain representation can be seen in Fig. 4.5. Some desirable characteristics of a transmitted signal are a large amount of energy and a large bandwidth. First, a large amount of energy is desirable since it allows the replica correlated signal to stand out relative to noise (the unwanted portion of the receive waveform). The energy of the signal can be increased by either increasing the amplitude of the transmit waveform or by increasing the period of the pulse of the transmit waveform. However, the period of the pulse is limited by the range of scatterers since transmitting and receiving cannot occur simultaneously on a single transducer. Second, a large bandwidth is desirable for a transmitted signal. Replica correlation

[1]This angle is based on a 10 to 1 range to depth ratio, which is consistent with SAS conventions

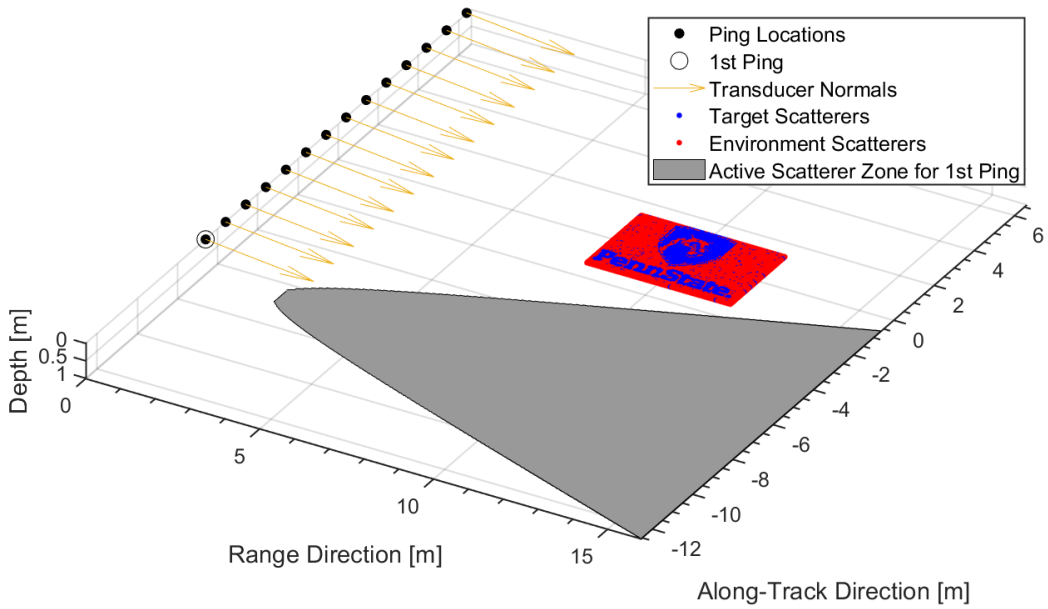

Figure 4.4. A three dimensional figure displaying the scatterer locations and magnitudes, vehicle ping locations, transducer normal vectors, and the parabola of visibility created by the -3 dB restriction on the beampattern.

describes the cross correlation between a receive waveform and its transmitted waveform. During replica correlation, pulse compression occurs and a sinc function is produced which can be used to accurately identify the location of individual scatterers. The width of the sinc functions defines the resolution of the SAS system in the range direction. The width of the pulse compressed replica correlation for a single echo can be defined as the -3 dB level relative to the peak or as the zero to zero width surrounding the peak. Reducing this width is possible by increasing the bandwidth of the transmit waveform. The width of the replica correlated signal varies inversely with the bandwidth, so a greater bandwidth is ideal. However, the maximum theoretical bandwidth of the signal is limited by the sampling rate and a higher sampling rate requires more computations. So, for the figures produced in this paper, the sampling rate was set at 600 kHz. The transmit signal was also windowed using a Chebyshev window and then zero-padded following the transmit signal to ensure that no wraparound occurred in the receive waveform. The period of the pulse is shorter than the smallest vehicle to scatterer travel time to avoid simultaneous transmission and reception. The length of the entire transmit signal, with zero padding, is designed to be equal to the next greatest power of 2 to take

advantage of computational efficiencies in the fast Fourier transform.

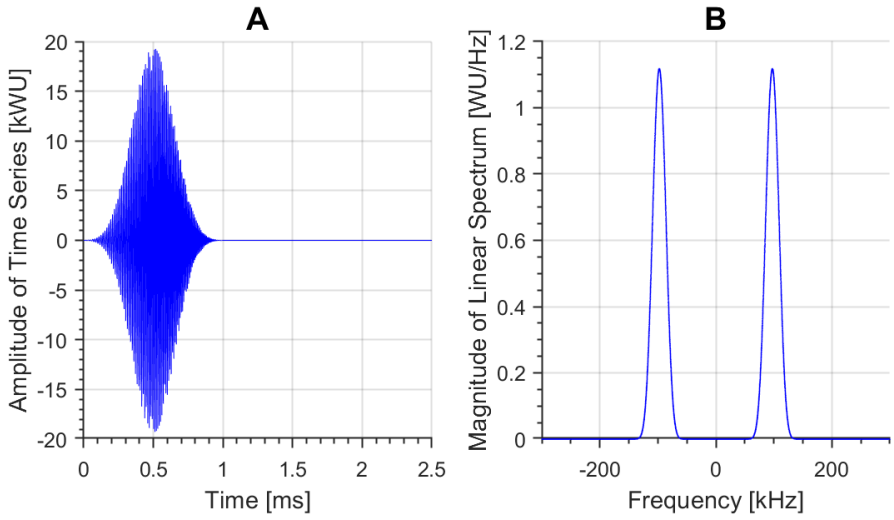

Figure 4.5. The transmitted waveform which is an LFM running from 60 kHz to 135 kHz, sampled at 600 kHz, with a Chebyshev window, followed by zero-padding. This transmit waveform is represented in both the time (subplot A) and frequency (subplot B) domains.

The next step in calculating the scatterer amplitude is utilizing the defined vectors to calculate important angles. The angles that are considered important include the transmit angle and the incident angle, which is the complement of the conventional grazing angle. The angles are calculated in *findGeometry.m*. These vectors and angles can be seen in Fig. 4.6.

The $\theta_{incident}$ angle is used to calculate Lambertian scattering. Lambertian scattering is an ideal model of scattering which states that the absorbed and reflected amplitudes of an incident signal both vary with the cosine of $\theta_{incident}$, meaning that the reflected signal scales the incident signal by $cos^2(\theta_{incident})$. This Lambertian scattering factor is applied by scaling the previously developed scattering amplitude. The $\theta_{transmit}$ angle is used to calculate the amplitude of the signal transmitted and received by the transducer according to its beampattern. The beampattern used for this simulation was sourced from *Principles of Underwater Sound* [8] and is designed for a disk array of transducers with a specific diameter. This is an approximation for the single transducer of a specific diameter defined for this model. In reality, the transducer beampattern is omnidirectional and would interact with any scatterer that has a $\theta_{transmit}$ less or equal to 90°. However, for computational efficiency, if the value of the beampattern is less than -3 dB for a scatterer, that scatterer is

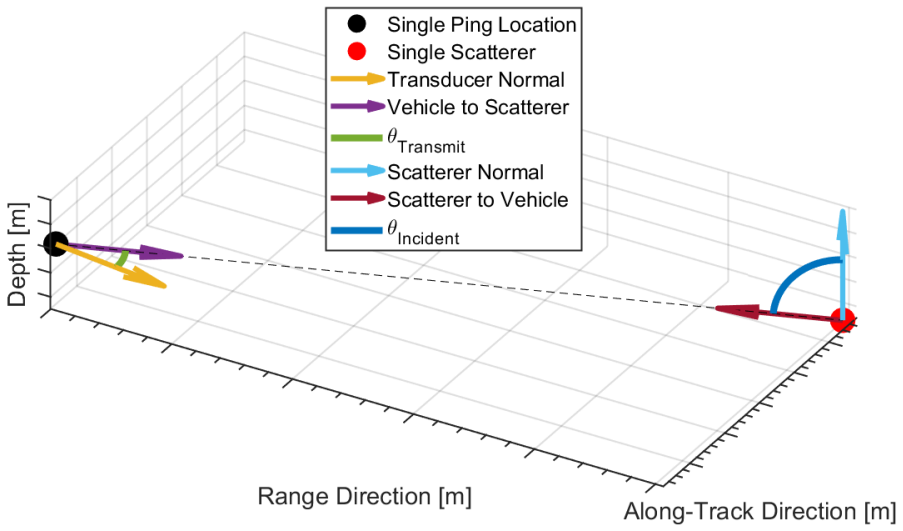

Figure 4.6. A three dimensional plot displaying a single vehicle ping location and a single scatterer along with the transducer normal, the scatterer normal, and vectors connecting the vehicle and scatterer. These vectors and coordinates are used to calculate angles which are needed to alter the transmit waveform.

considered to be invisible and the scattered waveform it emits is not considered due to the significantly reduced amplitude. The amplitude of the scattered waveform from any invisible scatterers is ultimately reduced by at least 6 dB because the effect of the beampattern is applied upon both transmission and reception. The beampattern used for this simulation can be seen in Fig. 4.7.

Limiting the considered scatterers to only include those with a beampattern greater than -3 dB places a restriction on $\theta_{transmit}$. Values greater than this limit are excluded and values smaller than this limit are included. This creates a cone of influence where the reflected signal from scatterers are considered. All scatterers were defined to be in a single plane and the intersection of that plane with the cone of influence creates a parabola. This parabola of influence is shown in Fig. 4.4.

The next step in producing simulated sonar data is to continue applying fundamental physical effects to the transmitted linear spectrum using the scene infrastructure which has been developed. The two most important physical effects which are included in this simulation are spherical spreading and a time delay. Spherical spreading is a simple scale factor which is applied to the entire linear spectrum

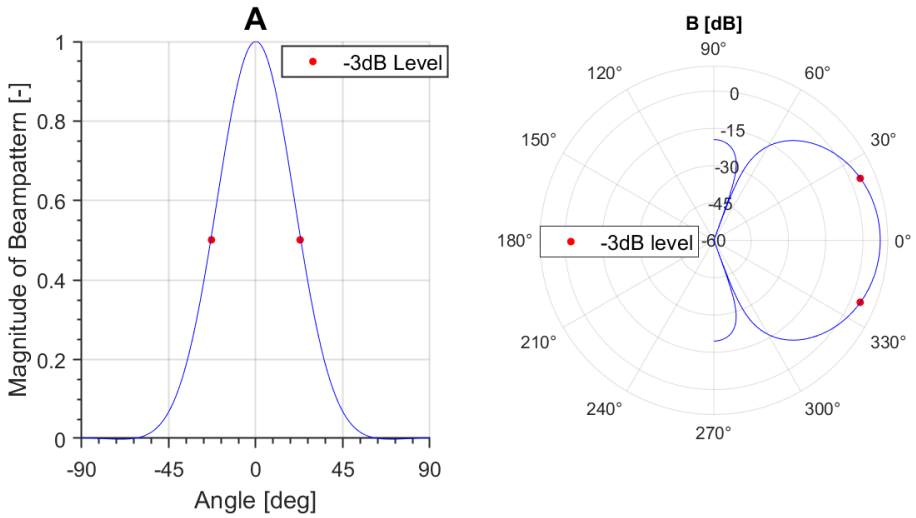

Figure 4.7. Plots displaying the beampattern sourced from *Principles of Underwater Sound* for a circular planar array of transducers with a specified diameter. Subplot A displays the absolute magnitude modification to transmitted and received signals. Subplot B displays the same information in decibels. The -3 dB levels on each plot are marked by red dots.

and is based on the distance between the vehicle and any scatterer. The result of spherical spreading and a time delay for a single scatterer can be seen in subplot A of Fig. 4.8.

Spherical spreading is a scalar which is easily applied to the transmit signal. The application of the time delay is why the transmit signal is modified in the frequency domain rather than in the time domain. In the time domain, a time delay is a shift by a certain number of indices. The time shift is equal to $N \times \Delta t$, if N is the number of samples shifted and Δt is the amount of time between consecutive samples. However, when shifting in the time domain, the user is restricted to either a time delaying by an integer number of samples, or by a non-integer number of samples combined with interpolation. The benefit of the frequency domain is that a non-integer time delay can be applied and ideal Fourier interpolation will be automatically implemented. Additionally, because conversion between a time series and linear spectrum is a linear operation, the frequency domain transmit signal can be modified by physical effects prior to converting it back into the time domain. The Fourier pair of convolution and multiplication can be utilized to selectively apply physical effects in either domain to minimize required computations. Additionally, the return signal from individual scatterers can be summed prior to conversion back

into the time domain to reduce the frequency with which the inverse fast Fourier transform needs to be used.

With the now developed components including a transmit signal, a beampattern, a scatterer amplitude (which combines scatterer density, a Lambertian scattering factor, pixel color, and a stochastic scale factor), a time delay, and spherical spreading, this acoustic propagation and scattering model can produce basic results. At a single ping position, this model calculates the return signal from each individual scatterer. The return from a single scatterer is subplot A of Fig. 4.8. After the return signal from each scatterer is calculated, these return signals are summed to produce the overall return signal at a single ping position. The return signal from two scatterers and from 500 scatterers are seen in subplots B and C, respectively, of Fig. 4.8. The simulation was run without any white noise added to the return signal. The time delay and decreased amplitude resulting from spherical spreading are both evident in Fig. 4.8. Additionally, in subplot B, there is destructive interference causing a beat effect which may not have intuitive features due to the under sampling of the transmit signal.

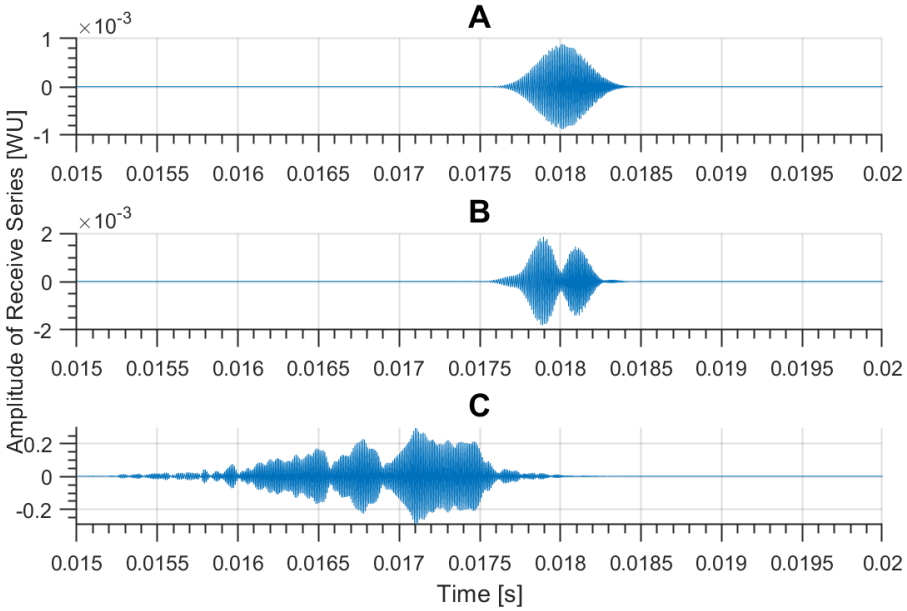

Figure 4.8. Time domain receive waveform from 1, 2, and 500 scatterers are found in subplots A, B, and C, respectively.

After this synthetic receive data has been produced, post processing can be

performed to create a beamformed SAS image from the raw time series. The first step is replica correlation with the transmit waveform. The result of this replica correlation is pulse compression and a greatly improved resolution, as seen in Fig. 4.9. This replica correlation can be done in either the time or frequency domain, but it is significantly faster in the frequency domain where multiplication between linear spectra can be used instead of convolution of the transmitted and receive series. After replica correlation is performed, the replica correlated receive signals are converted to the time domain. The next postprocessing step is to prepare the data for the wavenumber beamformer. To do this, the replica correlated receive series is demodulated. Demodulation shifts the frequency content of a signal to be centered at zero and is most efficiently applied in the time domain. Then, time varying gain can be applied to the receive data to counteract the effects of spherical spreading, which will also have the consequence of increasing the noise at greater ranges. The demodulated, replica correlated receive signal with time varying gain is the final product which can be input into the wavenumber beamformer.

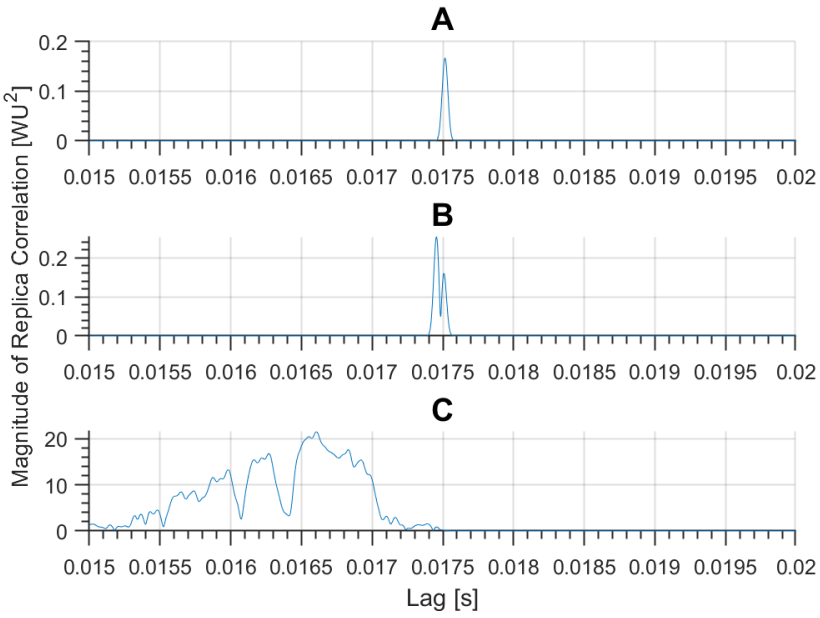

Figure 4.9. Time domain replica correlated receive signals from 1, 2, and 500 scatterers are found in subplots A, B, and C, respectively.

The returns at each individual ping position can then be visualized in a 2-D plot which shows the sonar smile, found in Fig. 4.10. Each row of this figure

is a single replica correlated synthetic receive signal. The color associated with each pixel indicates the amplitude at a specific receive range. The receive range was calculated from receive time by assuming a constant speed of sound in water, $c = 1500m/s$, and a simple $v = \frac{2 \times d}{\Delta t}$, where the factor of 2 is included due to the round-trip propagation. There appear to be two wavefronts in Fig. 4.10. The first wavefront is likely due to the text found in the image. The second wavefront is likely present due to the mascot found in the image.

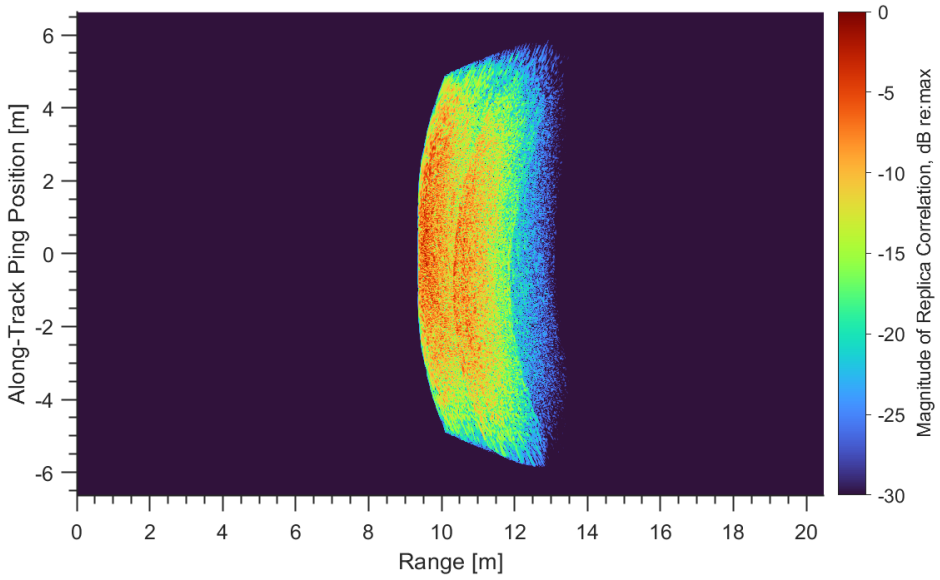

Figure 4.10. A representation of the synthetic replica correlated receive signals at each ping and listen position. The ping locations are along the vertical axis and the calculated receive range is along the horizontal axis. The color of each pixel corresponds to the magnitude of the signal received at each position from each range.

Chapter 5
Results and Discussion

5.1 Results

After synthetic sonar data has been produced, it can be visualized using a wavenumber beamformer. The output of the wavenumber beamformer for the synthetic data, which was produced using the described experimental procedures, from a few different trials can be seen in Fig. 5.1. Across each trial, the sample rate was 600 kHz, there was no compression of pixels (416 x 276), no noise was added to the simulation, and the pixels in the lettering and logo were converted to target scatterers, while the pixels in the background were converted to environment scatterers. The environmental scatterers received a few special modifications to avoid simulating a perfectly organized grid. Each method took approximately 6 hours of run time, with some slight variations, using the parallel computing toolbox developed by MATLAB with 8 workers on a laptop with an 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50 GHz. The first trial, seen in subplot A of Fig. 5.1 was run with a coherent scatterer setup. In this case, the stochastic scale factor was $Z = 1$ instead of $Z = X + iY$. This has the effect of modeling the scatterers exactly as designed in an array on a plane, which produces an image with a mirror-like appearance. The second trial, seen in subplot B of Fig. 5.1, used the traditional stochastic scale factor proposed in [14], where $Z = X + iY$ with X and Y independently and identically distributed as $\mathcal{N}(0, 1)$. This has the effect of ensuring each environment scatterer is incoherent with any other, which randomizes the appearance of the individual scatterers as intended. The third trial, seen in subplot C of Fig. 5.1, used a similar method, but substituted the stochastic scale factor for $Z = exp(iX)$, where $X = \mathcal{N}(0, 1)$.

This has the effect of keeping the randomized phase component between individual scatterers which keeps each scatterer incoherent from one another. However, it doesn't affect the amplitude of the scatterers which could have potentially been made negative by the $Z = X + iY$ stochastic scale factor. From visual inspection of the wavenumber results, this alteration of the complex circular Gaussian scale factor produces reasonable results which barely differ from the method used in subplot B. The use of an exponential is computationally expensive, but because this is done once for each scatterer, it is manageable for the trialed setup. The fourth trial, seen in subplot D of Fig. 5.1, was produced using a stochastic scale factor of $Z = 1$. However, after the position of each scatterer was defined in a grid along a plane, a Gaussian random variable, $\mathcal{N}(0, 0.015)$, was added to the z-coordinate of each scatterer in order to randomize each position and ensure that each scatterer is incoherent from one another using a different method. As can be seen through visual inspection, this trial produces similar results compared to both the complex circular Gaussian scale factor method and the random phase with magnitude 1 method.

One method of quantifying the quality of the image produced is using the contrast ratio, which is defined in (11) of [24]. The contrast ratio, approximately, is the ratio of the scattered sound and noise in a region of relatively brights scatterers (higher scatterer magnitude) to the scatterer sound and noise in a region of relatively dim scatterers (lower scatterer magnitude which can be approximated as zero for simplicity). This ratio describes the contrast between target scatterer regions and shadowed regions and a large difference in these regions indicates that an image is of high quality. This ratio can be calculated in decibels and, as mentioned in [24], a contrast ratio of 10 dB or greater is an indicator of a quality image. One method of determining the contrast ratio of this simulation is to run the simulation with a single scatterer. Ideally, only a single scatterer would appear in the image, but instead, there are artifacts resulting from noise in the simulation and in the beamformer. To evaluate the level of undesired noise, the contrast ratio can be approximated by taking the ratio of the scatterer magnitude at its main peak (where a scatterer is expected) to the scatterer magnitude at the greatest side lobe (where a scatterer is not expected). A realistic simulation should not have any artifacts from simulation which have an amplitude that is greater than any real-world effects that have been simulated. In the developed model, for a single

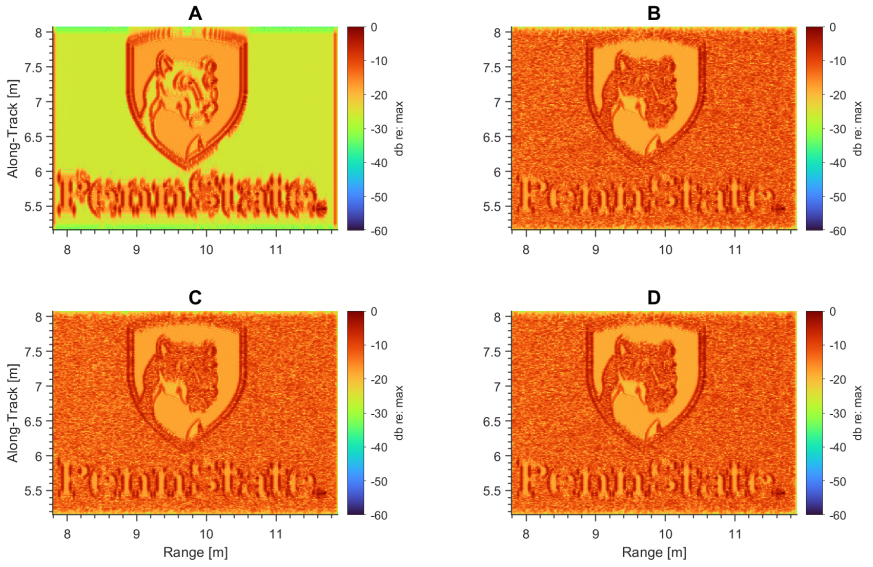

Figure 5.1. The wavenumber beamforming output of 4 trials which used different methods of producing coherent or incoherent scatterers. Method 1 in subplot A used a stochastic scale factor of $Z = 1$ which resulted in coherent scatterers. Method 2 in subplot B used a complex circular Gaussian stochastic scale factor. Method 3 in subplot C used a stochastic scale factor of $Z = exp(iX)$. Method 4 in subplot D used a stochastic scale factor of $Z = 1$, but added a random variation of $\mathcal{N}(0, 0.015)$ to the z component of the position of each scatterer.

scatterer, the contrast ratio was calculated to be approximately 49 dB, as seen in Fig. 5.2. For the simulated images found in subplot B of Fig. 5.1, the contrast ratio was calculated to be approximately 20 dB. This latter contrast ratio indicates that the model simulates the desired real-world scene without being dominated by artificial simulation artifacts.

5.2 Discussion

The primary variable adjusted to test the relationship between quality of simulation and computations required was the number of scatterers. The scatterers were defined to populate an area that was 2.76 m x 4.16 m. This setup was trialed with 1 to 1, 16 to 1, and 192 to 1 scatterer compression, which resulted in 114816, 7176, and 598 scatterers for each simulation, respectively. With 114816, 7176, and 598 scatterers, the model required approximately 6 hours, 20 minutes, and

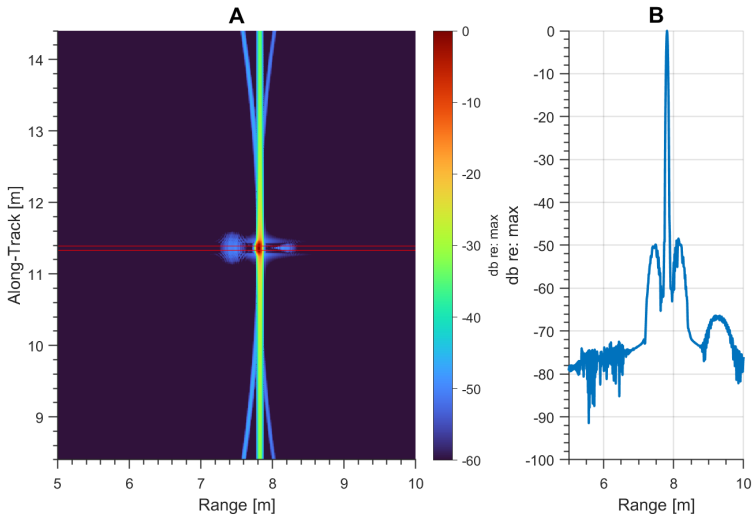

Figure 5.2. Subplot A displays the results of a wavenumber beamformer acting on synthetic data from a single scatterer produced using the derived model. Subplot B displays a horizontal cross section of subplot A, which can be used to calculate the contrast ratio and quantify the image quality.

less than 2 minutes of run time, respectively. The speckle effect seen in Fig. 5.1 could only be seen in the simulation with 114816 scatterers. With 7176 and 598 scatterers, individual scatterers were visible which indicates the data produced for those simulations is not realistic. The speckle effect was visible in Fig. 5.1 because the spatial resolution of the simulation was less than the defined spatial frequency of the scatterers. A simple adjustment to produce valuable results even with fewer scatterers is to reduce the defined dimensions of the scatterer area and ensure the scatterers are more spatially frequency than the resolution of the system. Qualitatively, these results can be compared to the results in [1] and considering the relatively coarse scatterer density and lack of advanced computational power, these results show agreement to real-world data and display the effectiveness of the developed model. Quantitatively, a contrast ratio of 20 dB for a simulated image also indicates that the quality of the produced image is good and that it is not overpowered by simulation artifacts. A good contrast ratio indicates that the SAS image produced using the derived model is quality and could be used as a substitute for real-world data under controlled and explicit training situations. This use of the data could yield some of the advantages described in the introduction.

Chapter 6
Summary and Conclusions

6.1 Summary

This thesis describes a project in which a basic version of an acoustic scattering and propagation model was implemented, based on [14] and using ideas presented throughout the literature review. The development of this model was done entirely through MATLAB code and broadly includes components of physics, signal processing, and statistics. The model developed includes infrastructure to convert any image to a planar rectangle of scatterers located in a 3-D scene. The model modifies the transmit signal using basic physical effects such as transmit and receive directivity functions, spherical spreading, a time delay, and a scatterer amplitude with both stochastic and deterministic components. This model includes the basic components of a scattering and propagation model. The primary utility of the model is allowing users to include new implementations of physical effects using various methods. The effect of including new physics in the model can be evaluated using at least the two metrics presented, image quality and run time. Using a laptop with an 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50 GHz, multiple simulations were performed which demonstrated the important relationship between computations and data quality. While simulations with low quality or few scatterers ran at a speed that was convenient for developing the model and testing code, the results produced were generally not realistic. Simulations with many scatterers had durations of approximately 6 hours and had acceptable image qualities and contrast ratios, which proved that the model was working as intended, but required significant computational power. The presence of a relationship between fidelity of

results and run time, both of which depend on the amount of detail included in the model, was the goal of the development of the presented model.

6.2 Conclusion

A basic acoustic scattering and propagation model has been developed and trialed on an image and different patterns of scatterers. The model developed can be used as a starting point for introducing physics into existing SAS models. Versions of the model with new physical effects toggled on and off can be compared qualitatively by comparing wavenumber beamformed images and quantitatively using the contrast ratio to achieve agreement with real-world results. Including more physical effects will improve model performance, but is not always pragmatic. Being able to evaluate the tradeoff between model performance and computations allows the models to be utilized as a specialized development and training tool.

Chapter 7
Future Work

Due to the considerable simplifications used to create this model, the amount of work that could be done to improve it is nearly endless. This model can contribute in a small way toward supporting the development of existing models which provide the economic and safety benefits mentioned in the introduction. Improvements in this model will make it more useful and will improve its qualification for being used for tasks similar to those of the models described in the literature review.

Potential improvements to the current model:

- An environment outside of the image could be added so that the logo is not floating in space
- The scene design could be expanded to produce more than strictly planar features; alternatively, a series of planar rectangles at varying angles and of various sizes could be used to construct detailed scenes
- A more sophisticated method of calculating the scatterer normal vector which is based on neighboring scatterer positions could be developed
- The velocity of the sonar could be included to allow the model to run in real time, to place constraints on resolution and range, and to model the Doppler effect
- Analytic scattering models for specific target shapes could be incorporated, while using the current PoSSM-based model to simulate the background filled with environment scatterers
- Both multiple scattering and shadowed scatterer effects could be modeled

- Non-constant velocity propagation of transmit signals could be included, potentially through an external model
- Multiple receivers could be included in the model
- Material properties and reflection coefficients could be included in lieu of colors and Lambertian scattering, respectively
- The narrowband assumptions could be removed

Experiments to try with the current model to more thoroughly test the effect of altering assumptions and variables:

- Try the simulation with a different: beampattern, window for the transmit signal, image, type of noise, scene geometry, etc.
- Determine how resolution is affected by changing the: size of the transducer, ping spacing, and bandwidth
- Place the center of the scatterers outside of the plane created by the transducer normal vectors so that the scatterers are not fully in view throughout the entire simulation
- Analyze the phase of the scatterer amplitude produced using wavenumber beamforming as opposed to only the magnitude
- Determine why there is an apparent repeated 'image' in the replica correlated synthetic data for simulations run with large scatterer spacings

Alterations to improve the speed of the simulation:

- Convert the current main script into a function with a clear set of inputs and outputs (likely structs) to improve simulation testing, allowing the user to test in sequence with varying settings without requiring manual intervention
- Set up MATLAB's GPU array toolbox to potentially improve speed
- Use function handles instead of separate functions in the main for loop

Appendix A
Code for Scattering and Propagation Model

The code developed for this thesis consists of a main script and a few supporting functions which are called in that script. The function *wk.m* is code for an omega-k beamformer and can be provided from an external source. Additionally, functions which I developed for a signal processing course are called throughout this code with descriptive names of the form *customFUNCTIONNAME.m*. These functions are not provided and must be developed or replaced if this code is to be run.

A.1 Main Script

modelSAS.m

```
clear;clc;close all;
tic;

%User options occur before the parfor loop
showPlots = true; %To see all plots, run this script
   without a parfor loop
scattererPattern = 'image';%'.', 'x', or 'image'
incoherenceMethod = 'possm';%'possm', 'possmMag1', '
   zRand', 'mirror'
```

```
8  %Set equal to 1 if all scatterers are environment; set
       equal to 0 if all scatterers are targets;
9  environmentThreshold = 0.1;
10 noiseAmp = 0; %Amplitude of noise
11 dBCutoff = true; %Decide whetheror not to use a cutoff
       of -3dB
12 filterOutput = false; %Decide whether to bandpass filter
        the output according to the transmit LFM
13 applyTVG = false;
14
15 %Much needed constants
16 fs = 600e3;
17 dt = 1/fs;
18 c = 1500;
19
20 %Define the relevant coordinates and amplitude based on
       an image for the scatterer(s).
21 %scattererCoords = [x,y,z] is a column matrix.
22 imageName = "ScattererImages\PSULogoShield.png";
23 imageDimensions = [2.76, 4.16];%./4 % height(x drxn);
       width (y drxn)
24 desiredImageCenter = [0,10,1]; %[along-track, range,
       depth] %This is typically a 10:1 range:depth ratio
25 scattererCondensing = [1,1];%[4,4];%[12,16];
26 imageBtmLftCorner = [desiredImageCenter(1)-
       imageDimensions(1)/2, desiredImageCenter(2)-...
27      imageDimensions(2)/2, desiredImageCenter(3)];
28 [allScattererCoords,allScattererNormals,
       allScattererPartialAmplitudes] =
       imageToScattererAmplitude( ...
29      imageName,imageBtmLftCorner,imageDimensions,
           scattererCondensing,incoherenceMethod, ...
30      environmentThreshold,showPlots);
31
```

```
32
33 scatDesignIdx = false(size(allScattererPartialAmplitudes
   ));
34 switch scattererPattern
35     case 'image'
36         scatDesignIdx = ~scatDesignIdx;
37     case 'x' %This must be used with [12,16] scatterer
       condensing
38         if scattererCondensing~=[12,16]
39             error("The X pattern must be used with
                  [12,16] scatterer condensing.")
40         end
41         scatDesignIdx(1:24:24*23) = true;
42         scatDesignIdx(23:22:24*22) = true;
43     case '.'
44         scatDesignIdx(1) = true;
45 end
46 allScattererCoords = allScattererCoords(scatDesignIdx,:)
   ;
47 allScattererNormals = allScattererNormals(scatDesignIdx
   ,:);
48 allScattererPartialAmplitudes =
   allScattererPartialAmplitudes(scatDesignIdx,:);
49
50 %Calculates the resolution in the range direction
51 desiredResolutionY = 1/100;
52 bandwidth = 1/(2*desiredResolutionY/c);
53
54 %Defines transmit signal properties
55 f1 = 60e3; %Transmit LFM starting frequency
56 f2 = f1 + bandwidth; %Transmit LFM Ending Frequency
57 Tp = 1e-3;%in seconds
58 A = 1e4; %Amplitude of transmit signal
```

```
59 fCenter = (f1+f2)/2; %Transmit LFM center/carrier
       frequency
60 lambdaCenter = c/fCenter;
61 %[bb, aa] = customFilterButterworthBP(5, f1, f2, fs);
62
63 %Calculates the resolution in the along-track direction
64 desiredResolutionX = 1/100;
65 physicalApertureDiameter = 2*desiredResolutionX;
66     %Angle is defined as beamwidth/2. This is dependent
          on the chosen beampattern
67 transducerMaxAngleRad = 0.52*(lambdaCenter/
       physicalApertureDiameter);
68
69
70 %Define the relevant coordinates here for the moving '
       monostatic'
71 %vehicle. X is vehicle travel direction. Y is the range
       direction. Z=0 is
72 %the sonar, Z=+z is the scatterer plane depth
73 distanceBetweenPings = physicalApertureDiameter/2;
74 [vehicleXCoordStart] = findVehiclePingOffset(
       allScattererCoords,transducerMaxAngleRad,"beginning")
       ;
75 [vehicleXCoordEnd] = findVehiclePingOffset(
       allScattererCoords,transducerMaxAngleRad,"end");
76 vehicleXCoords = (vehicleXCoordStart:
       distanceBetweenPings:vehicleXCoordEnd)';
77 numPings = length(vehicleXCoords);
78 vehicleYCoords = linspace(0,0,numPings)';
79 vehicleZCoords = linspace(0,0,numPings)';
80 vehicleCoords = [vehicleXCoords,vehicleYCoords,
       vehicleZCoords];
81 vehicleNormals = ones(numPings,3).*desiredImageCenter./
       sqrt(sum(desiredImageCenter.^2));
```

```
82
83 %Finds the maximum time delay between the vehicle and
       the scatterer at any point
84 extremeDistances = [allScattererCoords-vehicleCoords
       (1,:); allScattererCoords-vehicleCoords(end,:)];
85 maxDistance = max( sqrt( sum(extremeDistances.^2, 2) ) )
       ;
86 maxTimeDelay = 2*maxDistance/c;
87 maxTimeDelayNPts = ceil(maxTimeDelay*fs);
88
89 %Creates the transmit signal
90 tmSrsChirp = A.*chirp( ((0:Tp*fs-1)'.*1/fs) ,f1,Tp,f2,"
       linear",-pi/2);
91 [windowedChirp, ~] = customApplyWindowToTS(tmSrsChirp, "
       Chebyshev", 1);
92 transmitTSDesiredLength = 2^nextpow2( maxTimeDelayNPts+
       length(windowedChirp) );
93 transmitTS = [windowedChirp; zeros( (
       transmitTSDesiredLength-length(windowedChirp)), 1)];
94 timeVec = (0:transmitTSDesiredLength-1)'./fs;
95 demodulatingSeries = exp(-1j*2*pi*fCenter*timeVec);
96 transmitHilbert = customHilbertTransform(transmitTS,fs);
97 kernelSeries = transmitHilbert.*demodulatingSeries;
98
99 %Converts the time series to a linear spectrum
100 N = length(transmitTS);
101 XFreqs = ((-N/2+1):N/2).'.*(fs/N);
102 XFreqs = [XFreqs(N/2:N);XFreqs(1:N/2-1)];
103 XOmegas = 2*pi.*XFreqs; %XOmegas is 0 to fs/2 to -fs/2
       to 0
104 transmitLS = fft(transmitTS).*dt; %transmitLS is 0 to fs
105
106 %For visualizing the transmit signal
107 if showPlots == true
```

```
108     fig = figure(Name="Transmit Time Series and Linear
            Spectrum");
109     subplot(1,2,1)
110     plot(timeVec,transmitTS,'b-',LineWidth=0.5);xlim([0
            2.5*Tp])
111     xlabel("Time [s]");
112     ylabel("Amplitude of Time Series[WU]");
113     title("A");
114     grid on;
115     subplot(1,2,2)
116     plot(XFreqs,abs(transmitLS),'b.',MarkerSize=1)
117     xlabel("Frequency [Hz]");
118     ylabel("Magnitude of Linear Spectrum [WU/Hz]");
119     title("B");
120     grid on;
121     fig.Position = [200 100 800 400];
122 end
123
124 %Pre-allocates some helpful indices and matrices
125 idx.oneToN = 1:N;
126 idx.nOver2toN = (N/2):N;
127 idx.oneToNOver2Minus1 = 1:(N/2-1);
128 idx.twoToNOver2 = 2:(N/2);
129 idx.nOver2Plus2ToN = (N/2+2):N;
130 idx.Nyquist = N/2+1;
131 masterReceiveLS = zeros(N,numPings);
132 masterReceiveTS = zeros(N,numPings);
133 masterReceiveRC = zeros(N,numPings);
134 masterReceiveTSAnalytic = zeros(N,numPings);
135 noiseMatLS = fft(noiseAmp.*randn(N,numPings),[],1).*dt;
136 doneTS = false(1,3);
137 doneRC = false(1,3);
138
139 %%
```

```
140 parfor iPing = 1:numPings
141     %Finds the distances between the vehicle and each
          scatterer
142     [allVehicleScattererDistances,allThetaTransmits,
          allThetaIncidents] = findGeometry( ...
143         vehicleCoords(iPing,:),vehicleNormals(iPing,:),
              allScattererCoords, ...
144         allScattererNormals,showPlots,iPing);
145
146     %Identify which scatterers can be insonified (think
          of a flashlight illuminating something)
147     if dBCutoff == true
148         activeIndices = allThetaTransmits <=
              transducerMaxAngleRad;
149     else
150         activeIndices = true(size(allThetaTransmits));
151     end
152
153     numActiveScatterers = sum(activeIndices);
154     %Now, reduce the scatterers to only be the active
          ones
155     vehicleScattererDistances =
          allVehicleScattererDistances(activeIndices);
156     thetaTransmits = allThetaTransmits(activeIndices);
157     thetaIncidents = allThetaIncidents(activeIndices);
158     scattererPartialAmplitudes =
          allScattererPartialAmplitudes(activeIndices);
159
160     %Uses Lambertian scattering to calculate a portion
          of the scatterer amplitude
161     scattererAmplitudes = (cos(thetaIncidents).^2).*
          scattererPartialAmplitudes;
162     TRBeampattern = ( 2*besselj(1, (pi*
          physicalApertureDiameter/(c/fCenter)).*sin(
```

```
        thetaTransmits))...
163                     ./(pi*physicalApertureDiameter/(c/
                           fCenter).*sin(thetaTransmits)) )
                           .^2;
164
165     %Uses spherical spreading to calculate spreading
           both ways
166     sphericalSpreadingScaleFactor =
           vehicleScattererDistances.^(-2);
167
168     %Finds the time delay at each point
169     taus = (2*vehicleScattererDistances)/c;
170
171     receiveLSSum = noiseMatLS(:,iPing);
172     for iScat = 1:numActiveScatterers
173         %Applies the time delay to the transmitted
               signal
174         XDelay = exp(-1j.*XOmegas.*taus(iScat));
175         XDelay(idx.Nyquist) = 0;
176         receiveLSCurrent = transmitLS.*XDelay;
177
178         %Apply the transmit and receive beampatterns
179         receiveLSCurrent = receiveLSCurrent.*
               TRBeampattern(iScat);
180
181         %Apply spherical spreading both ways, assumes
               monostatic
182         receiveLSCurrent = receiveLSCurrent.*
               sphericalSpreadingScaleFactor(iScat);
183
184         receiveLSCurrent = [...
185             receiveLSCurrent(1); scattererAmplitudes(
                   iScat).*receiveLSCurrent(idx.twoToNOver2)
                   ;...
```

```
186             receiveLSCurrent(idx.Nyquist);...
187             conj(scattererAmplitudes(iScat)).*
                   receiveLSCurrent(idx.nOver2Plus2ToN)];
188
189         receiveLSSum = receiveLSSum + receiveLSCurrent;
190     end
191
192     masterReceiveLS(:,iPing) = receiveLSSum;
193
194     %Converts the receiveLS to receiveTS in the time
           domain
195     masterReceiveTS(:,iPing) = ifft(masterReceiveLS(:,
           iPing)).*fs;
196
197     %Bandpass filters the receive signal
198     if filterOutput == true
199         masterReceiveTS(:,iPing) = filter(bb, aa,
               masterReceiveTS(:,iPing));
200     end
201
202     %Applies the Hilbert Transform
203     [masterReceiveTSAnalytic(:,iPing),~,~,~] =
           customHilbertTransform(masterReceiveTS(:,iPing),
           fs);
204
205     %Demodulates the Hilbert Transformed signals
206     masterReceiveTSDemodulated(:,iPing) =
           masterReceiveTSAnalytic(:,iPing).*
           demodulatingSeries;
207
208    %Calculates the replica correlation, without
         normalization
209     [masterReceiveRC(:,iPing),lagVector] = customTStoCC(
           ...
```

```
210         kernelSeries,masterReceiveTSDemodulated(:,iPing)
                ,dt,"Circular","0toT",0);
211
212     fprintf(['Ping ', num2str(iPing), ' of ', num2str(
            numPings), ' is done\n'])
213
214
215     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
216     %Plotting subsection
217     if showPlots == true
218     % Visualize the transmit cone of influence:
219         if iPing == 1
220             dx=100;
221             dxScat = ceil(dx/4);
222             fig = figure(Name="Transducer -3dB
                    Beampattern in Scene");
223             hold on;
224             set(gca,'zdir','reverse');
225             set(gca, 'xdir','reverse');
226             grid on;
227             axis image;
228             view(-45,45);
229             plot3(vehicleCoords(1:dx:end,1),
                    vehicleCoords(1:dx:end,2),vehicleCoords
                    (1:dx:end,3),'k.',...
230                   MarkerSize=20,DisplayName="Ping
                        Locations")
231             plot3(vehicleCoords(1,1),vehicleCoords(1,2),
                    vehicleCoords(1,3),'ko',MarkerSize=10,
                    DisplayName="1st Ping")
232             quiver3(vehicleCoords(1:dx:end,1),
                    vehicleCoords(1:dx:end,2),vehicleCoords
                    (1:dx:end,3),...
```

```
233                 vehicleNormals((1:dx:end),1),
                       vehicleNormals((1:dx:end),2),
                       vehicleNormals((1:dx:end),3),...
234                 DisplayName="Transducer Normals")
235             plotIdx = abs(allScattererPartialAmplitudes)
                    >= environmentThreshold/10;
236             plot3(allScattererCoords(plotIdx,1),
                   allScattererCoords(plotIdx,2),
                   allScattererCoords(plotIdx,3),'b.',...
237                 MarkerSize=10,DisplayName="Target
                       Scatterers");
238             plot3(allScattererCoords(~plotIdx,1),
                   allScattererCoords(~plotIdx,2),
                   allScattererCoords(~plotIdx,3),'r.',...
239                 MarkerSize=10,DisplayName="Environment
                       Scatterers");
240             [X,Y,Z] = cylinder(0:0.01:1,20000);
241             X = 2.*norm(desiredImageCenter).*tan(
                   transducerMaxAngleRad).*X;
242             Y=2.*norm(desiredImageCenter).*tan(
                   transducerMaxAngleRad).*Y;
243             Z=2.*norm(desiredImageCenter).*Z;
244             X=X+vehicleCoords(1,1);cylPts = [reshape(X,[
                   numel(X),1]), reshape(Y,[numel(Y),1]),
                   reshape(Z,[numel(Z),1])].';
245             rotXMat = rotx(-atand(desiredImageCenter(2)/
                   desiredImageCenter(3)));
246             cylPtsRot = rotXMat*cylPts;
247             plotIdx = cylPtsRot(3,:)>desiredImageCenter
                   (3)*0.99 & cylPtsRot(3,:)<
                   desiredImageCenter(3)*1.01;
248             fill3(cylPtsRot(1,plotIdx),cylPtsRot(2,
                   plotIdx),cylPtsRot(3,plotIdx),[0.61 0.61
                   0.61],...
```

```
249                 DisplayName="Active Scatterer Zone for 1
                       st Ping");
250             xlabel("Along-Track Direction [m]");
251             ylabel("Range Direction [m]");
252             zlabel("Depth [m]");
253             legend(Location='best');
254             xlim([-2*max(vehicleCoords(:,1)) 1.2*max(
                   vehicleCoords(:,1))]);
255             ylim([0 1.5*max(allScattererCoords(:,2))]);
256             zlim([min(vehicleCoords(:,3))-2, max(
                   allScattererCoords(:,3))])
257             set(gca, 'zdir','reverse');
258             set(gca, 'xdir','reverse');
259             grid on;
260             legend();
261             axis image;
262             view(120,30);
263             fig.Position = [200 100 850 500];
264         end
265         %Visualize the beampattern
266         if iPing == 1
267             fig = figure(Name="Beampattern Plots");
268
269             subplot(1,2,1)
270             theta = -pi/2:pi/1000:pi/2;
271             thetaD = theta.*180/pi;
272             BPResponse = ( 2*besselj(1, (pi*
                   physicalApertureDiameter/(c/fCenter)).*
                   sin(theta))...
273                                 ./(pi*
                                       physicalApertureDiameter
                                       /(c/fCenter).*sin(theta))
                                        ).^2;
```

```
274             plot([-transducerMaxAngleRad*180/pi
                    transducerMaxAngleRad*180/pi],[10^(-3/10)
                     10^(-3/10)],'r.',MarkerSize=15);
275             hold on;
276             plot(thetaD,abs(BPResponse),'b-');
277             legend("-3dB Level",Location="northwest")
278             title("A");
279             xlabel("Angle [deg]");
280             ylabel("Magnitude of Beampattern [-]");
281             grid on;
282             xlim([-90 90]);
283             xticks(-90:45:90);
284             subplot(1,2,2)
285             BPdB = 10*log10(BPResponse);
286             lowestDB = min(BPdB);
287             BPdB = BPdB - lowestDB;
288             polarplot([-transducerMaxAngleRad
                    transducerMaxAngleRad],[-lowestDB-3 -
                    lowestDB-3],'r.',MarkerSize=15);
289             hold on;
290             pp = polarplot(theta,BPdB,'b-');
291             ppAxes = get(pp, 'Parent');
292             ppAxes.RTick = round(linspace(0,-lowestDB,5)
                    );
293             ppAxes.RTickLabel = {round(linspace(lowestDB
                    ,0,5))};
294             title("B [dB]");
295             legend("-3dB level",Location="west");
296             fig.Position = [200 100 800 400];
297         end
298
299         %Visualize the receive signal from scatterers
300         %This plotting cannot be done with a parfor loop
```

```
301         % if numActiveScatterers >= 1 && doneTS(1) ==
               false
302         %     figTS = figure(Name = "Visualize Summed
               Receive Signals From X Scatterers");
303         %     subplot(3,1,1);
304         %     plot(timeVec,masterReceiveTS(:,iPing));
305         %     grid on;
306         %     title("A");
307         %     %ylabel("Amplitude [WU]");
308         %     doneTS(1) = true;
309         % elseif numActiveScatterers >= 2 && doneTS(2)
               == false
310         %     figure(figTS);subplot(3,1,2);
311         %     plot(timeVec,masterReceiveTS(:,iPing));
312         %     grid on;
313         %     title("B");
314         %     ylabel("Amplitude of Receive Series [WU]")
               ;
315         %     doneTS(2) = true;
316         % elseif numActiveScatterers >= 500 && doneTS(3)
                == false
317         %     figure(figTS);subplot(3,1,3);
318         %     plot(timeVec,masterReceiveTS(:,iPing));
319         %     grid on;
320         %     title("C");
321         %     xlabel("Time [s]");
322         %     %ylabel("Amplitude [WU]");
323         %     figTS.Position = [200 100 800 500];
324         %     doneTS(3) = true;
325         % end
326
327         %Visualize the receive signal from a single ping
               post replica correlation
328         %This plotting cannot be done with a parfor loop
```

```
329         % if numActiveScatterers >= 1 && doneRC(1) ==
               false
330         %     figRC = figure(Name = "Visualize Replica
               Correlation from X Scatterers");
331         %     subplot(3,1,1);
332         %     plot(lagVector,abs(masterReceiveRC(:,iPing
               )));
333         %     grid on;
334         %     title("A");
335         %     %ylabel("|Replica Correlation| [WU^2]");
336         %     doneRC(1) = true;
337         % elseif numActiveScatterers >= 2 && doneRC(2)
               == false
338         %     figure(figRC);subplot(3,1,2);
339         %     plot(lagVector,abs(masterReceiveRC(:,iPing
               )));
340         %     grid on;
341         %     title("B");
342         %     ylabel("|Replica Correlation| [WU^2]");
343         %     doneRC(2) = true;
344         % elseif numActiveScatterers >= 500 && doneRC(3)
                == false
345        %      figure(figRC);subplot(3,1,3);
346         %     plot(lagVector,abs(masterReceiveRC(:,iPing
               )));
347         %     grid on;
348         %     title("C");
349         %     xlabel("Lag [s]");
350         %     %ylabel("|Replica Correlation| [WU^2]");
351         %     figRC.Position = [200 100 800 500];
352         %     doneRC(3) = true;
353         % end
354
355     %End Plotting subsection
```

```
356     end
357     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
358
359 end
360
361
362 %Finds the range equivalent of the receive time vector
363 receiveRange = (timeVec)*c/2;
364 runTime=toc;
365
366 %Adds time varying gain to the signal to counter the
       effect of spherical spreading
367 if applyTVG == true
368     tvg = (0:(N-1)).'.^4 .*(N.^-4);
369     masterReceiveRC = masterReceiveRC.*tvg;
370 end
371
372
373
374 %% This applies the beamformer to the simulated data
375
376 %Prepares the information produced in the script for the
       beamforming code
377 method = 'spline';
378 S.data = masterReceiveRC.';
379 S.x0 = desiredImageCenter(2);
380 S.xSwath = 2*S.x0;
381 S.Htx = physicalApertureDiameter;
382 S.Hrx = S.Htx;
383 S.fs = fs;
384 S.c = c;
385 S.f0 = fCenter;
386 S.Bw = bandwidth;
387 S.du = distanceBetweenPings;
```

```
388 S.domain = 't';
389
390 SBeamformed = wk(S,method);
391
392
393 %% More Plotting
394
395 if showPlots == true
396
397     %For visualizing the replica correlation data,
        imagesc
398     fig = figure(Name = "Imagesc Replica Correlation");
399     dbReference = max(max(abs(masterReceiveRC)));
400     imageSCData = 10*log10(abs(masterReceiveRC.')./
        dbReference);
401     imagesc(receiveRange,vehicleXCoords,imageSCData)
402     set(gca,'YDir','normal')
403     hc = colorbar;
404     hc.Label.String = "Magnitude of Replica Correlation,
         dB re:max";
405     colormap('turbo')
406     clim([-30 0])
407     %title("imagesc of replica correlated receive at
        each along-track position")
408     xlabel("Range [m]");
409     ylabel("Along-track Ping Position [m]");
410     axis image;
411     fig.Position = [200 100 800 500];
412
413     %Visualize Demodulation
414     df = fs/N;
415     freqs = (0:(N-1)).'.*df;
416     middleIndex = round(size(masterReceiveTS,2)/2);
```

```
417     LSMod = fft(masterReceiveTSAnalytic(:,middleIndex))
            .*dt;
418     LSDemod = fft(masterReceiveTSDemodulated(:,
            middleIndex)).*dt;
419     fig = figure(Name = "Demodulation");
420     plot(freqs,abs(LSMod),LineWidth=2,DisplayName="
            Modulated")
421     hold on;
422     plot(freqs,abs(LSDemod),LineWidth=2,DisplayName="
            Demodulated")
423     xline(fCenter,'r',DisplayName="Carrier Frequency")
424     xline(fs/2,'b',DisplayName="Nyquist Frequency")
425     hold off;
426     grid on;
427     legend()
428     xlabel("Frequencies")
429     ylabel("Magnitude of Linear Spectrum")
430     title('Modulated and Demodulated Signals Post
            Hilbert Transform')
431     fig.Position = [200 100 800 500];
432
433     %Visualize the reconstructed image of the seafloor
434     fig = figure(Name = "Omega=k output, Image of
            seafloor");
435     SBeamformed.data = SBeamformed.data./max(abs(
            SBeamformed.data(:)));
436     imagesc(SBeamformed.x,SBeamformed.y,10.*log10(abs(
            SBeamformed.data)))
437     set(gca,'YDir','normal')
438     hc = colorbar;
439     hc.Label.String = "db re: max";
440     colormap('turbo')
441     clim([-60 0])
442     xlabel("Range [m]")
```

```
443     ylabel("Along-track [m]")
444     %title("D")
445     axis image;
446     %xlim([7.92 12.08]);
447     %ylim([5.1 8.1]);
448     fig.Position = [200 100 1000 600];
449
450     %Visualize the contrast ratio which can be used to
            quantify the image quality
451     [~,iMaxX] = max(max(SBeamformed.data,[],1));
452     [~,iMaxY] = max(max(SBeamformed.data,[],2));
453     fig = figure(Name="Quantitative Measurement Slices")
            ;
454     subplot(1,3,1:2)
455     imagesc(SBeamformed.x,SBeamformed.y,10.*log10(abs(
            SBeamformed.data)))
456     set(gca,'YDir','normal')
457     hc = colorbar;
458     hc.Label.String = "db re: max";
459     colormap('turbo')
460     clim([-60 0])
461     xlabel("Range [m]")
462     ylabel("Along-track [m]")
463     title("A")
464     axis image;
465     %xlim([5 10]);ylim([8.4 14.4]);
466     hold on;
467     line([SBeamformed.x(1) SBeamformed.x(end)], [
            SBeamformed.y(iMaxY-3) SBeamformed.y(iMaxY-3)],'
            Color','red')
468     line([SBeamformed.x(1) SBeamformed.x(end)], [
            SBeamformed.y(iMaxY+3) SBeamformed.y(iMaxY+3)],'
            Color','red')
469     subplot(1,3,3)
```

```
470     plot(SBeamformed.x,10*log10(abs(SBeamformed.data(
            iMaxY,:))),LineWidth=2);
471     grid on;
472     title("B");xlabel("Range [m]");ylabel("db re: max");
473     %xlim([5 10]);
474     fig.Position = [200 100 1000 500];
475 end
```

A.2 Supporting Functions

imageToScattererAmplitude.m

```
1  function [scattererCoords,scattererNormals,
       scattererPartialAmplitudes] =
       imageToScattererAmplitude(...
2       imageName,imageBtmLftCorner,imageDimensions,
            scattererCondensing,incoherenceMethod,...
3       environmentThreshold,showPlots)
4  %This function converts an image into a grayscale image,
       the values of
5  %which become scatterer amplitudes between 0 and 1
6  %Inputs:
7  %imageName, the file name needed to access the image
8  %imageCenter, the [X,Y,Z] coordinates of the center of
       the image
9  %imageDimensions, the [X(height), Y(width)] of the image
10 %scattererCondensing, the [numX;numY] scatterers that
       should be combined
11 %into a single scatterer. Or, the ratio that will be
       used to compress the
12 %image, [ratio]: <1 compresses, >1 expands
13
14 %This loads in the image
```

```
15 seafloor = imread(imageName);
16 seafloorGrayScale = double( rgb2gray(seafloor) );
17 seafloorPixelAmp = 255-seafloorGrayScale;% Inverts the
     image so that dark colors have a higher amplitude
18 seafloorPixelAmp(seafloorPixelAmp==255) =
     seafloorPixelAmp(seafloorPixelAmp==255)-250;
19 seafloorPixelAmp(seafloorPixelAmp==0) = seafloorPixelAmp
     (seafloorPixelAmp==0)+10;
20 seafloorPixelAmp = seafloorPixelAmp/max(max(
     seafloorPixelAmp));
21 seafloorPAPlot = seafloorPixelAmp;
22
23 %This condenses the scatterers if needed using a
     homemade algorithm
24 if length(scattererCondensing) > 1 & scattererCondensing
     ~= [1,1]
25     condenseX = scattererCondensing(1);
26     condenseY = scattererCondensing(2);
27     [numRowsX,numColsY] = size(seafloorPixelAmp);
28     if mod(numRowsX/condenseX,1)~=0 || mod(numColsY/
         condenseY,1)~=0
29         error("The scatterers must be condensed to whole
               numbers")
30     end
31     iScatX = 1;
32     for iRowX = 1:condenseX:numRowsX %Height is in the x
           direction
33     iScatY = 1;
34         for iColY = 1:condenseY:numColsY %Width is in
             the y direction
35             condensedSeafloorGrayScale(iScatX, iScatY) =
                   mean(mean(...
36                 seafloorPixelAmp( iRowX:(iRowX+condenseX
                     -1), iColY:(iColY+condenseY-1) )));
```

```
37              iScatY = iScatY + 1;
38          end
39          iScatX = iScatX + 1;
40      end
41      seafloorPixelAmp = condensedSeafloorGrayScale;
42  elseif scattererCondensing ~= [1,1]
43      %This condenses the scatterers if needed using
            MATLAB's imresize
44      seafloorPixelAmp = imresize(seafloorPixelAmp,1./
            scattererCondensing);
45  end
46
47
48
49  %This calculates the scatterer density (Scatterers/m^2)
50  [numRows,numCols] = size(seafloorPixelAmp);
51  numScatterers = numRows*numCols;
52  imageHeight = imageDimensions(1);
53  imageWidth = imageDimensions(2);
54  imageArea = imageHeight*imageWidth;
55  scattererDensity = numScatterers/imageArea;
56
57  % zeta ensures that the field scattered from any two
        points is incoherent.
58  % mu and sigma are the mean and standard deviation of a
        normally
59  % distributed random variable
60
61  %There is a better way to do this: try using ind2sub,
        scaling, and moving
62  scattererCoords = zeros(numScatterers,3);
63  scattererNormals = [zeros(numScatterers,2), -1.*ones(
        numScatterers,1)];
64  scattererColorCrossSections = zeros(numScatterers,1);
```

```
65 scattererRandomTimeDelay = zeros(numScatterers,1);
66 iScat = 1;
67 for iScatW = 1:numCols %Width is in the y direction
68     for iScatH = 1:numRows %Height is in the x direction
69         scattererCoords(iScat,:) = [...
70             1-(iScatH-0.5)/numRows, (iScatW-0.5)/numCols
                , 0].*[imageHeight, imageWidth, 0];
71         scattererColorCrossSections(iScat) =
               seafloorPixelAmp(iScat);
72
73         %This divides the image into four quadrants
               where a different
74         %implementation of a scatterer amplitude is used
               in each quadrant
75         if scattererColorCrossSections(iScat) <=
               environmentThreshold
76             switch incoherenceMethod
77                 case 'possm'
78                     XRand = randn(RandStream('mt19937ar'
                           ,'Seed',iScat));%Mersenne Twister
79                     YRand = randn(RandStream('dsfmt19937
                           ','Seed',iScat));%SIMD-Oriented
                           Fast Mersenne Twister
80                     zeta = (XRand+1j.*YRand)./sqrt(2);
81                 case 'possmMag1'
82                     zeta = exp(1j*randn(RandStream('
                           mt19937ar','Seed',iScat)));
83                 case 'zRand'
84                     mu=0;
85                     sigma=0.015;
86                     scattererCoords(iScat,3) =
                           scattererCoords(iScat,3) +
                           normrnd(mu,sigma,1,1);
87                     zeta = 1;
```

```
88                  case 'mirror'
89                      zeta = 1;
90              end
91          else %Don't want the target to be distorted
92              zeta = 1;
93          end
94
95          scattererRandomTimeDelay(iScat) = zeta;
96
97          iScat = iScat + 1;
98      end
99  end
100 scattererCoords = scattererCoords + imageBtmLftCorner;
101 scattererPartialAmplitudes = scattererRandomTimeDelay.*
    sqrt(scattererColorCrossSections./scattererDensity);
102
103 if showPlots == true
104     % Visualize what the first half of this function
        does
105     fig = figure(Name="Scatterer Amplitudes");
106     hold on;
107     subplot(2,2,1)
108     imagesc(seafloor);colorbar;set(colorbar,'visible','
        off');
109     axis image
110     title("A");
111     ylabel("Scatterer Indices");%xlabel("Scatterer
        Indices");
112     subplot(2,2,2)
113     imagesc(seafloorGrayScale);cb=colorbar;cb.Label.
        String="Scatterer Amplitude";colormap(sky);
114     axis image
115     title("B");%xlabel("Scatterer Indices");ylabel("
        Scatterer Indices");
```

```
116     subplot(2,2,3)
117     imagesc(seafloorPAPlot);cb=colorbar;cb.Label.String
          ="Scatterer Amplitude";colormap(sky);
118     axis image
119     title("C");
120     xlabel("Scatterer Indices");
121     ylabel("Scatterer Indices");
122     subplot(2,2,4)
123     imagesc(seafloorPixelAmp);cb=colorbar;cb.Label.
          String="Scatterer Amplitude";colormap(sky);
124     axis image
125     title("D");
126     xlabel("Scatterer Indices");%ylabel("Scatterer
          Indices");
127     fig.Position = [200 100 900 500];
128
129     % Visualize what the second half of this function
          does
130     fig = figure(Name="Scatterer Amplitudes");
131     hold on;
132     subplot(2,2,1);hold on;
133     idx = scattererColorCrossSections >
          environmentThreshold;
134     plot3(scattererCoords(idx,1),scattererCoords(idx,2),
          scattererCoords(idx,3),'b.',DisplayName="large
          scatterer magnitude");
135     plot3(scattererCoords(~idx,1),scattererCoords(~idx
          ,2),scattererCoords(~idx,3),'r.',DisplayName="
          small scatterer magnitude");
136     title("A                                    ");
137     xlabel("Along-Track Position [m]");
138     ylabel("Range Position [m]");
139     legend(Location="best");
140     axis image;
```

```
141     view([90 90]);
142     set(gca, 'zdir','reverse');
143     set(gca, 'xdir','reverse');
144     subplot(2,2,2);hold on;
145     idx = abs(scattererPartialAmplitudes) >
          environmentThreshold/10;
146     plot3(scattererCoords(idx,1),scattererCoords(idx,2),
          scattererCoords(idx,3),'b.',DisplayName="large
          scatterer magnitude");
147     plot3(scattererCoords(~idx,1),scattererCoords(~idx
          ,2),scattererCoords(~idx,3),'r.',DisplayName="
          small scatterer magnitude");
148     title("B                                    ");
149     xlabel("Along-Track Position [m]");
150     ylabel("Range Position [m]");
151     legend(Location="best");
152     axis image;
153     view([90 90]);
154     set(gca, 'zdir','reverse');
155     set(gca, 'xdir','reverse');
156     subplot(2,2,3);
157     hold on;
158     plot(real(scattererPartialAmplitudes),imag(
          scattererPartialAmplitudes),'rx');
159     grid on;
160     title("C");
161     xlabel("Real Part of Amplitude");
162     ylabel("Imaginary Part of Amplitude");
163     axis image;
164     fig.Position = [200 100 900 600];
165 end
166
167 end
```

findVehiclePingOffset.m

```
1  function [correctedXValue] = findVehiclePingOffset(
       scattererCoords,transducerMaxAngleRad,beginningOrEnd)
2  %Assumes that the sonar only travels in the x direction
       and that the sonar
3  %is pointed at the center of a rectangular image
4  %Provides the starting and stopping offset so that the
       entire scene is seen
5  %Inputs:
6  %scattererCoords, in formation [X,Y,Z], where each row
       is one scatterer
7  %transducerMaxAngle, in radians
8  %beginningOrEnd, a string stating whether the beginning
       or ending portion
9  %of the scatterers is wanted
10 %Outputs:
11 %correctedXValue, the starting or ending vehicle
       position of the sonar so
12 %that the entire image is seen and the first and last
       ping doesn't contain
13 %any of the scatterers
14
15 if beginningOrEnd == "beginning"
16     %Finds the furthest scatterer in the smallest x
           value:
17     XExtremeScattererCoords = scattererCoords(
           scattererCoords(:,1)==min(scattererCoords(:,1))
           ,:);
18 elseif beginningOrEnd == "end"
19     %Finds the furthest scatterer in the largest x value
           :
20     XExtremeScattererCoords = scattererCoords(
           scattererCoords(:,1)==max(scattererCoords(:,1))
```

```
        ,:);
end

[nScat,~] = size(XExtremeScattererCoords);
iCentralScatterer = floor(nScat/2+1);

XCenterExtremeScattererCoord = XExtremeScattererCoords(
   end,:);
%Finds the starting x coordinate of the vehicle based on
    the defining scatterer
XDistance = sqrt(sum( XCenterExtremeScattererCoord(2:3)
   .^2 ));

xPingDistanceOffset = tan(transducerMaxAngleRad+0.5*pi
   /180)*XDistance;%see output description for rounding
   up

if beginningOrEnd == "beginning"
    correctedXValue = XExtremeScattererCoords(1,1) -
       xPingDistanceOffset;
elseif beginningOrEnd == "end"
    correctedXValue = XExtremeScattererCoords(1,1) +
       xPingDistanceOffset;
end

end
```

findGeometry.m

```
function [distances,thetaTransmits,thetaIncidents] =
   findGeometry(...
    vehicleCoord,vehicleNormal,scattererCoords,
       scattererNormals,showPlots,iPing)
%This function provides the distances between a vehicle'
```

```
    s path at every point
%and a scatterer at a single point. It also calculates
    ther important
%angles which affect the scatterer amplitude.
%Inputs:
%vehicleCoords, the [X,Y,Z] coordinates of the vehicle's
     path
%vehicleNormal, the [X,Y,X] normal vector of the vehicle
%scattererCoords, the [x,y,z] coordinates of the
    scatterer
%scattererNormals, the [X,Y,Z] normal vector of the
    scatterer
%transducerMaxAngleRad, the angle of the cone shape
    created by the
%transducer which is used to find active scatterers
%Output:
%distances, The distances between the vehicle and
    scatterer across all points
%thetaTransmits, the angle between the vehicle normal
    and the vector from the vehicle to the scatterer [rad
    ]
%thetaIncidents, the angle between the scatterer normal
    and the vector from the scatterer to the normal [rad]

%Calculate the distances between the vehicle and every
    scatterer
distances = vecnorm((vehicleCoord.*ones(size(
    scattererCoords))-scattererCoords).').';

%Creates vectors between the vehicle and each scatterer
vehicleToScatterers = scattererCoords-vehicleCoord;
scatterersToVehicle = -1.*vehicleToScatterers;

%thetaIncident is the angle from the surface normal at
```

```
    which the incoming wave is incident
thetaIncidents = acos( sum((scatterersToVehicle.*
    scattererNormals),2) ./...
     (vecnorm(scattererNormals.').'.*vecnorm(
        scatterersToVehicle.').') );%a dot b / |a||b|
%thetaTransmit is the angle from the transducer normal
    at which the outgoing and incoming waves propagate
thetaTransmits = acos( sum((vehicleToScatterers.*
    vehicleNormal),2) ./...
     (vecnorm(vehicleToScatterers.').'.*vecnorm(
        vehicleNormal.').') );%a dot b / |a||b|



%Visualize what this function does:
if showPlots == true && iPing == 1
     fig = figure(Name="Sonar and Scatterer Geometry");
     hold on;
     % Choose a scatterer
     scatIdx=floor(length(scattererCoords)/2);
     % Make quiver lengths appropriate
     scaleQuivers = 0.1;
     vehicleNormal = vehicleNormal.*scaleQuivers;
     scattererNormals = scattererNormals.*scaleQuivers;
     vehicleToScatterers = scaleQuivers.*
        vehicleToScatterers./(vecnorm(vehicleToScatterers
        .').');
     % Bring points closer together to improve visibility
     scaleCoords = 0.05;
     vehicleCoord = scaleCoords.*vehicleCoord;
     scattererCoords = scaleCoords.*scattererCoords;
     % Plot ping location, scatterer location, and
        normals
     plot3(vehicleCoord(1),vehicleCoord(2),vehicleCoord
```

```
    (3),'k.',MarkerSize=50,DisplayName="Single Ping
    Location")
51  plot3(scattererCoords(scatIdx,1),scattererCoords(
    scatIdx,2),scattererCoords(scatIdx,3),'r.',
    MarkerSize=50,DisplayName="Single Scatterer");
52  quiver3(vehicleCoord(1),vehicleCoord(2),vehicleCoord
    (3),...
53      vehicleNormal(1),vehicleNormal(2),vehicleNormal
        (3),...
54      LineWidth=3,MaxHeadSize=3,DisplayName="
        Transducer Normal")
55  quiver3(vehicleCoord(1),vehicleCoord(2),vehicleCoord
    (3),...
56      vehicleToScatterers(scatIdx,1),
        vehicleToScatterers(scatIdx,2),
        vehicleToScatterers(scatIdx,3),...
57      LineWidth=3,MaxHeadSize=3,DisplayName="Vehicle
        to Scatterer")
58  % Plot angles - this code was derived from MATLAB's
    help page on "Plot the angle between two vectors"
59  fill=1000;
60  xArc = linspace(vehicleNormal(1),vehicleToScatterers
    (scatIdx,1),fill);
61  yArc = linspace(vehicleNormal(2),vehicleToScatterers
    (scatIdx,2),fill);
62  zArc = linspace(vehicleNormal(3),vehicleToScatterers
    (scatIdx,3),fill);
63  norm = 2.*vecnorm([xArc;yArc;zArc])./scaleQuivers;
64  plot3(vehicleCoord(1)+xArc./norm,vehicleCoord(2)+
    yArc./norm,vehicleCoord(3)+zArc./norm,'-',...
65      MarkerFaceColor=[0.8500 0.3250 0.0980],LineWidth
        =4,DisplayName="\theta_T_r_a_n_s_m_i_t")
66  %Plot the vectors of the scatterer here so legend is
    easy to follow
```

```
67      quiver3(scattererCoords(scatIdx,1),scattererCoords(
           scatIdx,2),scattererCoords(scatIdx,3),...
68          scattererNormals(scatIdx,1),scattererNormals(
               scatIdx,2),scattererNormals(scatIdx,3),...
69          LineWidth=3,MaxHeadSize=3,DisplayName="Scatterer
               Normal");
70      quiver3(scattererCoords(scatIdx,1),scattererCoords(
           scatIdx,2),scattererCoords(scatIdx,3),...
71          -1*vehicleToScatterers(scatIdx,1),-1*
               vehicleToScatterers(scatIdx,2),-1*
               vehicleToScatterers(scatIdx,3),...
72          LineWidth=3,MaxHeadSize=3,DisplayName="Scatterer
               to Vehicle");
73      xArc = linspace(scattererNormals(scatIdx,1),-1.*
           vehicleToScatterers(scatIdx,1),fill);
74      yArc = linspace(scattererNormals(scatIdx,2),-1.*
           vehicleToScatterers(scatIdx,2),fill);
75      zArc = linspace(scattererNormals(scatIdx,3),-1.*
           vehicleToScatterers(scatIdx,3),fill);
76      norm = 2.*vecnorm([xArc;yArc;zArc])./scaleQuivers;
77      plot3(scattererCoords(scatIdx,1)+xArc./norm,
           scattererCoords(scatIdx,2)+yArc./norm,
           scattererCoords(scatIdx,3)+zArc./norm,'-',...
78          MarkerFaceColor=[0.6350 0.0780 0.1840],...
79          LineWidth=4,DisplayName="\theta_I_n_c_i_d_e_n_t
               ")
80      %Plot a line connecting the points
81      nConnect = 15;
82      noLegPlot = plot3(...
83          linspace(vehicleCoord(1),scattererCoords(scatIdx
               ,1),nConnect),...
84          linspace(vehicleCoord(2),scattererCoords(scatIdx
               ,2),nConnect),...
85          linspace(vehicleCoord(3),scattererCoords(scatIdx
```

```
            ,3),nConnect),...
86          'k--',DisplayName="");
87      set(noLegPlot, 'HandleVisibility', 'off');
88      % Make plot nicer
89      set(gca, 'zdir','reverse');
90      set(gca, 'xdir','reverse');
91      grid on;
92      axis image;
93      legend(Location="north");
94      set(gca, 'XTickLabel', []);
95      set(gca, 'YTickLabel', []);
96      set(gca, 'ZTickLabel', []);
97      xlabel("Along-Track Direction [m]");
98      ylabel("Range Direction [m]");
99      zlabel("Depth [m]");
100     view(120,30);
101     fig.Position = [400 100 800 500];
102 end
103
104 end
```

Bibliography

[1] Matthews, A. D., T. C. Montgomery, D. A. Cook, J. W. Oeschger, and J. S. Stroud (2006) "12.75" Synthetic Aperture Sonar (SAS), High Resolution and Automatic Target Recognition," in *OCEANS 2006*, pp. 1–7, iSSN: 0197-7385.
URL https://ieeexplore.ieee.org/document/4098865

[2] Brown, D. C., S. F. Johnson, I. D. Gerg, and C. F. Brownstead (2019) "Simulation and testing results for a sub-bottom imaging sonar," *Proceedings of Meetings on Acoustics*, **36**(1), p. 070001.
URL https://doi.org/10.1121/2.0001012

[3] Choi, W.-S., D. R. Olson, D. Davis, M. Zhang, A. Racson, B. Bingham, M. McCarrin, C. Vogt, and J. Herman (2021) "Physics-Based Modelling and Simulation of Multibeam Echosounder Perception for Autonomous Underwater Manipulation," *Frontiers in Robotics and AI*, **8**, publisher: Frontiers.
URL https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/fro

[4] Jackson, D., K. Bemis, G. Xu, and A. Ivakin (2022) "Sonar Observation of Heat Flux of Diffuse Hydrothermal Flows," *Earth and Space Science*, **9**(10), p. e2021EA001974, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021EA001974.
URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2021EA001974

[5] "Absorption," .
URL https://www.acs.psu.edu/drussell/Demos/Absorption/Absorption.html

[6] Porter, M. B. "The BELLHOP Manual and User's Guide: PRELIMINARY DRAFT," .

[7] Hunter, A. J. (2006) *Underwater Acoustic Modelling for Synthetic Aperture Sonar*.
URL http://ir.canterbury.ac.nz/handle/10092/1117

[8] Urick, R. J. (1983) *Principles of underwater sound*, 3rd ed ed., McGraw-Hill, New York.

[9] Pouliquen, E., O. Bergem, and N. G. Pace (1999) "Time-evolution modeling of seafloor scatter. I. Concept," *The Journal of the Acoustical Society of America*, **105**(6), pp. 3136–3141.
URL https://doi.org/10.1121/1.424644

[10] Jackson, D. R. and M. D. Richardson (2007) *High-Frequency Seafloor Acoustics*, Springer, New York, NY.
URL http://link.springer.com/10.1007/978-0-387-36945-7

[11] "Detailed Explanation of the Finite Element Method (FEM)," .
URL https://www.comsol.com/multiphysics/finite-element-method

[12] Kirkup, S. (1998) *The boundary element method in acoustics: a development in Fortran*, no. 1 in Integral equation methods in engineering, Integrated Sound Software, Hebden Bridge.

[13] Abawi, A. T. (2016) "Kirchhoff scattering from non-penetrable targets modeled as an assembly of triangular facets," *The Journal of the Acoustical Society of America*, **140**(3), pp. 1878–1886.
URL https://pubs.aip.org/jasa/article/140/3/1878/649485/Kirchhoff-scattering-

[14] Brown, D. C., S. F. Johnson, and D. R. Olson (2017) "A point-based scattering model for the incoherent component of the scattered field," *The Journal of the Acoustical Society of America*, **141**(3), p. EL210.

[15] Sammelmann, G. (2001) "Propagation and scattering in very shallow water," in *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*, vol. 1, pp. 337–344 vol.1.
URL https://ieeexplore.ieee.org/document/968749/?arnumber=968749tag=1

[16] Sammelmann, G. S. "PERSONAL COMPUTER SHALLOW WATER ACOUSTIC TOOL-SET (PC SWAT) 7.0: LOW FREQUENCY PROPAGATION AND SCATTERING," .

[17] Potokar, E., S. Ashford, M. Kaess, and J. G. Mangelson (2022) "HoloOcean: An Underwater Robotics Simulator," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3040–3046.
URL https://ieeexplore.ieee.org/abstract/document/9812353

[18] "HoloOcean: Realistic Sonar Simulation | Request PDF," in *ResearchGate*.
URL https://www.researchgate.net/publication/366611363$_{H}oloOcean_{R}ealistic_{S}onar_{S}$

[19] Abawi, A. T. and P. Krysl (2017) "Coupled finite element/boundary element formulation for scattering from axially-symmetric objects in three dimensions," *The Journal of the Acoustical Society of America*, **142**(6), pp. 3637–3648.
URL https://pubs.aip.org/jasa/article/142/6/3637/695107/Coupled-finite-elemer

[20] Goddard, R. P. (2008) *The Sonar Simulation Toolset, Release 4.6: Science, Mathematics, and Algorithms:*, *Tech. rep.*, Defense Technical Information Center, Fort Belvoir, VA.
URL http://www.dtic.mil/docs/citations/ADA487996

[21] Kargl, S. G., K. L. Williams, and A. L. Espana "ACOUSTIC SCATTERING FROM AN OBJECT IN A MUD LAYER OF A SHALLOW WATER WAVEGUIDE," *Conference Proceedings*.

[22] Kargl, S. G., A. L. España, K. L. Williams, J. L. Kennedy, and J. L. Lopes (2015) "Scattering From Objects at a Water–Sediment Interface: Experiment, High-Speed and High-Fidelity Models, and Physical Insight," *IEEE Journal of Oceanic Engineering*, **40**(3), pp. 632–642, conference Name: IEEE Journal of Oceanic Engineering.
URL https://ieeexplore.ieee.org/document/6932510

[23] Dalton, K. S., T. E. Blanford, and D. C. Brown (2022) "Simulating elastic targets for sonar algorithm development," Denver, Colorado, p. 070002.
URL https://pubs.aip.org/asa/poma/article/2842464

[24] Cook, D. A. and D. C. Brown (2018) "Synthetic Aperture Sonar Image Contrast Prediction," *IEEE Journal of Oceanic Engineering*, **43**(2), pp. 523–535, conference Name: IEEE Journal of Oceanic Engineering.
URL https://ieeexplore.ieee.org/document/7999174

[25] Marston, T. M. and D. S. Plotnick (2023) "Omega-K beamforming: Practical ramifications of alternative formulations," *JASA Express Letters*, **3**(7), p. 074802.
URL https://doi.org/10.1121/10.0020303