Modeling Maritime Radar Scattering

A Major Qualifying Project
submitted to the Faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
by

Matthew Allen

Allen Blaylock

Benjamin Davidson

Date: August 28th, 2012

MIT Lincoln Laboratory Supervisor: Dennis Blejer

Approved:

Professor Edward A. Clancy, Major Advisor

This work is sponsored under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Abstract

The focus of this project was the design and implementation of a maritime radar simulation developed in MATLAB to aid in the understanding of the effects of ocean waves on radar. The purpose of this simulation is to be used as a toolbox for the future development of detection algorithms for small boats on or near the ocean surface. The need for such toolbox is highlighted by two factors. First, little data exists pertaining to radar clutter resultant from deep ocean waves. Second, small water craft used for drug running, smuggling, and piracy pose an increasing threat to national security and military assets. Such threats can be reduced by using our toolbox to develop better detection algorithms. The team delivered three simulations to the MIT Lincoln Laboratory staff. The first simulation focused on the integration of a one-dimensional ocean model and a chirp radar model. The second deliverable extended the first simulation to include a two-dimensional ocean model and a boat wake model. The third simulation introduced a phased array radar model. These simulations were verified against publicly available data and models.

Acknowledgements

We would like to recognize and express gratitude to all the individuals who helped us complete our project through their combined support and expertise. Without these individuals, we could not have accomplished the many tasks we undertook with such quality.

We would first and foremost like to thank our project supervisor, Dennis Blejer, for dedicating his time and energy to mentor us by providing guidance to the various problems we faced and for sharing his wealth of knowledge of radar systems and the maritime environment. We would also like to thank Jennifer Watson for the opportunity to work on this project and her outstanding support. We appreciate and acknowledge Loretta Wesley for helping us organize our calendars and providing many reminders and supplies (especially cookies). We owe thanks to Byun Chansup for his timely help with LLGrid and pMATLAB for our work in parallelization. We also are grateful for the IT support from Scott Ehrlich who provided us with the hardware and software needed to execute our work effectively. In addition, Emily Anesta and Seth Hunter gave us valuable feedback and made us feel comfortable at MIT Lincoln Laboratory. We would also like to thank all of the members of group 105 and MIT Lincoln Laboratory for the opportunity to work with them.

We would lastly like to thank our project advisor, Edward Clancy, for his dedication and guidance during the course of our MQP. We greatly appreciated the constructive review and feedback of our writing and content, the helpful organization of our project goals and progress, as well as the guidance during our weekly meetings.

Statement of Authorship

This section contains an outline of the contributions of each member from our group. Many sections have had small contributions from other members of the group before reaching their final state. The details of this authorship presented are by the primary contributors to each task. Note that some work was accomplished over the summer prior to the official start date of the project. This project has been completed in partial fulfillment of the requirements for the degree of Bachelors of Science in the field of Electrical and Computer Engineering.

Matthew Allen

Summer Technical Accomplishments:

1) One-Dimensional Ocean Model (Contributed)
2) Boat Wake Model

Background Research:

1) Ocean Modeling
2) Wake Modeling

Code Development:

1) One-Dimensional Ocean Model (Contributed)
2) Boat Wake Model
3) Quasi-Two-Dimensional Ocean Model

Primary Author:

1) Non-Scanning, One-Dimensional Ocean Model
    a. One-Dimensional Ocean Model – Background, Methods
2) Non-Scanning, Quasi-Two-Dimensional Ocean Model
    a. Quasi-Two-Dimensional Ocean Model – Background, Methods
    b. Boat Wake – Background, Methods
3) Table of symbols

Allen Blaylock

Background Research:

1) Parallelization
2) Graphical User Interface

Code Development:

1) Parallelization
2) Graphical User Interface

Primary Author:

1) Introduction – Purpose, Scope, Sponsor
2) Parallelization (all)
3) Graphical User Interface (all)
4) Appendix D
5) Primary Editor

Benjamin Davidson

Summer Technical Accomplishments:

1) Non-Scanning Radar Model
2) Phased Array Radar Model
3) One-Dimensional Ocean Model

Background research:

1) Radar Modeling
2) Integration and Calibration

Code Development:

1) Non-Scanning Radar Model
2) Phased Array Radar Model
3) One-Dimensional Ocean Model
4) Integration (all three deliverables)

Primary Author:

1) Non-Scanning, One-Dimensional Ocean Model
    a. Non-Scanning Radar Model – Background, Methods
    b. Integration and Calibration – Background, Methods
    c. Results and Discussion
2) Non-Scanning, Quasi-Two-Dimensional Ocean Model
    a. Integration
    b. Results and Discussion
3) Phased Array, Quasi-Two-Dimensional Ocean Model (all)
4) Executive summary and conclusion
5) Appendix A to C
6) Primary Editor

Executive Summary

Currently, small boats and semi-submersible vessels are a threat faced by enforcement agencies due to their use in drug-running, smuggling, and piracy. Moreover, there exist significant risks to U.S. naval assets from small boats loaded with explosives. A recent instance of this problem was the attack on the USS Cole in 2000, where a small boat loaded with ordinance exploded against the destroyer's hull; the aftermath is shown in Figure 1.

Figure 1. Damage to USS Cole after small craft bombing ([Image of USS Cole], 2012).

In order to thwart such attacks and threats, enforcement agencies are interested in the detection of small boats. For the last fifty years, the problem of detecting large targets with radar has been well understood; however, there is little understanding and data supporting the detection of small boats on the ocean using radar. These detection problems are compounded by the fact that rough seas can severely inhibit the detection performance of small boats. A useful tool to gain understanding of a problem without data is a simulation. Although many radar simulations have been made in the past, they do not account for small vessels and oceanic factors.

The purpose of this project was to develop a robust MATLAB package for use in modeling radar returns from a maritime environment with small boats. The model is comprised of three major parts: the radar, the ocean, and the target simulations. The objective was to provide an easy to use graphical user interface into which radar, target, and ocean parameters could be input to produce one of three different ocean/radar simulations selected by the user. In addition, the simulation leverages parallel compute processes to reduce computation time and make the product more useful. Using the data returned from our simulation, signal processing experts at MIT Lincoln Laboratory can develop better processing algorithms for the detection of small vessels under various sea conditions.

There are two important components of this project, the first of which is the radar simulation. Radar, which is an acronym for RAdio Detection And Ranging, refers to a target detection system that has been used since World War II to determine the range, velocity, azimuth, and elevation of objects. Radar returns of a pulse compressed chirp radar system were modeled in the frequency-pulse-time domain using the electromagnetic signal time delay to point targets. Using an inverse Fourier transform over frequency and velocity of propagation of electromagnetic waves, we were able to model range processing. Similarly, we used a Fourier transform over time and the Doppler shift to extract velocity information from the target. Later, we modeled the phase shift between successive elements of a phased array radar to resolve targets in azimuth as well. The output of such radar processing is the range time intensity, which describes the range vs. time relationship; and the range Doppler profile, which relates range and velocity of targets.

The second major component of this project is the ocean surface model. The ocean is a very large and complex fluid system that is still not fully understood by experts in the field. There are many instances of energy transfer into the ocean from weather patterns, earthquakes, boats, and various other ocean disturbances. The nonlinear nature of these disturbances makes the system very complex and difficult to model. However, we used knowledge of fluid mechanics and oceanography to model the ocean surface height. Specifically, we modeled a one-dimensional ocean surface as a sum of sinusoids defined in amplitude and frequency by the Pierson-Moskowitz power spectral density. Next, the wave height of the ocean was scaled via the Beaufort scale. We modeled a quasi-two-dimensional ocean by replicating the one-dimensional model and optionally adding Gaussian variance and smoothing.

The MATLAB packages produced included models which simulate a one- and quasi-two-dimensional ocean as well as a chirp radar. These models were combined to produce radar returns for both the one- and two-dimensional oceans. Specifically, each sampled point on the ocean was considered as a scattering point target. More point targets were added to the simulation to represent boats and their wakes. The motion of the boats was modeled by basic kinematics equations allowing for each target to have an initial velocity as well as a constant acceleration. In addition, boat wake was modeled via the cusp wave crests of the Kelvin wake. Finally, the ocean scattering characteristics were calibrated to publically available ocean scattering distributions. The two models are illustrated in Figure 2 and Figure 3.

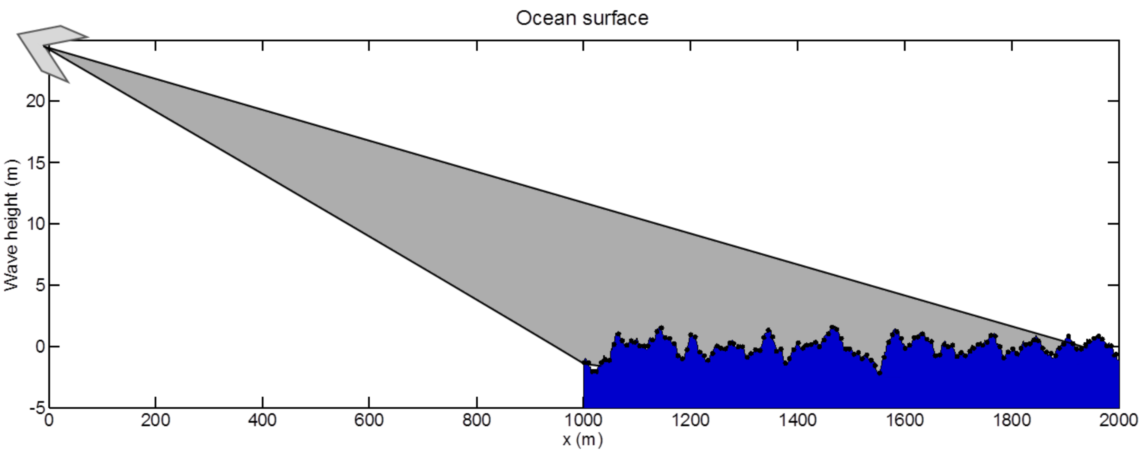


Figure 2. Visual representation of non-scanning radar and one-dimensional ocean integration.

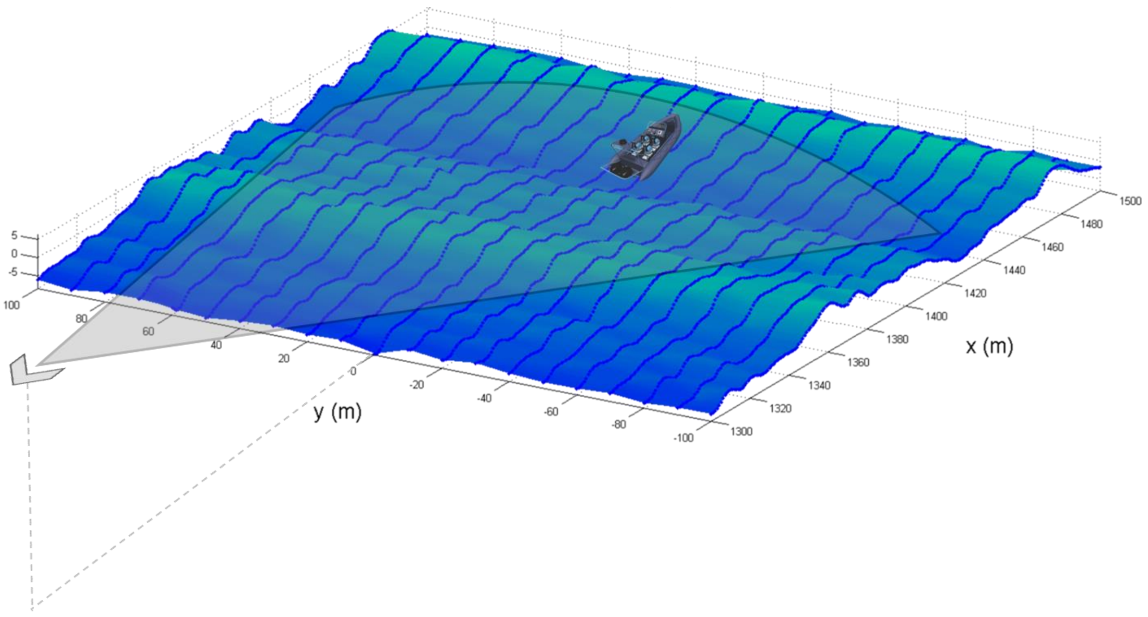
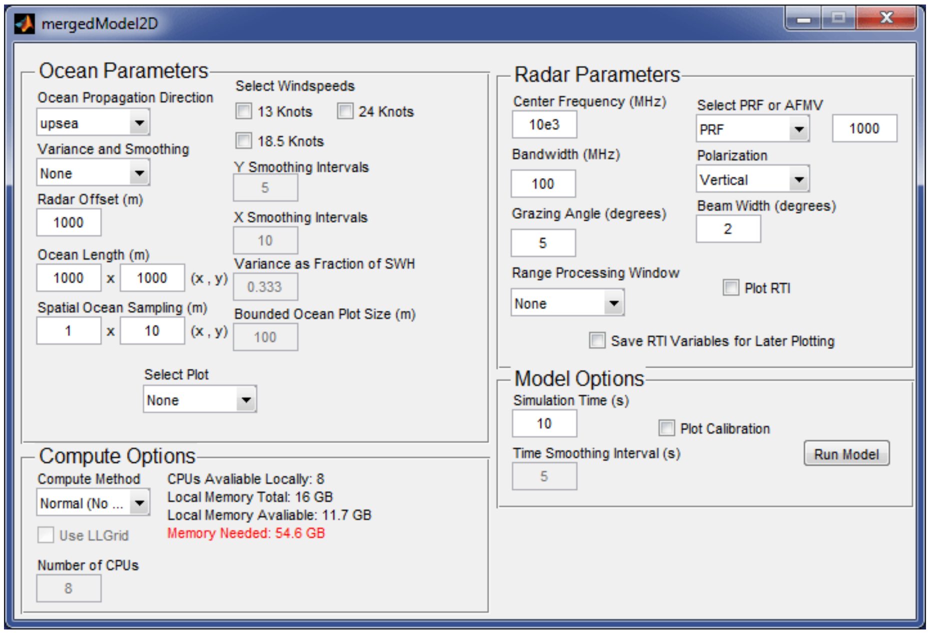


Figure 3. Visual representation of non-scanning radar, quasi-two-dimensional ocean, and target integration.

The group was successful at delivering both one-dimensional and two-dimensional integrated models to MIT Lincoln Laboratory. In addition, a phased array component simulation was developed and the framework for integration with the quasi-two-dimensional ocean was completed. The LLGrid, a supercomputer at Lincoln Laboratory, was leveraged for parallel computing. Using our models and final simulations, we were able to produce time series plots of the:

- one-dimensional ocean surface,
- quasi-two-dimensional ocean surface,
    - uniformly extrapolated
    - uniformly extrapolated with additive independent variance
    - uniformly extrapolated with additive correlated variance
- range time intensity plots and range Doppler profiles of
    - small boats,
    - one- and two-dimensional ocean surfaces,
    - small boats and ocean surfaces, and
    - small boats, their wake, and the quasi-two-dimensional ocean surface; and
- range angle plots and range Doppler profiles (phased array radar) of small boats and their wake.

A sample of these time series plots are provided in the accompanying attachment to this report. The framework of our model was constructed such that it is easy for a user to produce these plots quickly with parallelization and easily using our graphical user interface. The quasi-two-dimensional ocean generation graphical user interface is shown in Figure 4 and a sample range time intensity plot and range Doppler profiles are shown in Figure 5 and Figure 6.


mergedModel2D

Ocean Parameters
Ocean Propagation Direction: upsea
Variance and Smoothing: None
Radar Offset (m): 1000
Ocean Length (m): 1000 x 1000 (x , y)
Spatial Ocean Sampling (m): 1 x 10 (x , y)
Select Windspeeds: ☐ 13 Knots ☐ 24 Knots ☐ 18.5 Knots
Y Smoothing Intervals: 5
X Smoothing Intervals: 10
Variance as Fraction of SWH: 0.333
Bounded Ocean Plot Size (m): 100
Select Plot: None

Compute Options
Compute Method: Normal (No ...
☐ Use LLGrid
Number of CPUs: 8
CPUs Avaliable Locally: 8
Local Memory Total: 16 GB
Local Memory Avaliable: 11.7 GB
Memory Needed: 54.6 GB

Radar Parameters
Center Frequency (MHz): 10e3
Bandwidth (MHz): 100
Grazing Angle (degrees): 5
Range Processing Window: None
Select PRF or AFMV: PRF 1000
Polarization: Vertical
Beam Width (degrees): 2
☐ Plot RTI
☐ Save RTI Variables for Later Plotting

Model Options
Simulation Time (s): 10
Time Smoothing Interval (s): 5
☐ Plot Calibration

Figure 4. The quasi-two-dimensional ocean surface generation user interface.

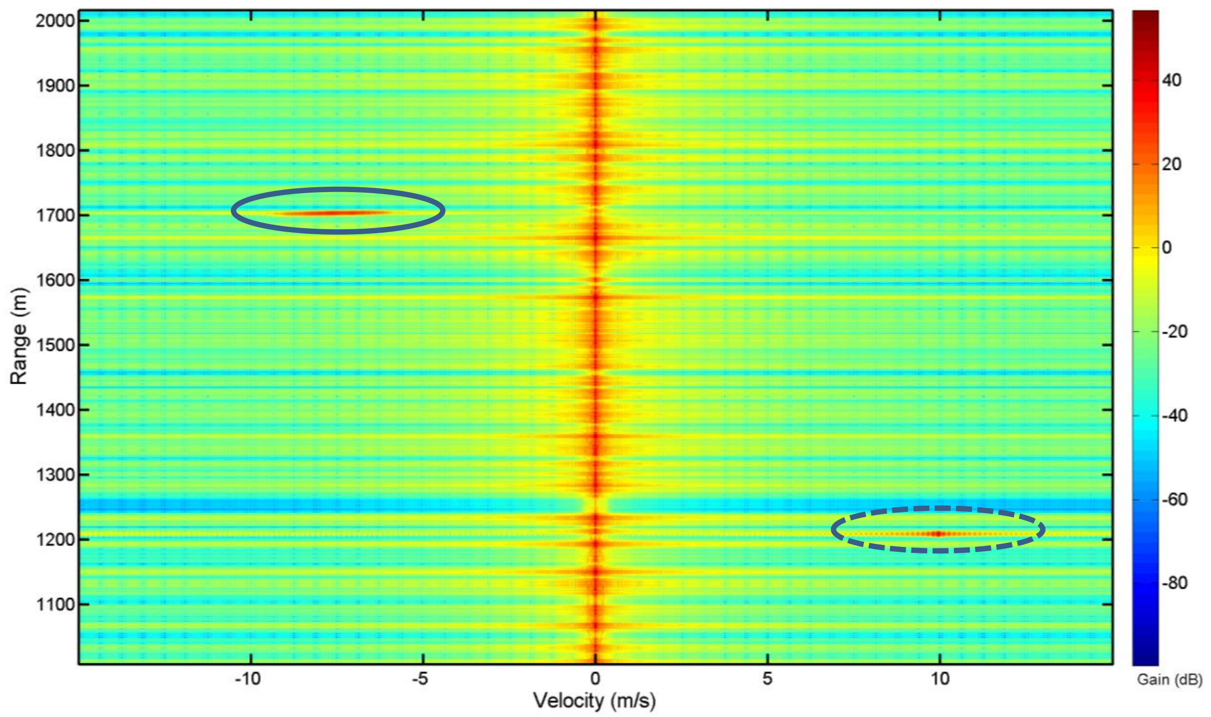


Figure 5. Range Doppler profile of one-dimensional ocean and two targets. The solid circle highlights an accelerating target while the dashed circle highlights a target with constant velocity.

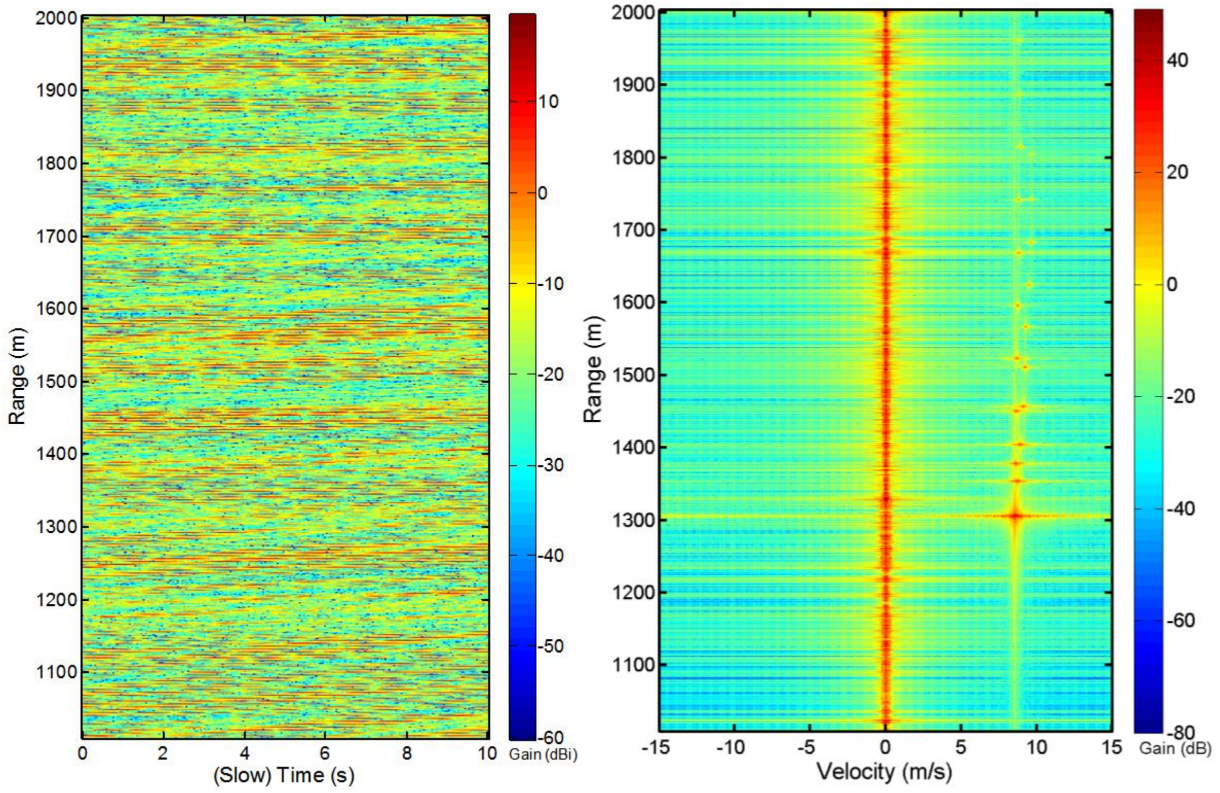


Figure 6. (left) Range time intensity plot of one-dimensional ocean surface with no targets; (right) range Doppler profile of quasi-two-dimensional surface with correlated variance and one target with wake (range = 1300 m, traveling toward the radar at 10 m/s at a 30 degree angle relative to the radar).

Table of Contents

Table of Tables

Table of Figures

Table of Symbols

Bold indicates a random process or variable. The symbol $(\overrightarrow{\ })$ indicates a vector.

Δf	Frequency domain sample interval (Hz)
Δm	Difference between computed and empirical modes
λ	Wavelength (m)
λ_0	Center frequency wavelength (m)
λ_{cusp}	Cusp wave wavelength (m)
μ	Mean (units same as data)
$\boldsymbol{\eta}$	Ocean surface height (m)
$\boldsymbol{\eta_{2D}}$	Two-dimensional ocean surface height (m)
$\boldsymbol{\eta_{2D_V}}$	Two-dimensional ocean surface height with variance (m)
$\boldsymbol{\eta_{2D_{VS}}}$	Two-dimensional ocean surface height with variance and smoothing (m)
$\boldsymbol{\eta_{scaled}}$	Scaled ocean surface (m)
θ	Phased array scan angle (degrees)
θ_{kelvin}	Kelvin arm angle (degrees)
$\theta_s[t_s]$	Phased array radar scan angle as function of slow time
Θ_{LKA}	Left kelvin arm angle (degrees)
Θ_{RKA}	Right kelvin arm angle (degrees)
σ	Radar cross section (m^2), standard deviation (units same as Data)
$\boldsymbol{\sigma_\eta}$	Standard deviation of ocean surface (m)
σ_{2D}	Two-dimensional standard deviation (m)
σ^0	Normalized radar cross section (unitless)
τ	Target index (unitless)
φ	Doppler phase shift (rad)
ϕ	Ocean frequency phase (rad)
ϕ_{cusp}	Cusp wave propagation angle (degrees)
ϕ_i	Phased array phase shift (rad)
ω	Angular frequency (rad/s)
ω_0	Center angular frequency (rad/s)
$\omega_{99\%}$	Frequency containing 99% of power of the Pierson-Moskowitz spectrum (rad/s)
ω_d	Doppler angular frequency (rad/s)
ω_m	Pierson-Moskowitz maximum angular frequency (rad/s)
$\mathscr{r}_n[t_s]$	Modified target range (see Appendix A.3) (m)
a	Acceleration (m/s^2)
A	Ocean wave scaling amplitude (m)
A_f	Radar interrogation area (m^2)
B	Bandwidth (Hz)
B_w	Beam width (rad)
c	Speed of electromagnetic signals in a vacuum (m/s)
$c_{pc}(t)$	Pulse compressed chirp signal
$c(t)$	Chirp signal
d	Distance between adjacent phased array antennas (m)

d_0	Initial distance (m)
$d_{alias-free}$	Alias-free distance (m)
$\mathcal{F}$	Fourier transform
$\mathcal{F}^{-1}$	Inverse Fourier transform
f_0	Center frequency (Hz)
f_d	Doppler frequency (Hz)
f_i	Chirp signal initial frequency (Hz)
f_{rate}	Chirp signal rate of frequency modulation (unitless)
g	Acceleration of gravity (m/s^2), calibration dilation factor (unitless)
$G(\omega)$	Pierson-Moskowitz spectrum (W/rad/s)
h	Radar height (m)
$\vec{H}$	Cusp wave crest vector
H_S	Significant wave height (m)
$H_{S(Beaufort)}$	Beaufort scale significant wave height (m)
$\boldsymbol{H}_{S(ocean\ model)}$	Ocean model significant wave height (m)
$H_{S(scale)}$	Significant wave scale (Unitless)
H_x	X component of cusp wave crest vector (m)
H_y	Y component of cusp wave crest vector (m)
k	Wavenumber for non-dispersive waves (1/m)
k_0	Center wavenumber (1/m)
k_d	Wavenumber for dispersive waves (1/m)
ℓ	Phased array transmitting element index
L_{ROI}	Kelvin arm length existing within the region of interest (m)
m_c	Computed mode of the sea scattering distribution
m_e	Empirical mode of a distribution
n	Phased array antenna receiving element index
$\vec{n}$	Normal unit vector to surface
$n_{ROI\ end}$	Ending cusp wave crest number within the region of interest
$n_{ROI\ start}$	Starting cusp wave crest number within the region of interest
N	Number of samples, number of cores
$\mathrm{N}(\mu, \sigma^2)$	Normal distribution
N_ω	Number of angular frequencies
$N_{\omega_{99\%}}$	Index of ocean sampling limiting frequency
N_E	Number of phased array antenna elements
$N_{required}$	Number of sampling points required to avoid ocean repetition
N_{ROI}	Number of targets existing in the region of interest
N_{t_s}	Number of slow times
$\mathrm{N_T}$	Number of targets
P	Portion of code that can be made parallel
P_{PT}	Position of point target (m)
PRF	Pulse repetition frequency (Hz)
r	Range (m)
r_τ	Range to target (m)
$r_{alias-free}$	Alias-free range (m)
r_ℓ	Phased array range from a transmitting antenna to a target (m)

$r_{\ell,\tau}[t_s]$	Distance from transmitting element to moving target (m)
r_n	Phased array range from a target to a receiving antenna (m)
$r_{n,\tau}[t_s]$	Distance from a moving target to receiving element (m)
$r_{repetition}$	Ocean repetition distance (m)
$r_{resolution}$	Radar range resolution (m)
r_{ROI}	Region of interest radius (m)
RAP	Range angle plot
RDP	Range doppler profile
$\mathbf{RGM}$	Random gaussian matrix
R_s	Radar start range (m)
RTI	Range time intensity
RTI_{ms}	Mode shifted range time intensity
$RTI_{ms,d}$	Mode shifted and dilated range time intensity
RTI_{ms,d,ms_2}	Calibrated range time intensity (twice mode shifted and dilated)
s_τ	Target slope (unitless)
t	Time (s)
t_0	Initial time (s)
$t_{alias-free}$	Alias-free time (s)
t_f	Fast time (s)
$t_{resolution}$	Radar time resolution (s)
t_s	Slow time (s)
T	Chirp signal duration (s)
T_{dwell}	Dwell time (s)
T_P	Pulse width (s)
$u(\,)$	Unit step function
$\overrightarrow{u_{los}}$	Radar line of sight unit vector
$U(a,b)$	Uniformly distributed random variable between a and b
v	Velocity (m/s)
$v_{alias-free}$	Alias-free velocity (m/s)
v_{phase}	Phase velocity (m/s)
v_r	Radial velocity (m/s)
$v_{resolution}$	Velocity resolution (m/s)
V	Radar return (V)
V_{boat}	Boat velocity (m/s)
V_{cusp}	Cusp wave velocity (m/s)
V_{wind}	Wind velocity (m/s)
W_ω	Windowing function over frequency
W_n	Windowing function over elements
W_{t_s}	Windowing function over slow time
x_τ	Absolute target distance (m)
y_τ	Absolute target cross-distance (m)
y_E	Absolute element cross-distance (m)
z	Radar signal two-way travel distance (m)

1 Introduction

Currently, small boats and semi-submersible vessels, such as that depicted in Figure 7, are a threat faced by enforcement agencies due to their use in drug-running, smuggling, and piracy. Moreover, there exist significant risks to U.S. naval assets from small boats loaded with explosives. A recent instance of this problem was the attack on the USS Cole in 2000, where a small boat loaded with ordinance exploded against the destroyer's hull; the aftermath is shown in Figure 8.

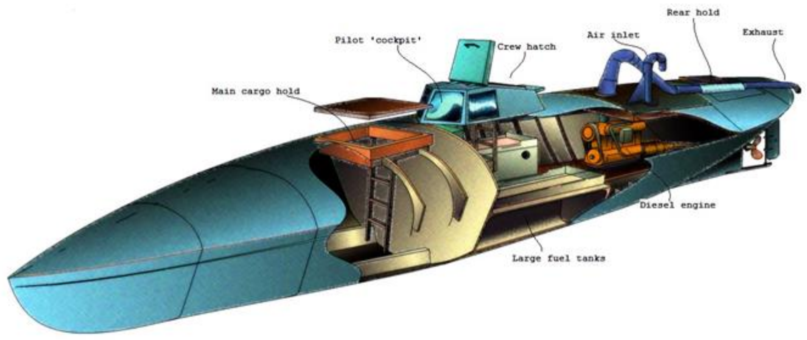


Figure 7. Semi-Submersible vessel ([Image of semi-submersible vessel], 2012).

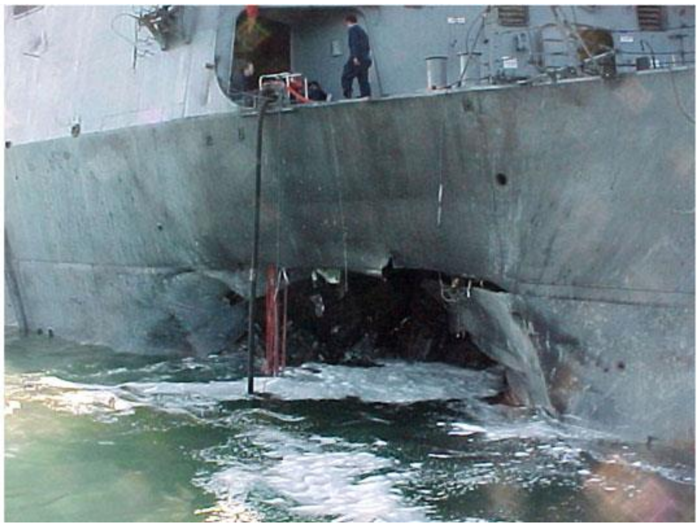


Figure 8. Damage to USS Cole after small craft bombing ([Image of USS Cole], 2012).

In order to thwart such attacks and threats, enforcement agencies are interested in the detection of small boats. For the last fifty years, the problem of detecting large targets with radar has been well understood; however, there is little understanding and data supporting the detection of small boats on the ocean using radar. These detection problems are compounded by the fact that rough seas can severely inhibit the detection performance of small boats. A useful tool to gain understanding of a problem without data is a simulation. Although many radar simulations have been made in the past, they do not account for small vessels and oceanic factors.

1.1 Purpose

The purpose of this project was to develop a robust software package for use in modeling radar returns from a maritime environment with small vessel targets. The model is comprised of three major parts: the radar, the ocean, and the target simulations. The objective was to provide an easy to use graphical user interface into which radar, target, and ocean parameters could be input to produce one of three different ocean/radar simulations selected by the user. In addition, the simulation leverages parallel compute processes to reduce computation time and make the product more useful. Using the data returned from our simulation, signal processing experts at MIT Lincoln Laboratory can develop better processing algorithms for the detection of small vessels under various sea conditions.

1.2 Scope

This project focused on three areas of investigation including the basics of radar, oceanography, and parallel computing. Both single antenna chirp radar as well as phased array radar were explored and modeled for later combination with ocean surface models. Deep sea ocean waves described in amplitude by the Beaufort scale and in frequency composition by the Pierson-Moskowitz spectrum were used to construct a simulated ocean. These models require enormous data sets which in turn create the requirement for algorithmic efficiency and parallelization. MIT Lincoln Laboratory has its own high performance computing cluster, the LLGrid, which was utilized to operate on the large data sets in a reasonable amount of time i.e. minutes instead of hours. These tasks can then be translated into the following project deliverables:

- Non-scanning one-dimensional ocean radar scattering simulation: This simulation is the model of chirp radar interrogating a one dimensional ocean. The ocean has a given length which is sampled in wave height at a fixed spatial interval. This model is limited to a target moving toward or away from the radar and can only be used for range and velocity processing.

- Non-scanning quasi-two-dimensional ocean radar scattering simulation: This model employs the radar specified above; however, the ocean consists of replications of the one-dimensional ocean orthogonal to the wave propagation direction. The ocean has optional Gaussian variance which is dependent on the time and location of the ocean surface. Targets on this ocean support cross-sea movement and optional wake.
- Phased array radar scattering simulation: This is a component model which uses phased array radar on boats and optional wake. The phased array enables range, velocity, and azimuth processing.
- Graphical user interface: The three models are accessible through a graphical user interface with input validation. Through the interface, the user can select the model to run as well as specify parameters such as wind speed and radar pulse reputation frequency.
- Parallelization: Each of the three models was parallelized to leverage multiple cores present on the majority of computers today as well as the MIT Lincoln Laboratory compute cluster, LLGrid.

1.3 Sponsor

MIT Lincoln Laboratory, founded in 1951, is known for its ground breaking work in radar. The Laboratory was commissioned to enhance the United States air defense. Today, there is still a heavy focus on radar and related fields at MIT Lincoln Laboratory. Group 105, the group supporting this MQP, focuses on airborne radar systems and techniques. Recently, group 105 has increased their focus on advanced maritime radar processing.

2 Non-Scanning, One-Dimensional Ocean Radar Scattering Simulation

The first deliverable in this project was a non-scanning radar simulation which interrogates a one-dimensional ocean. A visual representation of this simulation is shown in Figure 9. The simulation features up and down sea propagation as well as targets with initial distance, velocity, and acceleration. The non-scanning, one-dimensional ocean case is the most fundamental simulation and can be used as a toolbox for future small boat detection algorithm improvement. This simulation is extended in Chapters 3 and 4 for the case of a two-dimensional ocean.

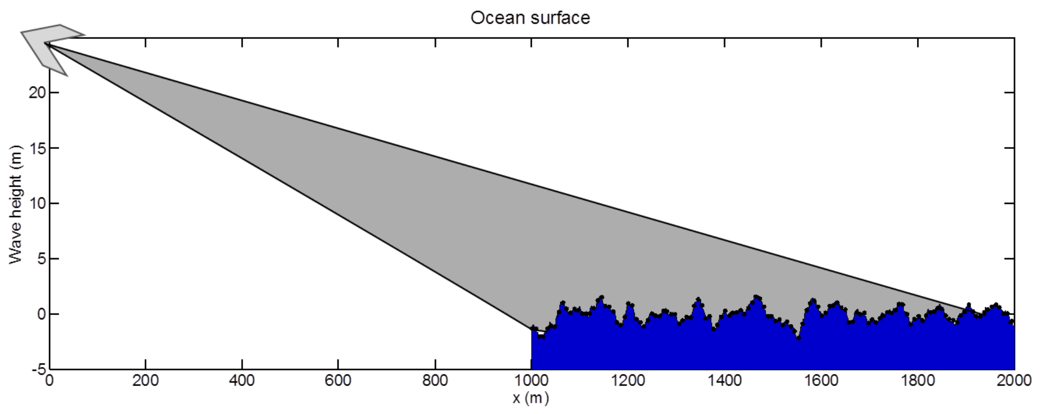


Figure 9. Visual representation of non-scanning radar, one-dimensional ocean, and target integration.

2.1 Background

This subsection focuses on the background necessary for the implementation of the first deliverable. Radar fundamentals are first discussed followed by the relevant principles of oceanography. Finally, we discuss the background necessary to integrate radar and ocean concepts into a single model with realistic outputs.

2.1.1 Radar.

Radar, which is an acronym for RAdio Detection And Ranging, refers to a target detection system that has been used since World War II to determine the range, velocity, azimuth, and elevation of objects. Specifically, radar uses electromagnetic energy to detect targets of interest, which include aircraft, vehicles, people, and even the natural environment. Today, radar technology is used in applications ranging from automotive parking devices to monitoring the potential launch of guided missiles. Figure 10 shows just one of these applications.

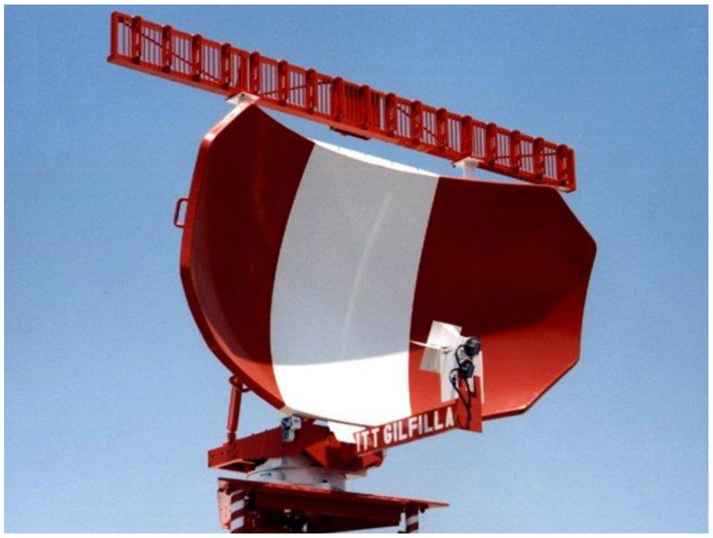


Figure 10. Terminal Airfield Surveillance Radar. An example of a radar used for airport surveillance (Wolff, 2012).

One of the primary reasons that radars are so prevalent in our society is that radars are resistant to adverse weather conditions that affect competing technology such as optical and infrared sensors. In this project, we focused on the simulation of small boats in a maritime environment. As such, radar is essential because of the relatively high frequency of occurrence of unfavorable weather conditions. In fact, there are "no competitive techniques that can accurately measure long ranges in both clear and adverse weather as well as can radar" (Skolnik M. , 2001, p. 2).

Shown in Figure 11, the general principle of radar operation is the same for all radars. An important component is the antenna, which can act as a transmitting antenna or a receiving antenna. Often, a single antenna is used on a time shared basis to perform both functions. First, electromagnetic energy is radiated into space from a transmitting antenna. Next, some of the energy reaches a target, where a portion of the energy is reradiated, or scattered, back toward the radar. This back-scattered energy is referred to as the echo signal. Then, a fraction of the echo signal is collected by the receiving antenna. Finally, the received energy is processed based on the end requirement of the user (Toomay & Hannen, 2004, pp. 3-4).

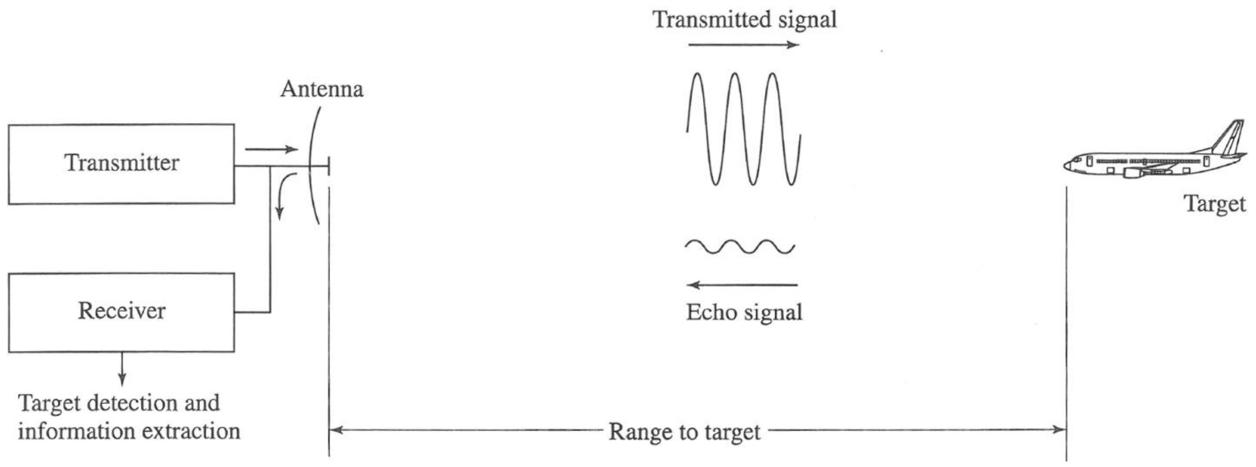


Figure 11. General principle of radar operation. A signal is transmitted and a reflected echo signal is received (Skolnik M. , 2001, p. 2).

The radar antenna is characterized by an antenna pattern, also called the radiation pattern. This characteristic refers to the angular dependence of the signal strength of transmitted and returned echo signals (Toomay & Hannen, 2004, p. 19). Antenna patterns are generally used to describe the antenna hardware that a radar system uses. These patterns will be discussed in further detail in Section 4.1.

Another important characteristic of radar antennas is the polarization used for the transmitted electromagnetic wave. This polarization is defined as the orientation of the electric field. Although many polarizations exist, including linear, circular, and elliptical, “most radar antennas are linearly polarized” (Skolnik M. , 2001, p. 545). Furthermore, linear polarization is most common for maritime radar applications. Two linear orientations are important to maritime radar: vertical, denoted as VV, and horizontal, denoted as HH. Polarization is particularly important for radar scattering from the ocean surface, next discussed in Section 2.1.3.

2.1.1.1 Chirp signals and pulse trains.

With such a wide variety of capabilities, there exist several different types of radar. These types are best differentiated by the number and location of antennas and the type of waveform radiated. When comparing these types of radar, there is always a tradeoff between complexity and capability. Perhaps one of the largest differences is between Continuous Wave (CW) radar and pulse train radar. CW radar sends a constant, relatively low power signal for a long duration, while pulse radar sends short-duration, relatively high power signals repeatedly. Although CW radars are relatively easy and cheap to implement, they lack the ability to detect range easily.

Pulse train radar does not have this limitation; thus, it is more suitable for the maritime environment. Pulse train radar often uses a monostatic antenna configuration; that is, the transmit and receive functions are time shared with a single antenna. In the maritime environment, this configuration is often advantageous because the phase centers of the transmit and receive antennas are the same, which leads to more accurate measurements.

As shown by the black dark bold line in Figure 12, a pulse train is a periodic rectangular envelope waveform in time. In this project, we considered a pulse train to be a sequence of pulsed waveforms. The period of the pulse train is an important parameter which is equal to the reciprocal of the pulse repetition frequency (PRF). Another important parameter for pulse train radar applications is the bandwidth of the pulsed waveform, which has significant impact on range resolution, as discussed later. Pulse trains are particularly important to radar because "it is a convenient method of increasing the signal duration without a proportionate decrease in the bandwidth" (Rihaczek, 1969, p. 287). Increased signal duration is beneficial because it is directly related to increased total energy on scattering targets, which increases detection performance. The negative ramifications of a decrease in bandwidth are discussed in the following paragraphs.

When describing pulse train radar systems and associated processing algorithms, it is very important to distinguish between slow and fast time. Figure 12 helps to show the difference. Slow time refers to pulse-to-pulse time; whereas, fast time refers to inter-pulse time.

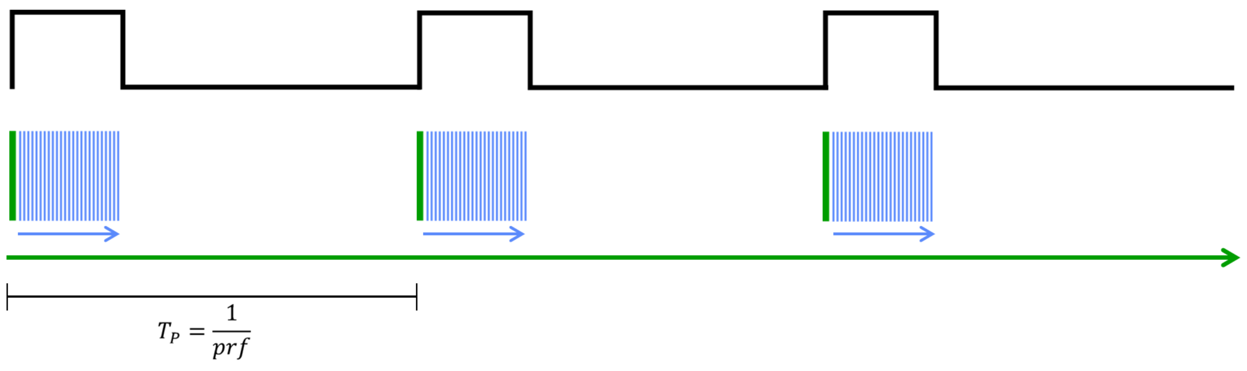


Figure 12. A simple pulse train. Thicker, green lines represent slow time and thinner, blue lines represent fast time. Pulse width, T_P is the inverse of pulse repetition frequency.

Radar systems that use pulse trains can use a variety of waveforms. The most important specification for such pulsed waveforms is the bandwidth of the signal as range processing is not possible with waveforms that have infinitesimally small bandwidth. For example, Figure 13a depicts a signal with infinitesimal bandwidth. It is impossible to determine with certainty the center point, or time origin, of this sinusoidal waveform; thus, it would not be suited for radar applications. On the other hand, a filled frequency spectrum waveform, shown in Figure 13b, would be ideal for radar range processing because there is a non-repeating waveform with most of the power contained within a short time interval. In addition, such peaks are advantageous because they allow better separation of two returns, which is related to the time resolution of signals:

$$t_{resolution} = \frac{1}{B} \tag{2 - 1}$$

where B represents the bandwidth of the signal. Unfortunately, "such broadband signals are entirely impractical to transmit" because of the DC component and spectral requirements (Rihaczek, 1969). A practical signal with similar unambiguous properties is a narrowband signal, shown in Figure 13c. The time function of this signal is amplitude modulated, and the envelope, represented by the dotted line, can be used for detection after demodulation.

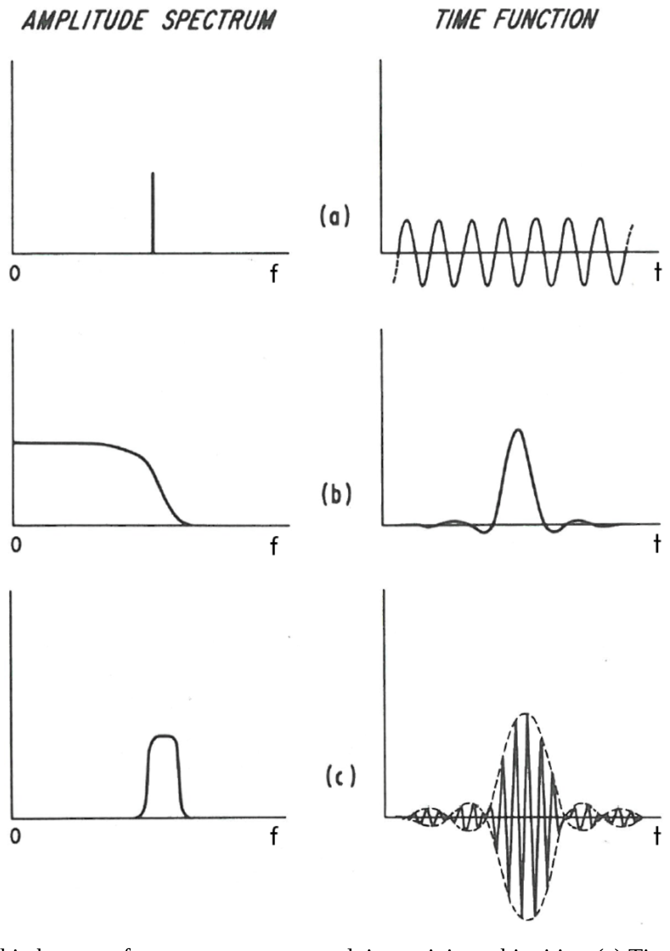


Figure 13. Relationship between frequency spectrum and time origin ambiguities. (a) Time origin ambiguity of a single spectral line; (b) lack of time origin ambiguity by means of a filled spectrum; (c) lack of time origin ambiguity of a narrow-band frequency spectrum (Rihaczek, 1969, p. 4).

For this project, we consider only the chirp waveform for pulse train radar because it is best suited for the maritime environment (Skolnik M. , 2001, p. 341). A chirp signal is a frequency modulated sinusoidal signal, as shown in Figure 14c and as described by Equation (2 - 2):

$$c(t) = \sin\left(2\pi\left(f_i + \frac{f_{rate}}{2}t\right)t\right) \qquad (2 - 2)$$

where f_i is the starting frequency and f_{rate} is the rate of frequency modulation expressed by:

$$f_{rate} = \frac{B}{T} \qquad (2 - 3)$$

such that T is the duration of the chirp signal. Note that the frequency spectrum ranges from f_i to $f_i + B$. In this report, we use f_0 to describe the center frequency of such a signal such that $f_i + B$ equals $f_0 + \frac{B}{2}$. Figure 14a shows that the amplitude of a chirp signal is constant over time. In addition, the figure shows the relationship between start time, end time, and duration. Figure 14b shows the linear frequency modulation in time of the chirp signal. Although these signals can be modulated in other ways (quadratic and exponential modulation is common), we use linear modulation in this project. If f_1 and f_2, the bandwidth of the chirp, are chosen sufficiently close to each other, the bandwidth of the signal satisfies the aforementioned narrow-band frequency spectrum that is desired (as in Figure 13c). The time function in Figure 14c does not suggest easy distinguishability of the time origin. A common practice is to use pulse compression to improve distinguishability (Skolnik M. , 2001, p. 342). Pulse compression works by advancing lower frequency content and delaying higher frequency content. The result is shown in Figure 14d, and is clearly unambiguous.

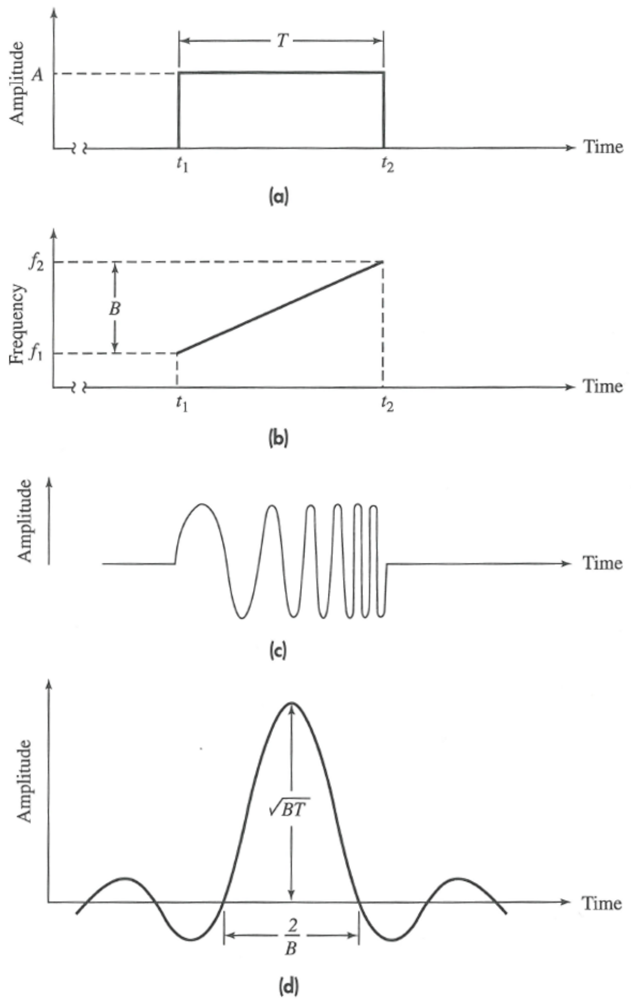


Figure 14. Linear chirp wave characteristics. (a) Amplitude-time dependence; (b) frequency-time dependence; (c) representation of waveform; (d) pulse compressed chirp waveform (Skolnik M. , 2001, p. 344).

2.1.1.2 Non-Scanning radar and processing techniques.

In this report, we first investigate the case for non-scanning radar. In this context, we use this term to describe monostatic chirp waveform pulse train radar with uniform antenna pattern. In a monostatic radar system, the transmit and receive antennas are at the same, fixed location, which implies that the total travel distance z of any signal from the transmitter to the target to the receiver is:

$$z = 2r \tag{2 - 4}$$

where r is the one-way range to target (see Figure 11). A uniform antenna pattern is a constant amplitude radiation pattern with no dependence on angle, which implies that the non-scanning radar has no ability to resolve targets in azimuth. On the other hand, the non-scanning radar is capable of range and velocity detection through range and Doppler processing, respectively. For example, consider two targets 5 km away from a non-scanning radar which are both travelling toward the radar at the same speed where one target is at 0 degrees relative to the radar and the other is at 10 degrees relative to the radar. Both targets will be located in the same range and velocity bins; therefore, the targets are not resolved.

Using radar, range can be found based on the time difference between transmission of the signal and the arrival of the echo. This process is illistrated in Figure 15. Electromagnetic waves are non-dispersive in air; therefore, they travel at the same speed. The symbol c, known as the speed of light, has a value of $299792458\ m/s$, which is the speed of electromagnetic waves in a vacuum. Using simple kinematics, and substituting Equation (2 - 4),

$$r = \frac{ct}{2} \qquad (2 - 5)$$

where t is the elapsed time between transmission and reception of the radar signal. The notion of time resolution discussed in the previous section can be extended to range resolution by substituting t in Equation (2 - 5) with $t_{resolution}$ in Equation (2 - 1):

$$r_{resolution} = \frac{c}{2B} \qquad (2 - 6)$$

Radar engineers typically use the term “range bin” to specify the interval of physical ranges that map to a specific discrete range.

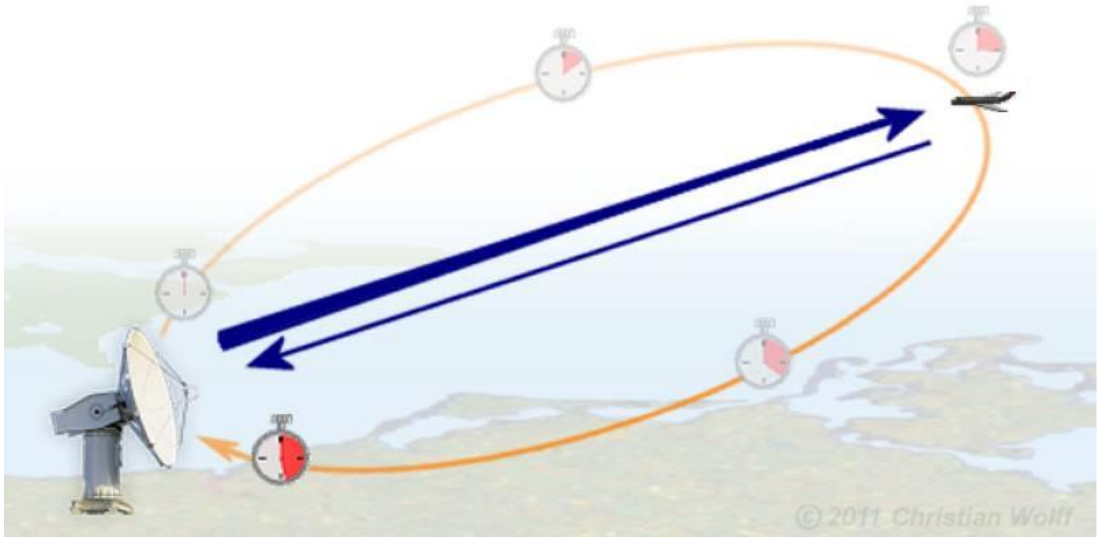


Figure 15. Range processing (Wolff, 2012).

In range processing, there exists an unambiguous range over which the received signal is associated with the transmitted pulsed radar signal. In maritime radar, relatively shorter ranges (less than 20km) are expected, and range will generally alias in discrete Fourier processing (discussed in Section 2.2.1.2) before it becomes theoretically ambiguous. The aliasing time is (Lathi, 2005, p. 542):

$$t_{alias-free} = \frac{1}{\Delta f} \tag{2 - 7}$$

where Δf is the sample interval of the frequency domain. Similarly, the notion of alias-free range can be extended with substitution of Equation (2 - 5):

$$r_{alias-free} = \frac{c}{2\Delta f} \tag{2 - 8}$$

These concepts are similarly discussed by Skolnik (2001, p. 2).

In addition to range information, velocity information is another useful result of radar processing. Doppler processing is the method used to determine the velocity of moving targets. This process is performed over multiple pulses, thus over slow time. The central theory underlying Doppler processing is the Doppler shift, which is the change in frequency of a wave due to relative motion. In this case, the Doppler shift is not measured relative to the previously discussed chirp waveforms. Instead, the Doppler shift is measured according to the shifted pulse train frequency. This change of frequency can be determined by finding the rate of change of phase (Skolnik M. , 2001, p. 105). A derivation of Doppler shift follows: The round trip total phase change φ of the pulsed signal with wavelength λ for a target at range r is

$$\varphi = 2\pi \cdot \frac{2r}{\lambda} = \frac{4\pi r}{\lambda} \tag{2 - 9}$$

The rate of change of phase rate, as described, is equal to Doppler angular frequency:

$$\omega_d = \frac{d\varphi}{dt} = \frac{4\pi}{\lambda} \cdot \frac{dr}{dt} = \frac{4\pi v_r}{\lambda} = 2\pi f_d \tag{2 - 10}$$

where f_d is the Doppler frequency shift and v_r is the radial velocity. Rearranging Equation (2 - 10):

$$f_d = \frac{2v_r}{\lambda} = \frac{2f_i v_r}{c} \tag{2 - 11}$$

For radar, Doppler shift is the measured quantity and velocity is the desired result:

$$v_r = \frac{f_d c}{2f_i} \tag{2 - 12}$$

Positive velocities represent targets moving toward from the radar; inversely, negative velocities represent targets moving away the radar.

Similar to range processing, resolution and alias-free range are important to Doppler processing. The velocity resolution is given by:

$$v_{resolution} = \frac{c}{2f_0T_{dwell}} \qquad (2 - 13)$$

where the signal frequency spectrum ranges from $f_0 - \frac{B}{2}$ to $f_0 + \frac{B}{2}$ and T_{dwell} is the time on target, in other words, the interval of slow time over which Doppler processing is computed. When discretized, velocity resolution can be expressed as

$$v_{resolution} = \frac{c \cdot prf}{2f_0N} \qquad (2 - 14)$$

where N is the number of pulses in the sample of time. The alias-free velocity can be determined by:

$$v_{alias-free} = \pm\frac{c \cdot prf}{4f_0} \qquad (2 - 15)$$

The full alias-free range is described by $c \cdot prf/(2f_0)$, but it is common to split this range into positive and negative velocities.

One very important approximation that is used for modeling Doppler response is the quasi-static approximation. This approximation allows the assumption that the target does not move during inter-pulse time (Griffiths, 1999). Consider a target moving toward a radar. Electromagnetic energy is sent from the transmitting antenna at some time t_0. At this time, the target is at range r. Because the energy is travelling at the speed of light, it takes a fraction of a second to reach the target; call this time t_ε. Of course, the target has been moving during this fraction of a second; called the radial distance travelled r_ε. In this scenario, the result of range processing would indicate that the target is at range $r - r_\varepsilon$ at time $t + t_\varepsilon$. Using this information, it is possible to modify Equation (2 - 5). However, this adjustment is not necessary in our simulation because r_ε is insignificantly small, and therefore, has negligible effects. Using the quasi-static approximation, we can model the range of a moving target with Equation (2 - 5).

2.1.2 One-dimensional ocean model.

The ocean is a very large and complex fluid system that is still not fully understood by experts in the field. There are many instances of energy transfer into the ocean from weather patterns, earthquakes, boats, and various other ocean disturbances. The nonlinear nature of these disturbances makes the system very complex and difficult to model. However, there is a general understanding and knowledge of fluid mechanics and oceanography that can be used to describe a basic ocean. Ocean waves have a significant effect on radar returns, and therefore, are important to our model. Work toward this model started with research into oceanography and ocean sea states.

Unlike electromagnetic waves in air, deep ocean waves are dispersive, that is, the velocity of ocean wave propagation is not constant for different frequencies or wavelengths. The dispersion of ocean waves depends on the depth of the ocean. Specifically, as the depth increases, the water waves are more dispersive. In this project, we consider the deep water dispersion case which requires that the depth of the ocean be greater than half of one wavelength of the largest ocean wave. Additionally, we use the term phase velocity to describe the velocity at which the crest of an ocean wave travels.

The dispersion relation for deep water waves is described by Equation (2 - 17) where k_d is the wave number for dispersive waves, or spatial frequency, described by Equation (2 - 16), g is the acceleration of gravity, and λ is the wavelength of a wave. These equations show that angular frequency of an ocean wave has a nonlinear relation to wavelength, and is proportional to acceleration of gravity.

$$k_d = \frac{2\pi}{\lambda} \tag{2 - 16}$$

$$\omega^2 = k_d g = \frac{2\pi g}{\lambda} \tag{2 - 17}$$

The general velocity of a wave can be described as the product of the wavelength and the frequency as in (2 - 18). From this equation, we can derive the equation for phase velocity of a dispersive wave at a given angular frequency, wave number, or wavelength, as shown in Equation (2 - 19).

$$v = \lambda f \tag{2 - 18}$$

$$v_{phase} = \frac{g}{\omega} = \sqrt{\frac{g}{k_d}} = \sqrt{\frac{\lambda g}{2\pi}} \tag{2 - 19}$$

Equation (2 - 19) shows that the phase velocity of a given ocean frequency is inversely proportional to the angular frequency. Squaring both sides of Equation (2 - 19) shows that wavelength is proportional to the phase velocity squared.

With the aid of these equations, we can now emphasize the importance of the dispersion relation. Figure 16 shows two graphs: a graph of the phase velocity versus wavelength of an ocean wave and a graph of the angular frequency versus wavelength. In Figure 16a, notice that the phase velocity of different waves is not constant, in opposition to the electromagnetic wave case in air. Figure 16b illustrates that angular frequency is inversely proportional to the square root of the wavelength. These plots highlight that for a very short wavelength, the wave will have a relatively low phase velocity and high angular frequency, as one would expect.

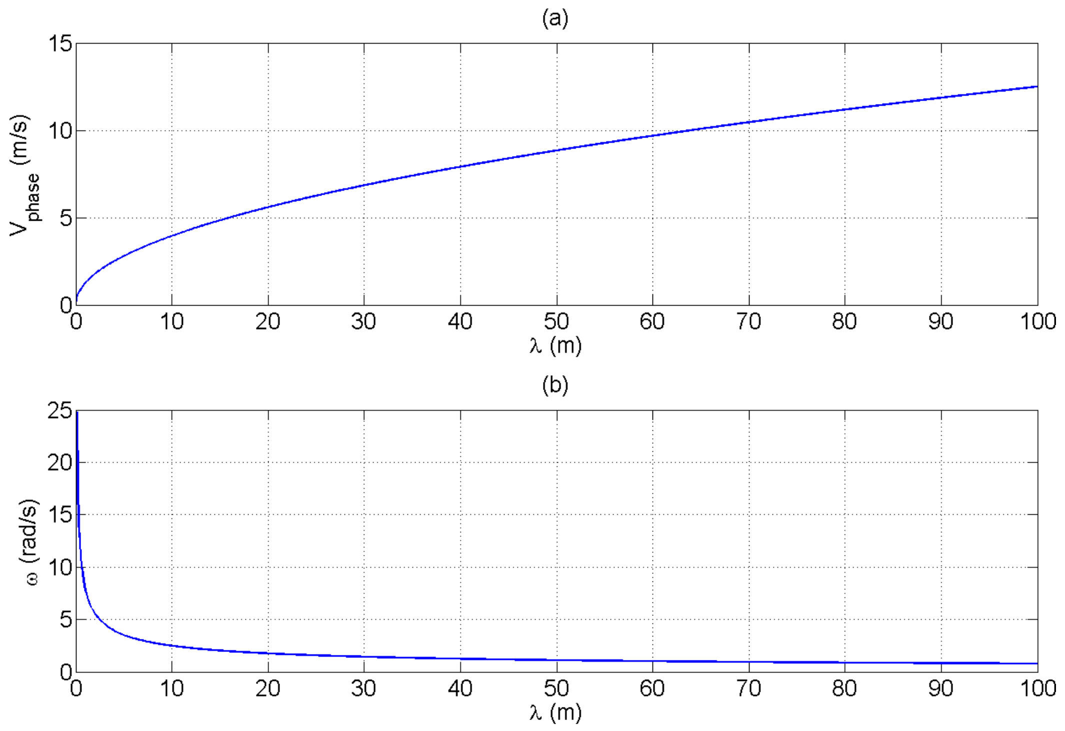


Figure 16. Graphical representation of the dispersion relation. Figure 16(a) shows the phase velocity vs. wavelength of an ocean wave. Figure 16(b) shows the angular frequency vs. wavelength of an ocean wave.

Ocean waves are created primarily by a wind blowing over the ocean. There is an energy transfer from the wind to the ocean that varies with wind speed. The energy transfer also depends on the length and width of ocean over which the wind is blowing. The fetch is the length over which the wind blows. The duration of time over which the wind blows is another factor that affects the ocean waves. The duration is important in developing a sea, as longer times allow for more energy to be transferred into the sea. A fully developed sea occurs when the energy transfer reaches equilibrium with energy losses due to friction and other internal losses. (Kanevsky, 2009, p. 9)

Short wavelength waves, referred to as capillary waves, are created initially when the ocean is subjected to wind. Capillary waves have wavelengths of a few centimeters with frequencies greater than 48 Hz. For longer durations of constant wind speeds, energy is transferred to longer wavelength waves called capillary-gravity waves with frequencies between 5 and 48 Hz. This process continues to transfer energy into even longer wavelength waves called gravity waves with frequencies less than 5 Hz. Gravity waves can become hundreds of meters long during this process. At some point in time, the sea state will finally be considered a fully developed sea. We only consider the case of a fully developed sea in our model. (Kanevsky, 2009, pp. 14-15)

A fully developed sea can be described by its spectral composition. The Pierson-Moskowitz spectrum was developed in 1964 and is a well-known approximation for the power spectral density of a fully developed sea. The spectrum is an approximation because it is an idealized best-fit curve of empirical results. The Pierson-Moskowitz spectrum is a function of angular frequency at a specified wind speed and is described by the following equations:

$$G(\omega) = 8.1 \times 10^{-3} g^2 \omega^{-5} e^{-1.25\left(\frac{\omega_m}{\omega}\right)^4} \qquad (2 - 20)$$

$$\omega_m = \frac{0.83g}{V_{wind}} \qquad (2 - 21)$$

where V_{wind} is the velocity of the wind, g is the acceleration due to gravity, ω_m is the maximum angular frequency, and ω is the angular frequency (Kanevsky, 2009, pp. 14-15). Figure 17 shows the spectrum resultant from various wind speeds.

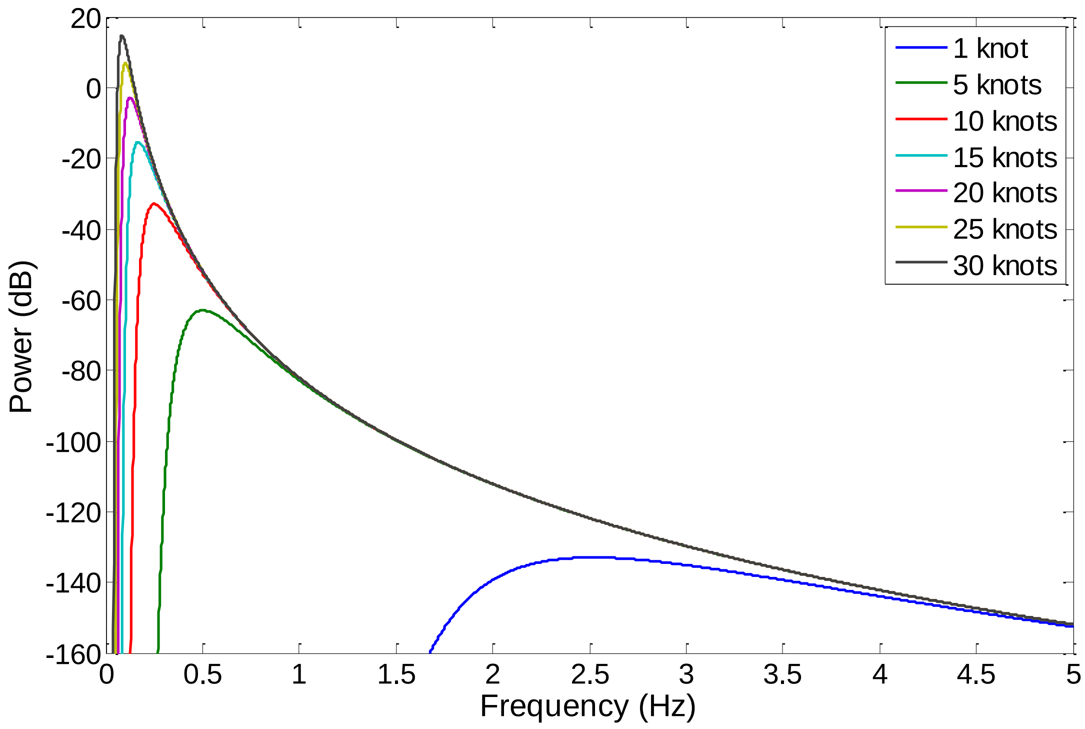


Figure 17. Power spectral density of ocean waves represented by the Pierson-Moskowitz spectrum. Plotted for wind speeds of 1, 5, 10, 15, 20, 25, and 30 knots.

As can be seen in Figure 17, the peak of the spectrum at lower wind speeds has relatively low power and high frequency. As the wind speed increases the peak of the spectrum increases in power while decreasing in frequency. This dependence is expected, as at higher wind speeds the ocean waves are composed of longer wavelength waves with higher amplitudes. (Kanevsky, 2009, p. 15)

The Beaufort wind scale relates the dependence of the ocean surface amplitude on wind speed in deep water. This scale is universally used by the maritime community and has been significantly updated since its creation. The Beaufort scale maps ocean roughness into 13 categories from degree 0 (calm) to degree 12 (hurricane). The National Oceanic and Atmospheric Administration (NOAA) lists degree, description, wind speed, and wave height of the categories in the scale. Table 1, shown below, summarizes this relationship (The Beaufort Wind Scale, 2007).

Table 1. Beaufort wind scale.

Degree	Description	Wind Speed (knots)	Wave Height (m)
0	Calm	0	0

1	Light Air	1-3	0.1
2	Light Breeze	4-6	0.2 - 0.3
3	Gentle Breeze	7-10	0.6 - 1
4	Moderate Breeze	11-16	1 - 1.5
5	Fresh Breeze	17-21	2 - 2.5
6	Strong Breeze	22-27	3 - 4
7	Near Gale	28-33	4 - 5.5
8	Gale	34-40	5.5 - 7.5
9	Strong Gale	41-47	7 - 10
10	Storm	48-55	9 - 12.5
11	Violent Storm	56-63	11.5 - 16
12	Hurricane	Over 73	Over 16

Significant wave height (SWH) is used in oceanography as a standard measure of relative wave height. Significant wave height is defined as the mean of the highest one-third of ocean waves measured from crest to trough. It can also be approximated by four times the standard deviation of the ocean surface, as shown in Equation (2 - 22) (Kanevsky, 2009, p. 16):

$$H_S \approx 4\sigma \qquad (2 - 22)$$

The significant wave height can be used to validate the correlation between wind speed and statistical ocean surface height measurements through the Beaufort scale.

2.1.3 Integration and calibration.

As claimed in Chapter 1, the problem of detection of small boats on rough ocean surface is not well understood. However, work has been performed to begin understanding the radar scattering phenomenology of the ocean. A useful term when describing this phenomenology is ocean clutter, which describes the radar return of the ocean. In Section 2.2.2, techniques for calibrating the significant wave height of the ocean to known heights using the Beaufort scale were presented. In the exact same manner, there exists scattering characteristics from collected data to which it is possible to calibrate simulated radar scatter.

One major characteristic suggested by ocean scatter data is the correlation between radar return signal strength and the slope of the ocean. More specifically, the power of radar returns is directly proportional to the ocean wave slope. This phenomenon can be most simply described by ray optic theory. In the maritime environment, radars are elevated and angled down toward the ocean. Another characteristic descriptor of maritime radar applications is a relatively low grazing angle, which is the angle of incidence of the transmitted electromagnetic waves to a flat ocean surface (Skolnik, 2008, p. 15.9). As shown in Figure 18, when the slope of the ocean is relatively high and positive, a relatively high amount of energy returns to the receiving antenna. On the other hand, when the slope of the ocean is low or negative, less energy returns.

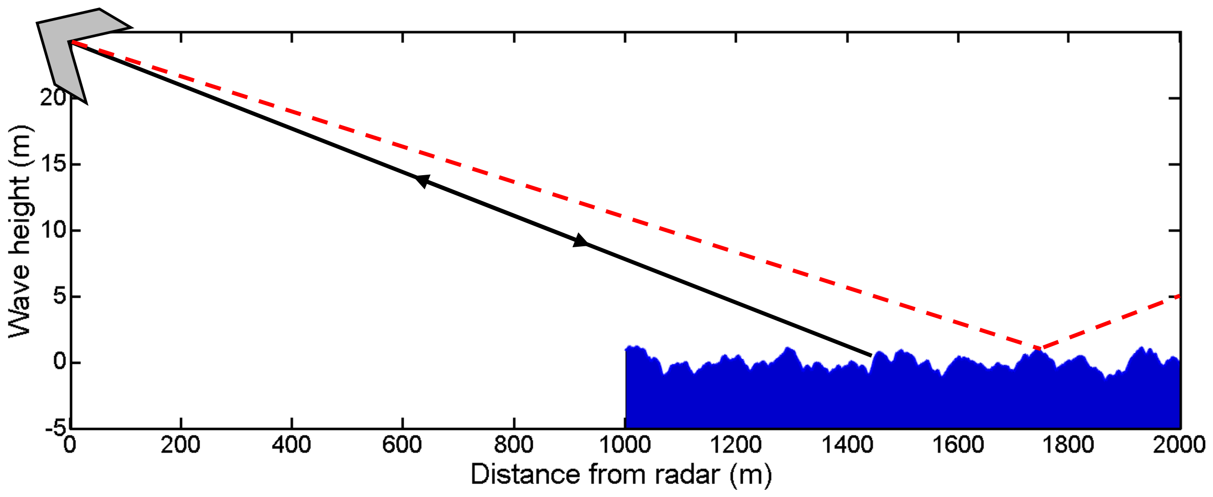


Figure 18. Effect of ray optics on ocean interrogation. Ocean modeled from 1000 to 2000 m. The shaded V represents the location and angle of a radar. The dark, black line represents high back-scatter on a high, positive-sloped section of ocean. The dotted, red line represents low back-scatter on a low, positive-sloped section of ocean.

Radar Cross Section (RCS) is an important parameter for target modeling. RCS, denoted as σ, helps describe the detectability of a target to radar. A higher RCS indicates a more detectable target. Some factors that affect the radar cross section include the material, the absolute size of the target, the incident angle, the reflected angle, and the polarization. In effect, RCS is a measure of the power of the return signal from a target. It is often beneficial to model the normalized RCS, that is (Skolnik, 2008, p. 15.7):

$$\sigma^0 = \frac{\sigma}{A_f} \qquad (2 - 23)$$

where A_f is the surface area of the interrogated environment. The full ramifications of cross section are out of the scope of our project; thus, we limit our background of the subject to just what is necessary for modeling.

In reality, the ocean is a dynamic system with two-dimensional propagation. For this first deliverable, we model only one-dimensional propagation. Although one-dimensional propagation is a very crude approximation to a two-dimensional model, there are many advantages to such a model. The biggest advantage is that the up-sea and down-sea propagation (one-dimensional propagation toward or away from the radar, respectively) are the most stressing cases for target detection. These directions of propagation are the most stressing cases because the ocean has the highest cross section as compared to other directions of propagation. Therefore, the relatively highest amount of energy is returned to the receiver, and it is more difficult to distinguish a target in higher ocean clutter.

One of the most important characterizations of radar scatter is the distribution of power content. In 1984, Ewell, Tuley, and Horne investigated this distribution and the associated statistics of both VV and HH polarized radar using X-band radar at a grazing angle of five degrees (pp. 100-104). In addition, the compressed pulse length was 10 ns and the radar had 2.4 degrees of azimuth beam width (discussed in Section 4.1). Wind speed ranged from 13-24 knots. Ewell et al. found that the distribution was roughly log-normal, as shown by Figure 19. A log-normal distribution is Gaussian on a logarithmic scale. This distribution is specified by two parameters: standard deviation, σ; and mean, μ.

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{(\ln(x)-\mu)^2}{2\sigma^2}}, \quad x > 0 \qquad (2 - 24)$$

A Rayleigh distribution also approximated the ocean scattering statistics, but the log-normal distribution is more convenient for distribution fitting during modeling. Long (2001), compiled the data from Ewell et al. and performed more detailed statistics, shown in Table 2. In our representation of these data, we assumed a linear relation between wind speed and the mean and standard deviation intervals reported by Long. The probability density functions and cumulative density functions for these data are shown in Figures 20 through 22.

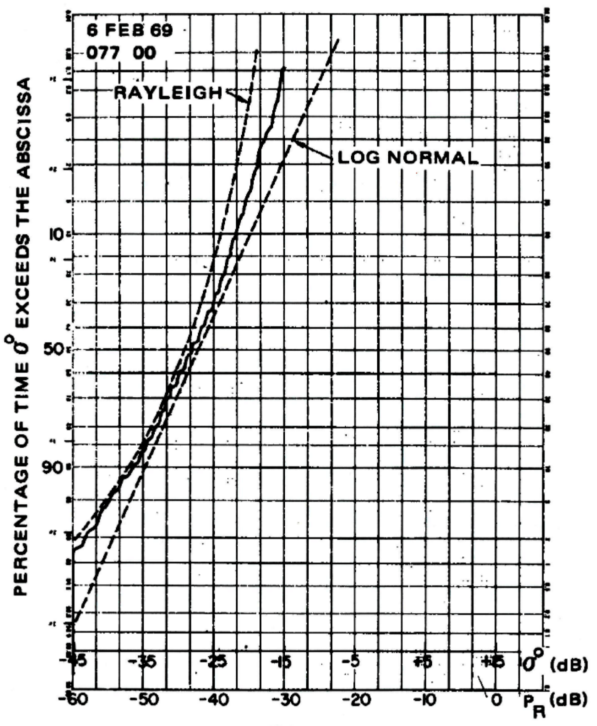


Figure 19. Ocean scattering distribution. The solid line represents the actual data, and the dashed lines represent fitted log-normal and Rayleigh distributions (Long, 2001, p. 239).

Table 2. Log-Normal parameters for ocean scattering at various polarizations and wind speeds (Long, 2001).

Polarization	Wind Speed (knots)	Mean $\sigma^0\ (dB)$	Standard Deviation $\sigma^0\ (dB)$
Vertical	13	-34.3	3.2
	18.5	-29.55	3.35
	24	-24.8	3.5
Horizontal	13	-45.1	6.1
	18.5	-37.7	6.6
	24	-30.3	7.2

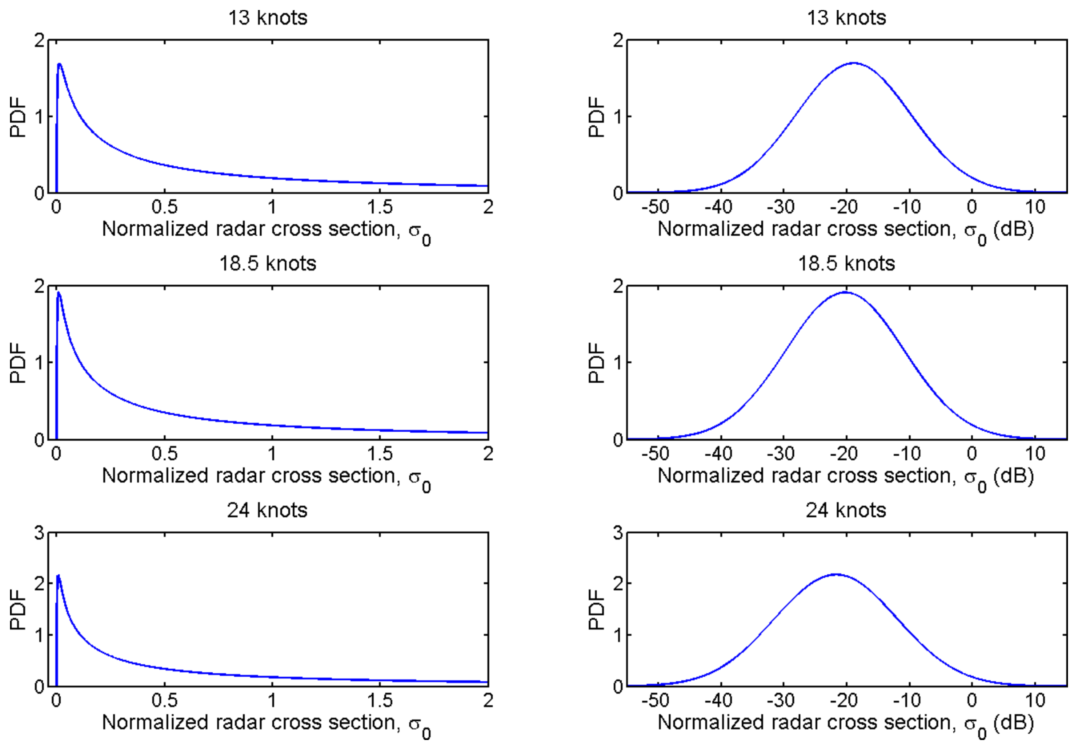


Figure 20. VV polarized log-normal probability density function. The plots on the left show a normal scale and the plots on the right show a log-normal scale.

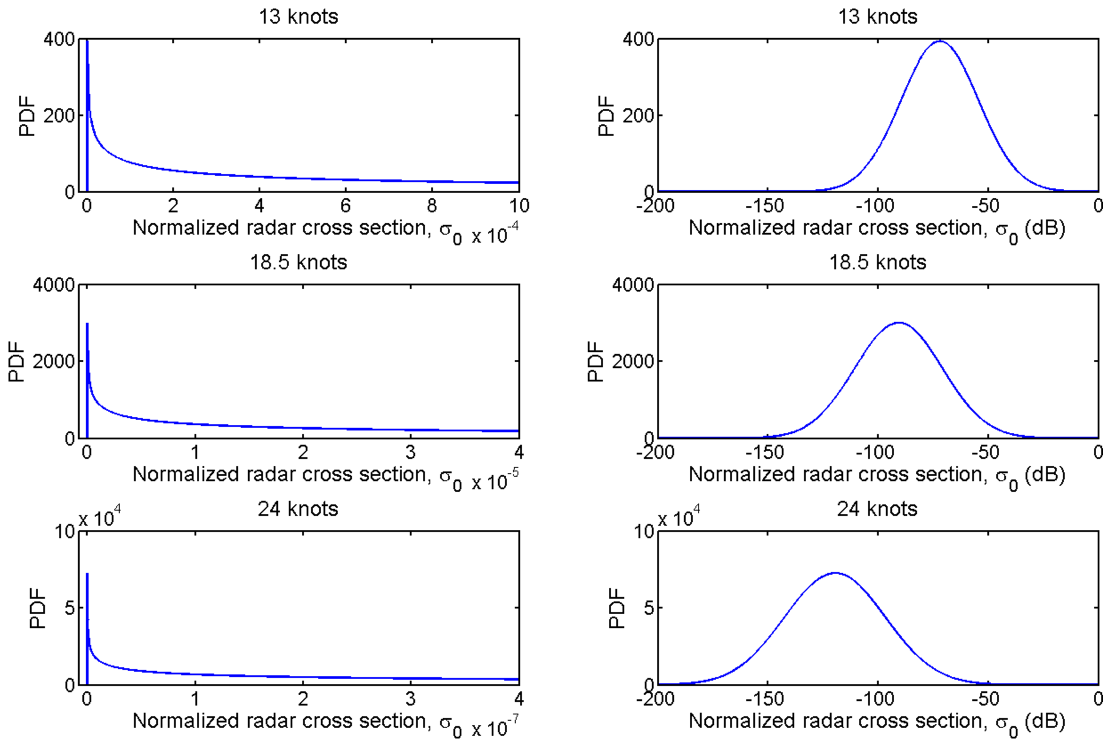


Figure 21. HH polarized log-normal probability density function. The plots on the left show a normal scale and the plots on the right show a log-normal scale.

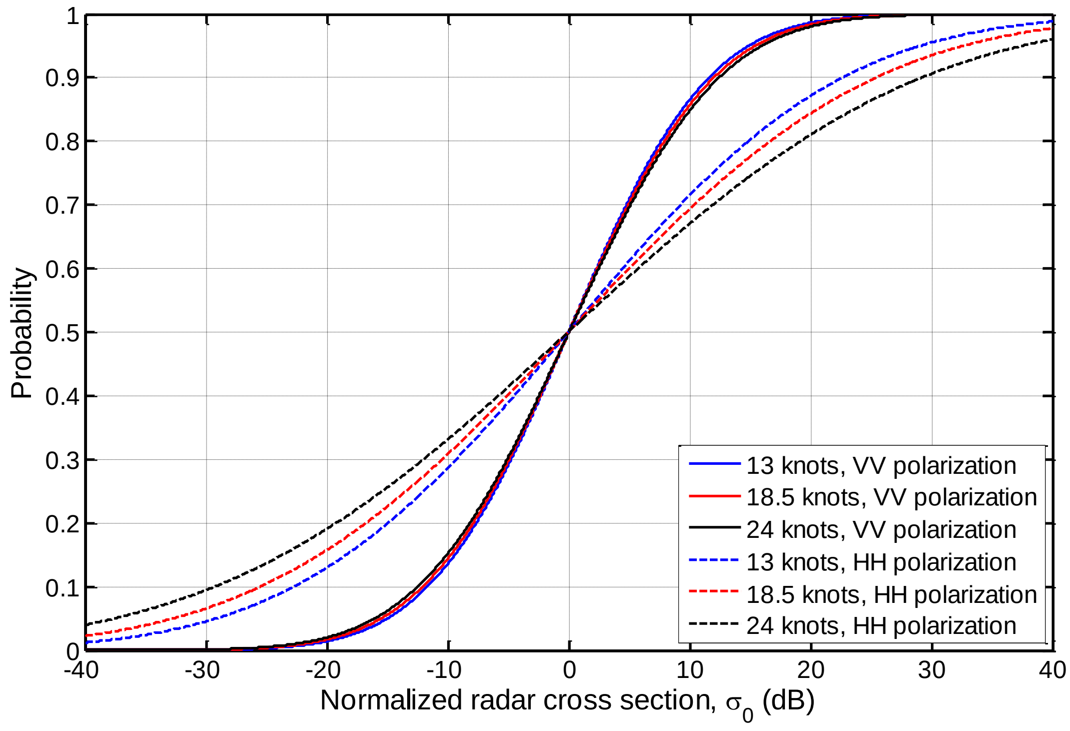


Figure 22. Cumulative distribution functions of log-normal ocean scatting parameters.

2.2 Methods

In this section, we present the implementation of both the radar model and the ocean model in the case of a non-scanning, one-dimensional ocean scattering simulation. We conclude the section with a discussion of how the two component models interact together and a discussion of their combination into a single software simulation package.

2.2.1 Radar.

In this section, we explore the methods of modeling non-scanning radar. We break this discussion into five parts: 1) chirp waveform modeling; 2) target modeling; 3) range processing; 4) Doppler processing; and 5) range-Doppler processing.

2.2.1.1 Chirp waveform modeling.

Recall that chirp waveforms are often used for pulse train radar in the maritime environment. Our goal is to model the returned pulse compressed chirp waveforms. It is possible to model the chirp signal and perform pulse compression using a matched filter. However, a simpler method that is approximately equivalent is to use a stepped frequency representation of the chirp waveform whose matched filter is an inverse Fourier transform. First, the time and frequency representations of a chirp signal with center frequency $20\,GHz$ and bandwidth

$20\,GHz$ were plotted, as shown in Figure 23. Such a large bandwidth was chosen only for demonstration purposes. Because the signal is modeled discretely, the chirp signal is accurately termed a stepped frequency waveform. Specifically, the discrete implementation implies that the frequency vs. time relation in Figure 14b is a step function. In MATLAB, Figure 23a was computed using discrete time and Equations (2 - 2) and (2 - 3). Figure 14b was generated by taking the discrete Fourier transform (DFT). In this paper, all DFTs were implemented using a fast Fourier transform (FFT).

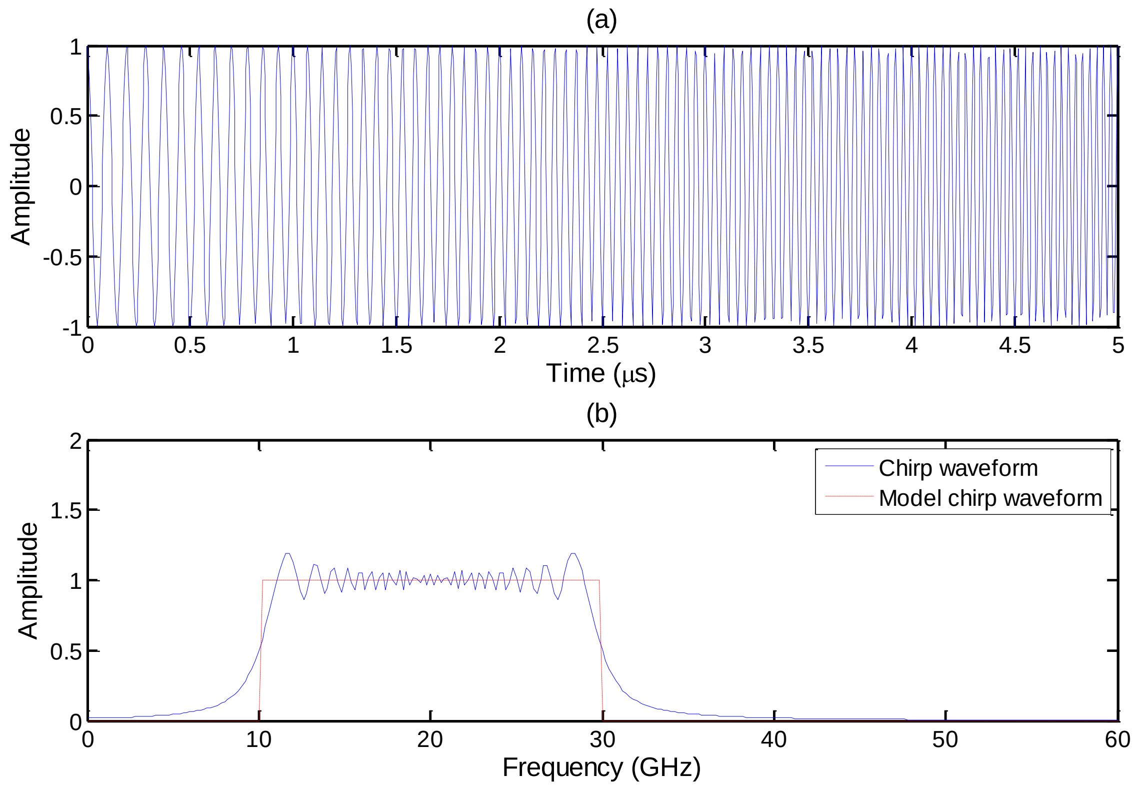


Figure 23. Time and frequency representation of $10 - 30\ GHz$ chirp waveform. (a) Shows the time representation and (b) shows the single-sided frequency representation. In (b), the blue, solid line represents the single-sided frequency domain of the chirp waveform and the red, dashed line represents what we use to model a pulse compressed chirp waveform.

A stepped frequency waveform was modeled as a frequency shifted $gate$ function (a $rect$ function). Pulse compression for the stepped frequency waveform is achieved via an inverse Fourier transform. The inverse Fourier transform of a $gate$ function starting at ω_1 and ending at ω_2 is an amplitude scaled, modulated, normalized $sinc$ function, as denoted by Equation (2 - 25) and derived in Appendix A.1. This type of time domain signal is exactly the desired form shown in Figure 14d.

$$c_{pc}(t) = \mathcal{F}^{-1}[u(\omega-\omega_1) - u(\omega+\omega_2)] = B \cdot e^{j\omega_0 t_f} sinc(Bt_f) \qquad (2\text{ - }25)$$

Where $c_{pc}(t)$ is the pulse compressed chirp signal, ω is the angular frequency, ω_0 is the center angular frequency ($\omega_0 = \frac{\omega_2-\omega_1}{2} = 2\pi f_0$), and t_f represents fast time. Due to the simplicity of this approximation, it is more convenient to represent and discuss the chirp signal in the frequency domain. Moreover, because the chirp signal is real-valued, the conjugate symmetry property holds (Lathi, 2005, p. 682):

$$X(-\omega) = X^*(\omega) \qquad (2\text{ - }26)$$

The conjugate symmetry property allows us to represent the pulse compressed chirp waveform simply with a single-sided frequency spectrum.

2.2.1.2 Range processing.

Recall that range processing is predicated on the time difference of transmitted and received radar signals, and serves the purpose of finding the range response. To accomplish range processing, it was first necessary to model the time delay of chirp signals. Modeling the time delay was performed by using a time shifting transfer function in the frequency domain (Lathi, 2005, p. 705):

$$f(t_f - t_0) \Leftrightarrow F(\omega)e^{-j\omega t_0} \qquad (2\text{ - }27)$$

where t_0 is the time shift in fast time, f is a generic function, and F is the Fourier transform of f. In this case, we know that the time shift is defined by the range to a target. Solving Equation (2 - 5) for t and substituting in Equation (2 - 27):

$$f\left(t_f - \frac{2r}{c}\right) \Leftrightarrow F(\omega)e^{-2j\omega r/c} = F(\omega)e^{-2jkr} \qquad (2\text{ - }28)$$

Recognize that the non-dispersive wavenumber, $k = \omega/c$, is a function of ω, but we leave the radar return equation in terms of k for simplicity. We can now apply this time shift function to chirp waveforms. The chirp radar range processing system model is represented in Figure 24, where $C_{pc}(\omega)$ represents the Fourier transform of $c_{pc}(t)$.

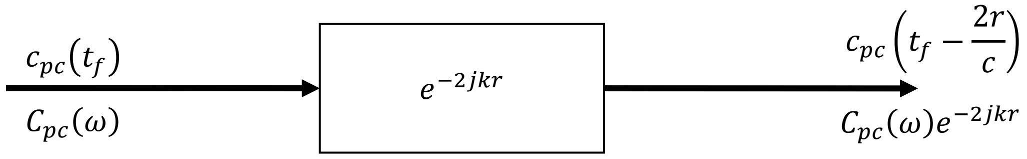

$c_{pc}(t_f)$
$C_{pc}(\omega)$
e^{-2jkr}
$c_{pc}\left(t_f - \frac{2r}{c}\right)$
$C_{pc}(\omega)e^{-2jkr}$

Figure 24. Representation of the range processing system. The “black box” represents a system with the indicated transfer function.

We denote the (discretized) radar return of a stationary point target at range r as

$$V[\omega] = C_{pc}[\omega]e^{-2jkr} \qquad (2 - 29)$$

To further simplify processing, we note that if we limit our consideration of the frequency domain to only the range from ω_1 to ω_2, then $C_{pc}[\omega] = 1$ and Equation (2 - 29) simplifies to:

$$V[\omega] = e^{-2jkr}, \quad \omega_1 \leq \omega \leq \omega_2 \qquad (2 - 30)$$

In this consideration, no information or accuracy is lost because the amplitude spectrum is 0 outside of this range. In every following radar return equation in this paper, we assume $\omega_1 \leq \omega \leq \omega_2$

All values must be discretized and wave number must be sampled in order to model the radar returns in MATLAB. Wave number was approximated by equal space sampling the angular frequency spectrum N times, then dividing by c. Given the maximum desired alias-free range and radar bandwidth, N can be found according to:

$$N = \left\lceil \frac{B}{\Delta f} \right\rceil + 1 = \left\lceil \frac{2B \cdot r_{alias-free}}{c} \right\rceil + 1 \qquad (2 - 31)$$

The next step in range processing was to find the time response of the radar return. The analytical solution can be easily extended starting with Equation (2 - 25) and applying Equation (2 - 27):

$$c\left(t_f - \frac{2r}{c}\right) = \mathcal{F}^{-1}[V(\omega)] = B \cdot e^{j\omega_0\left(t-\frac{2r}{c}\right)} sinc\left(B\left(t_f - \frac{2r}{c}\right)\right) \qquad (2 - 32)$$

In modeling, the determination of time response of the radar return from the frequency representation is performed using the inverse discrete Fourier transform (IDFT). In implementation, we used the inverse fast Fourier transform (IFFT). It is this underlying discrete Fourier transform that causes aliasing discussed in Section 2.1.1.2. To efficiently use the IFFT, and additionally gain interpolation, the radar return vector is zero padded to the next power of 2 above the size of the radar return vector.

The resulting function from the previous step is available in the fast time domain. We convert from (fast) time to range using (2 - 5). In summary, the processing flow is depicted in Figure 25. Note that we represented the power in dB and scaled the maximum power to 0 dB.

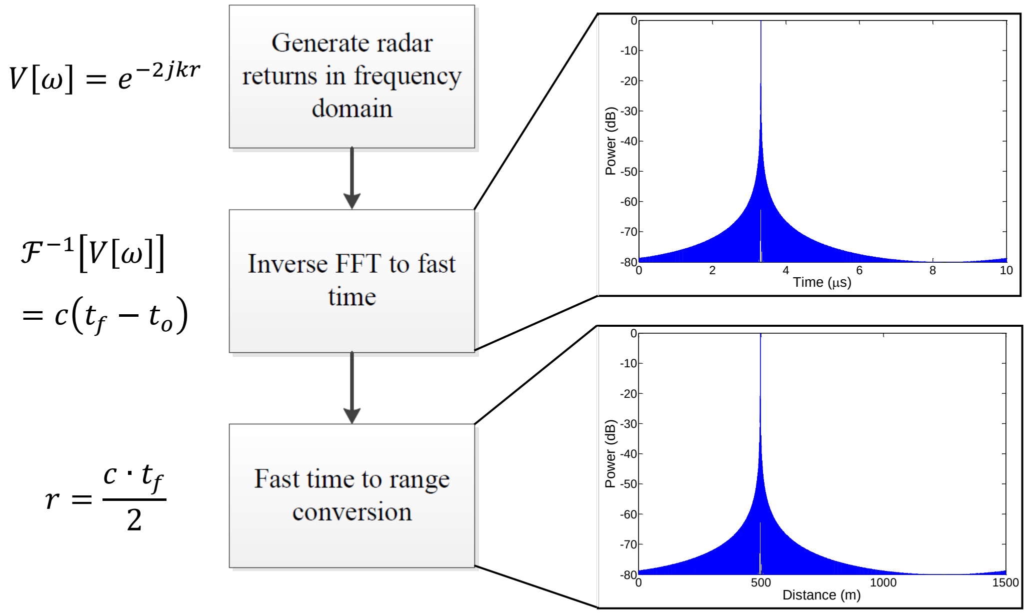

$V[\omega] = e^{-2jkr}$
Generate radar returns in frequency domain
$\mathcal{F}^{-1}[V[\omega]] = c(t_f - t_o)$
Inverse FFT to fast time
$r = \frac{c \cdot t_f}{2}$
Fast time to range conversion
Power (dB)
Time (µs)
Distance (m)

Figure 25. Work flow diagram of range processing. (top) Time response and (bottom) range response of target $500\ m$ away from radar.

The process just described always produces a range response that starts at 0 range and extends to the alias-free maximum range. A caveat to range processing is that it can become very computationally intensive if high alias-free maximum range is desired because N becomes large. This caveat can be avoided if a start range of 0 is not required by multiplying by an extra time shift transfer function in the radar return:

$$V[\omega] = e^{-2jkr} e^{-2jk \cdot R_s} \tag{2 - 33}$$

where R_s is the new start range.

Consider the case of two point targets. The radar returns of these targets can be modeled as a sum of each individual radar response:

$$V[\omega] = e^{-2jk \cdot R_s}(e^{-2jkr_1} + e^{-2jkr_2}) \tag{2 - 34}$$

where r_1 and r_2 represent the range of the first and second target, respectively. This concept of multiple targets can be extended where τ is the target index and $\mathrm{N_T}$ is the number of total targets:

$$V[\omega] = e^{-2jk \cdot R_s} \sum_{\tau=1}^{\mathrm{N_T}} e^{-2jkr_\tau} \tag{2 - 35}$$

Now consider two targets that are close to each other in range. The side lobes of the *sinc* function make it harder to distinguish two close targets. For example, consider an attempt to distinguish the two targets in Figure 26 using a threshold detection algorithm. The minimum threshold level that could be chosen is $-13.2\ dB$. However, this minimum threshold level can be lowered through sidelobe reduction (Levanon, 1988, p. 271). Sidelobe reduction is accomplished through frequency domain windowing:

$$V[\omega] = W_{\omega}[\omega] e^{-2jk \cdot R_s} \sum_{\tau=1}^{\mathrm{N_T}} e^{-2jkr_{\tau}} \qquad (2 - 36)$$

where $W_{\omega}[\omega]$ is a window function over angular frequency. A common window that is used in this application is the Hamming window because it significantly reduces sidelobe levels while maintaining much of the main beam width. The example presented in Figure 26 with Hamming windowing is shown in Figure 27. Note that distance on the abscissa is directly related to time. In particular, the maximum sidelobe is $-42.4\ dB$ and the beam width increased by approximately 46%. Kaiser windowing, shown in Figure 28, is also used throughout this project for optimal sidelobe reduction (which is often more important than main beam width). For the Kaiser window, the maximum sidelobe is $-92.4\ dB$ and the beam width has increased by 122% relative to the non-windowing example.

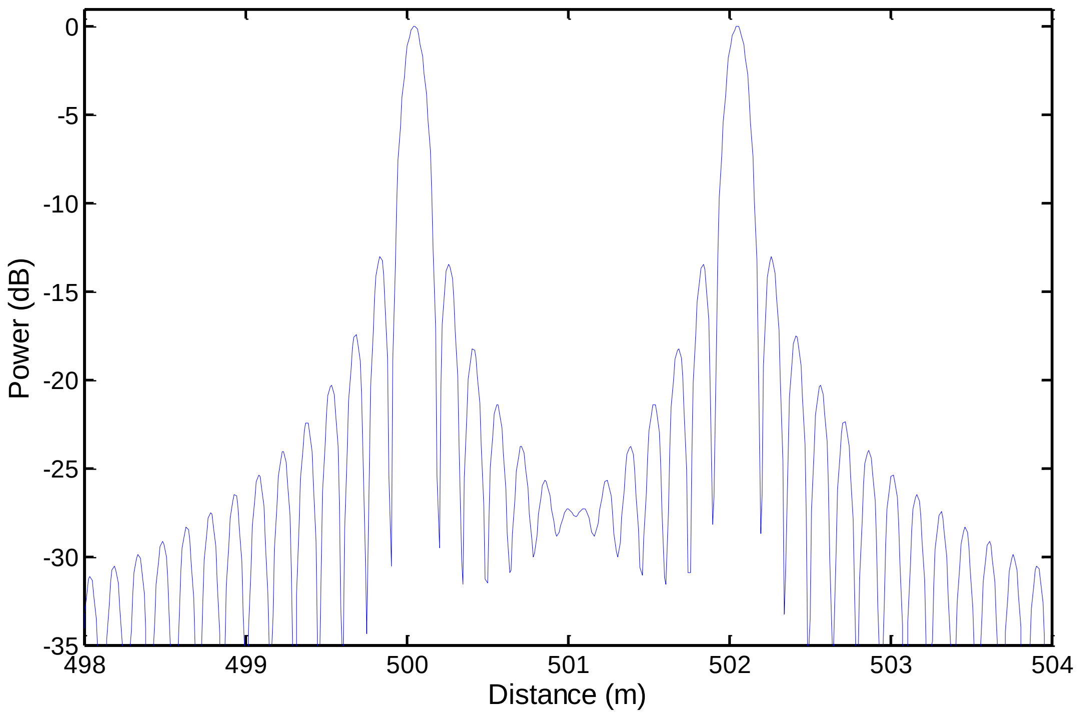


Figure 26. Range response of two close targets. Target 1 and target 2 are located 500 and 502 m away from the radar, respectively. In this example, the center frequency was 10 GHz and the bandwidth was 1 GHz.

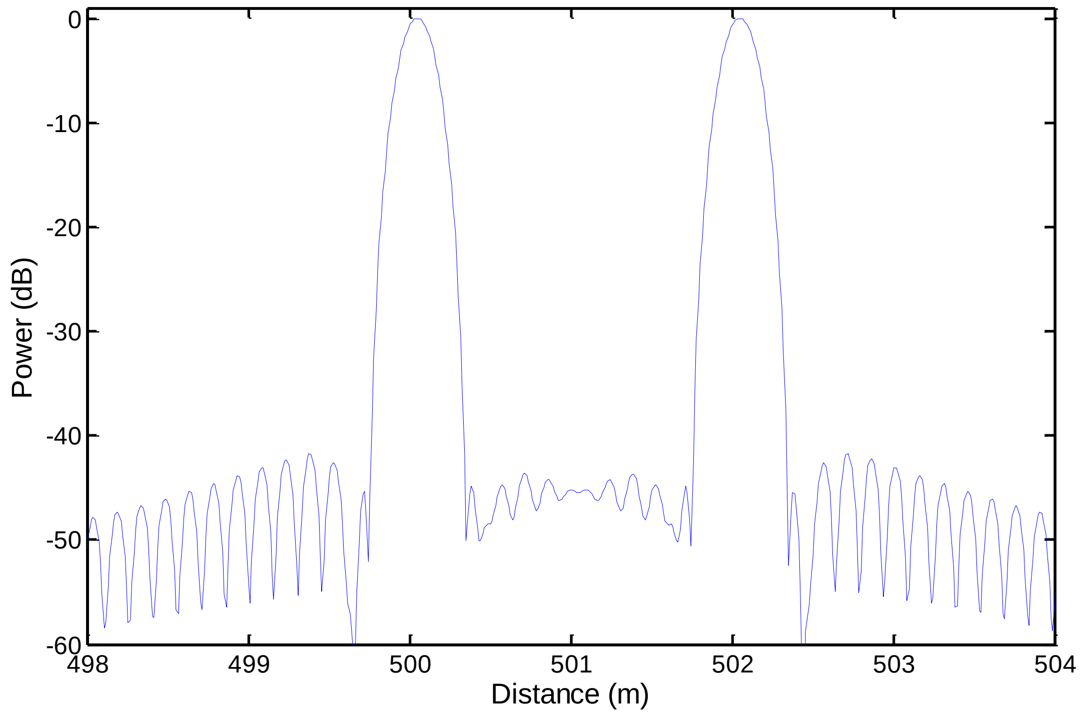


Figure 27. Range response of two close targets, Hamming windowed. Target 1 and target 2 are located 500 and 502 m away from the radar, respectively. The center frequency was 10 GHz and the bandwidth was 1 GHz.

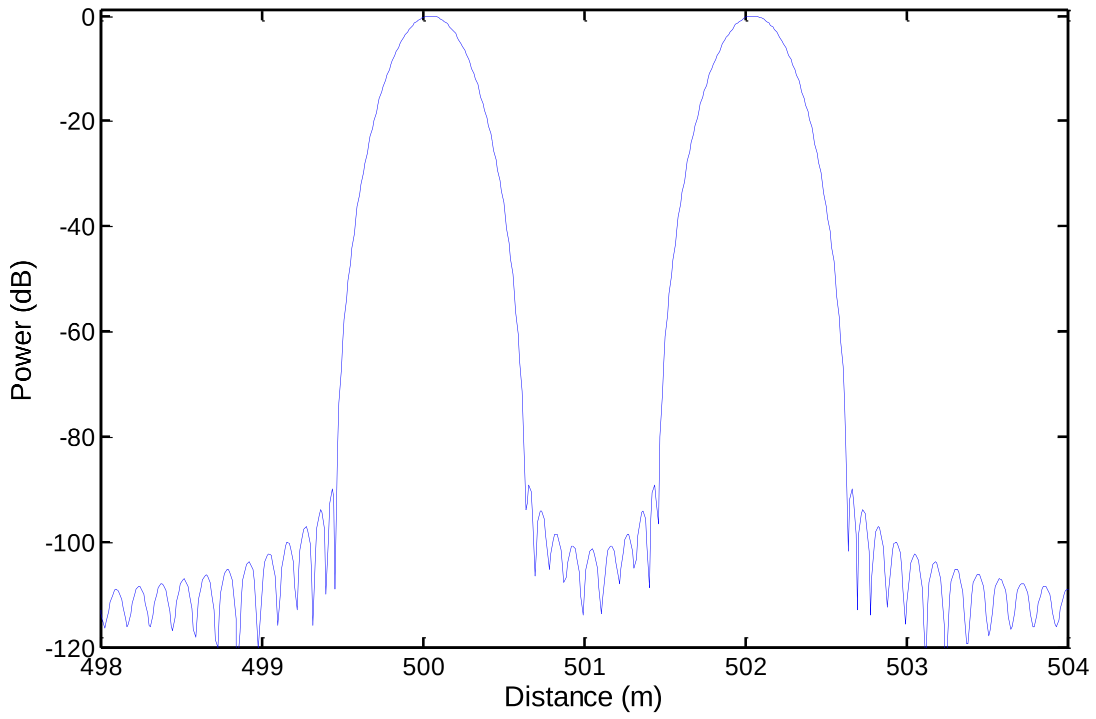


Figure 28. Range response of two close targets, Kaiser windowed ($\alpha = 12$). Target 1 and target 2 are located 500 and 502 m away from the radar, respectively. The center frequency was 10 GHz and the bandwidth was 1 GHz.

We note that the power of the transmitted signal decreases inversely proportional to the square of the one-directional range, which implies that we should have modified (2 - 30) to the form:

$$V[\omega] = \frac{e^{-2jkr}}{r^4} \tag{2 - 37}$$

Although such modification would make our model more physically realistic, we choose to omit this consideration because the effects of the $1/r^4$ term are often filtered out in radar processing. By not modeling the $1/r^4$ term, we eliminate the need to filter in our simulation.

2.2.1.3 Doppler processing.

Doppler processing was modeled using the quasi-static approximation such that a target in motion could be considered as a stationary target at every slow time. This approximation allows a linear time invariant system representation of Doppler processing. To model movement in range, we simply use

$$r[t_s] = d_0 + vt_s + \frac{a{t_s}^2}{2} \tag{2 - 38}$$

where d_0 represents initial distance, v represents the velocity of a target, a represents the acceleration of a target, and t_s represents slow time. In a one-dimensional model, cross range is not measured, so the direction of movement can only be toward (positive velocities) or away (negative velocities).

The radar return concept can be extended from Equation (2 - 30) for a moving target. Here, we fix angular frequency to ω_0, the center angular frequency, and by relation, we fix wave number to k_0. We vary slow time instead of angular frequency.

$$V[t_s] = e^{-2jk_0 r[t_s]} \tag{2 - 39}$$

Notice that this signal is of the form of a frequency shift in the frequency domain. Doppler processing also supports the radar return of multiple targets:

$$V[t_s] = \sum_{\tau}^{\mathrm{N_T}} e^{-2jk_0 r_\tau[t_s]} \tag{2 - 40}$$

In Appendix A.2, we solve for the analytical solution of the Doppler frequency representation of this radar return using a discrete Fourier transform (DFT). In implementation, we used the fast Fourier transform (FFT). We limit $r(t_s)$ to vt_s for simplicity of derivation, and will later show the effect of acceleration and initial distance on the Doppler response. Frequency ω_d is the Doppler angular frequency and T_p is the inter-pulse time $\left(\frac{1}{prf}\right)$:

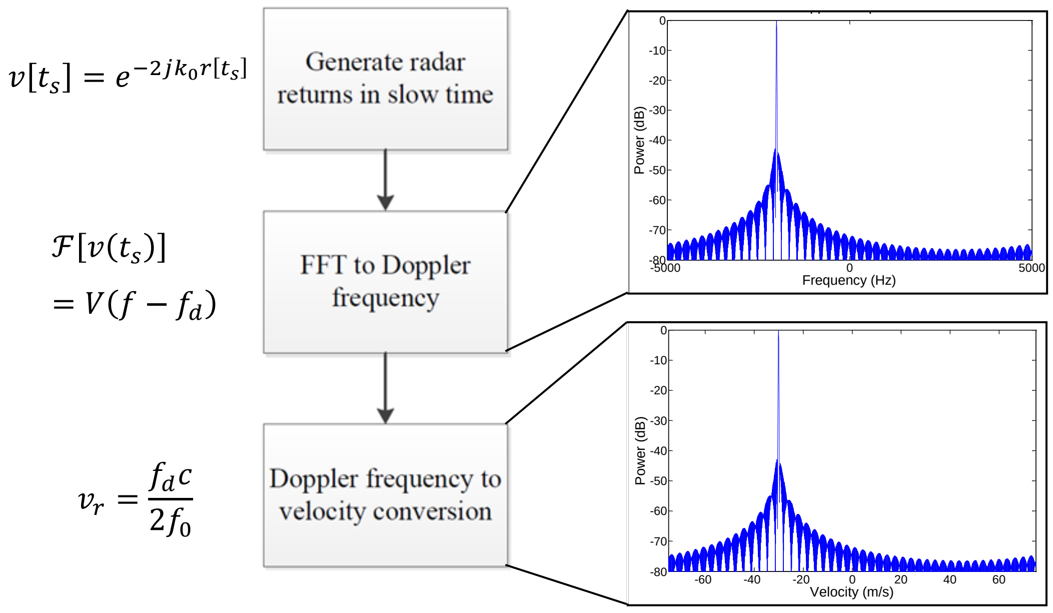
$$V(t_s) \Leftrightarrow \pi T_p e^{-j(2k_0 v+\omega_d)t_0} sinc\left(\frac{(2k_0 v+\omega_d)T_p}{2\pi}\right) = V(\omega_d + 2k_0 v) \qquad (2 - 41)$$

The analytical solution is an amplitude scaled, modulated $sinc$ function in Doppler frequency. Naturally, the Doppler frequency domain ranges from 0 to $F_{s,Doppler}$. The function is circularly shifted by $F_{s,Doppler}/2$ so that the domain effectively ranges from $-F_{s,Doppler}/2$ to $F_{s,Doppler}/2$. This circular shift allows us to model targets moving both toward and away from the radar. The final step is to use the Doppler shift, Equation (2 - 12), to convert Doppler frequency to radial velocity.

Because the form of the Doppler processing $sinc$ function is the same as the form as the range processing $sinc$ function, the discussion on the use of windowing for sidelobe reduction applies to Doppler response sidelobe reduction as well:

$$V[t_s] = W_{t_s}[t_s] \sum_{\tau}^{\mathrm{N_T}} e^{-2jk_0 r_\tau[t_s]} \qquad (2 - 42)$$

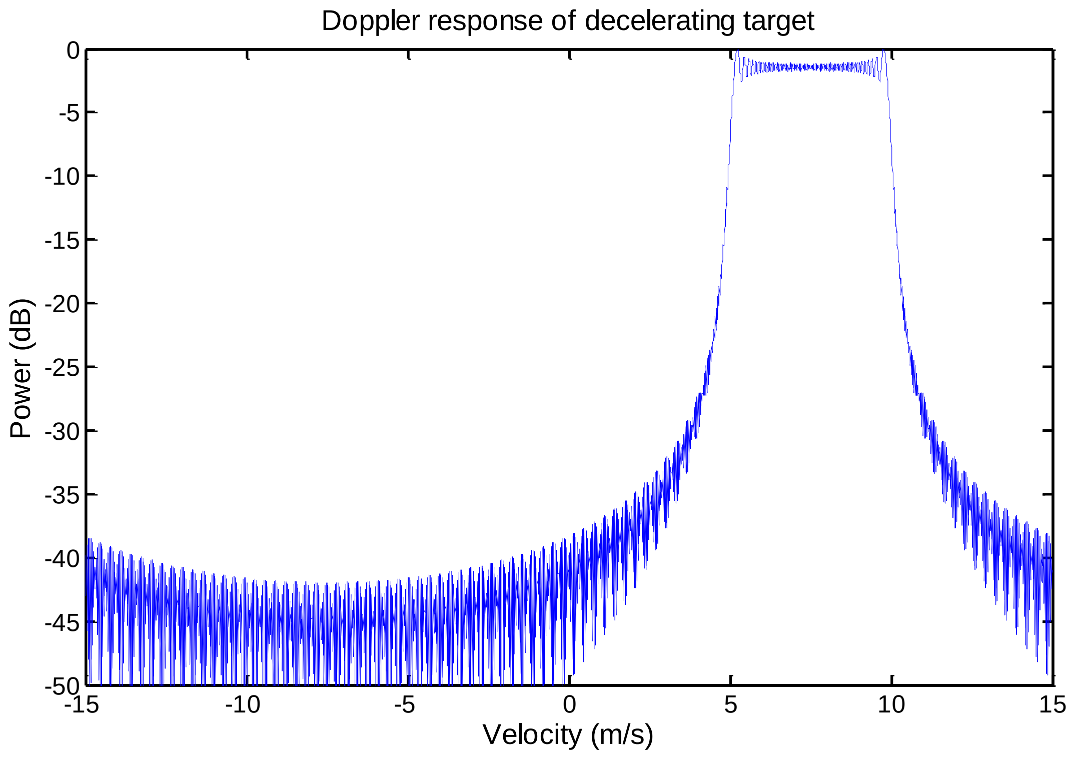
In summary, the processing flow is depicted in Figure 29.



$v[t_s] = e^{-2jk_0 r[t_s]}$
Generate radar returns in slow time
$\mathcal{F}[v(t_s)] = V(f - f_d)$
FFT to Doppler frequency
$v_r = \frac{f_d c}{2f_0}$
Doppler frequency to velocity conversion
Power (dB)
Frequency (Hz)
Velocity (m/s)

Figure 29. Work flow diagram of Doppler processing. (top) Doppler frequency response and (bottom) velocity response of target traveling away from radar at 30 m/s, $f_0 = 10\ GHz$.

When a target is accelerating, the same Doppler processing algorithm is used. As shown in Figure 30, the peak of the Doppler response is wider than the Doppler response peaks for targets with no acceleration. The fundamental reason for this effect is that the target is traveling at a range of velocities during the dwell time. For example, in Figure 30, the target starts at a velocity of $5\ m/s$ and has an acceleration of $5\ m/s^2$. The dwell time is $1\ s$, and in that time range, the velocity ranges from $5\ m/s$ to $10\ m/s$. Notice that it is more difficult to distinguish this range of velocities when the response is windowed.



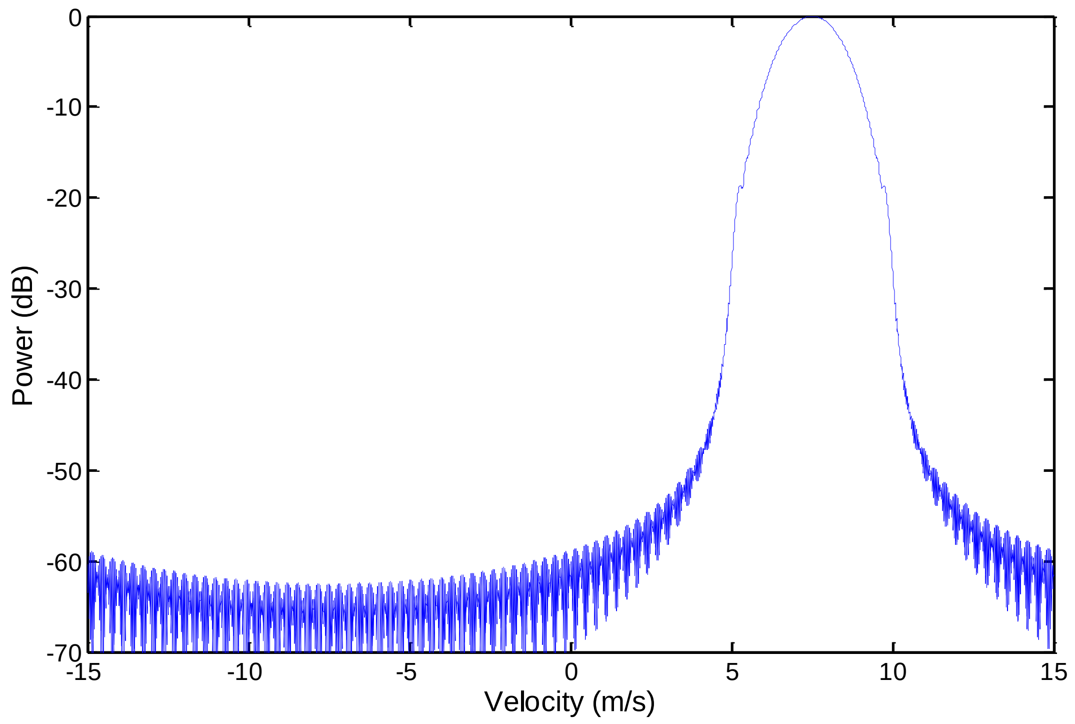


Figure 30. Doppler response of a decelerating target un-windowed (top) and Hamming windowed (bottom). Parameters used to generate this response include: $v_{alias-free} = 15\frac{m}{s}, T_{dwell} = 1\ s, f_0 = 10\ Ghz, d_0 = 500m, v = 5\ m/s, a = \ 5\ m/s^2$

2.2.1.4 Range-Doppler processing.

Range-Doppler processing is a combination of range processing and Doppler processing. The quasi-static approximation, i.e. we assume no change in range during slow time intervals, is maintained for simplicity. Specifically, range-Doppler processing is a combination of Equations (2 - 36) and (2 - 42). That is, radar returns must be generated in both frequency and slow time:

$$V[\omega, t_s] = W_\omega[\omega]W_{t_s}[t_s]e^{-2jk\cdot R_s}\sum_{\tau=1}^{\mathrm{N_T}} e^{-2jkr_\tau[t_s]} \qquad (2 - 43)$$

In MATLAB, Equation (2 - 43) is computed using a three-dimensional matrix: the first and second dimensions vary angular frequency and slow time, respectively, and the third dimension represents the radar returns of different point targets. This 3D matrix space is represented in Figure 31. The third dimension is summed to produce the two-dimensional matrix $V(w, t_s)$.

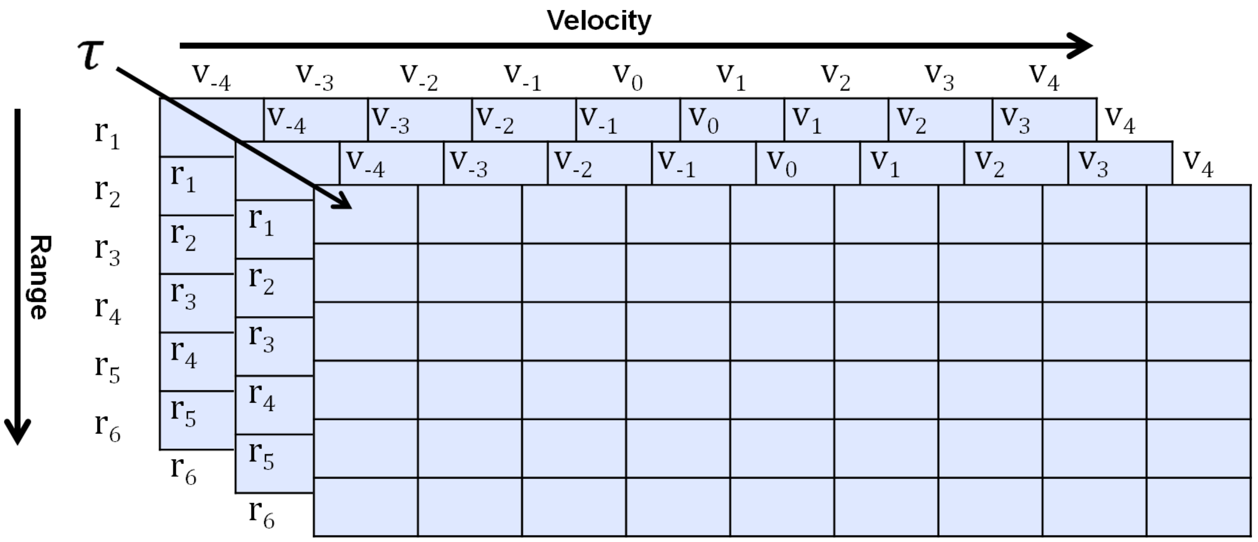

Velocity
τ
Range
v_{-4} v_{-3} v_{-2} v_{-1} v_0 v_1 v_2 v_3 v_4
r_1 r_2 r_3 r_4 r_5 r_6

Figure 31. MATLAB matrix implementation.

The following pages further illustrate the processing. By performing an IFFT over angular frequency, range responses in slow time result, and can be visualized using a Range Time Intensity (RTI) plot, shown in Figure 32. RTI plots are constructed according to Figure 33. On the left, each graph represents a range response for consecutive slow time. These responses are placed side by side, and visualized using a 3D surface plot. The RTI plot is the top-down view of that 3D surface plot. The color represents magnitude in dB related by the colorbar.

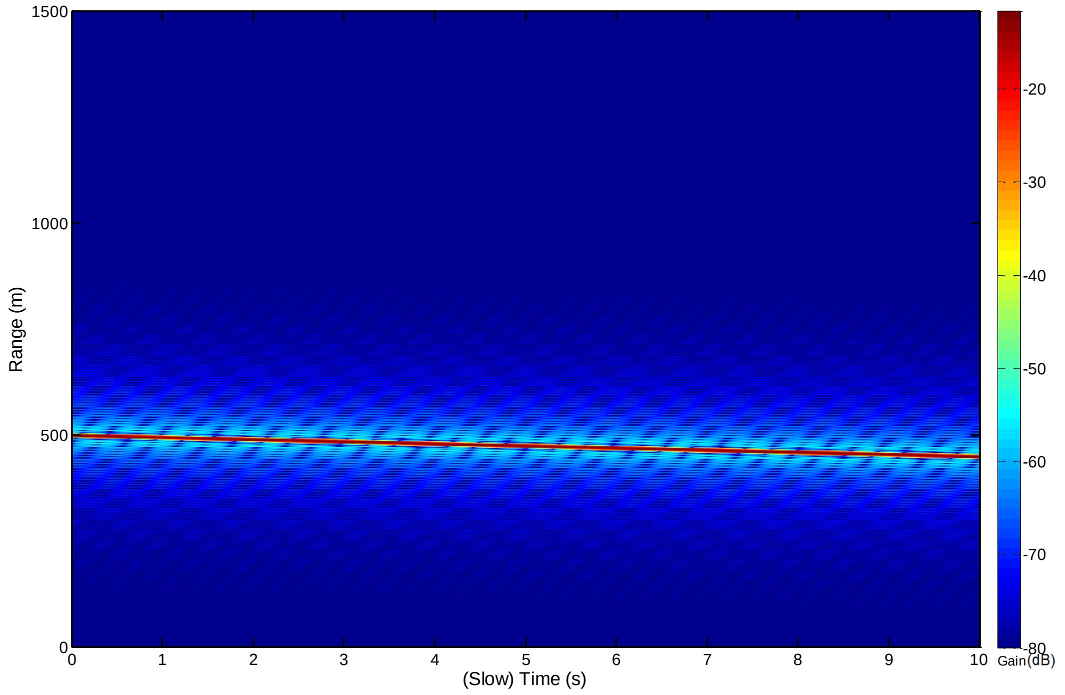


Figure 32. Range time intensity plot for target starting at a range of $500\ m$ with velocity $5\ m/s$ toward the radar and no acceleration.

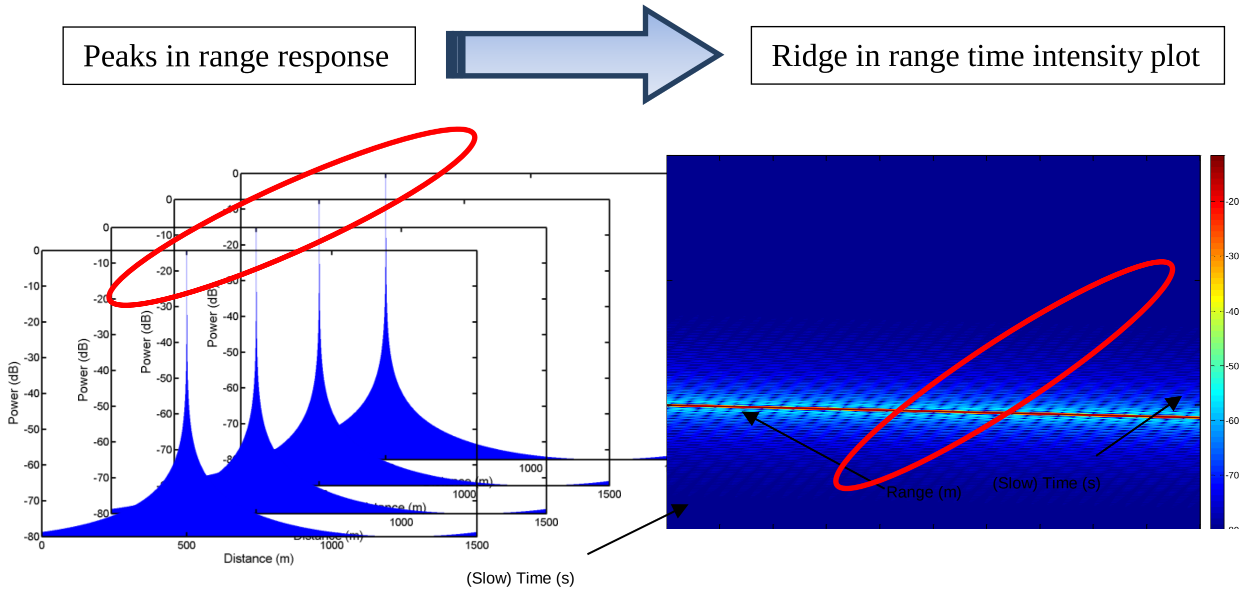


Figure 33. Generation of range time intensity plot.

Next, a DFT over slow time is performed on the RTI, and Doppler frequency is converted to radial velocity, as represented by Equation (2 - 44). The result is a range-Doppler profile (RDP), shown in Figure 34. Now, a point target can be resolved in both range and velocity in this range-Doppler space. Notice that two distinct targets at the same range can be distinguished by their velocities using a RDP.

$$\mathcal{F}_{t_s}[RTI(\omega, t_s)] = V[r, f_d] \rightarrow V\left[r, \frac{f_d c}{2f_0}\right] = RDP[r, v_r] \qquad (2 - 44)$$

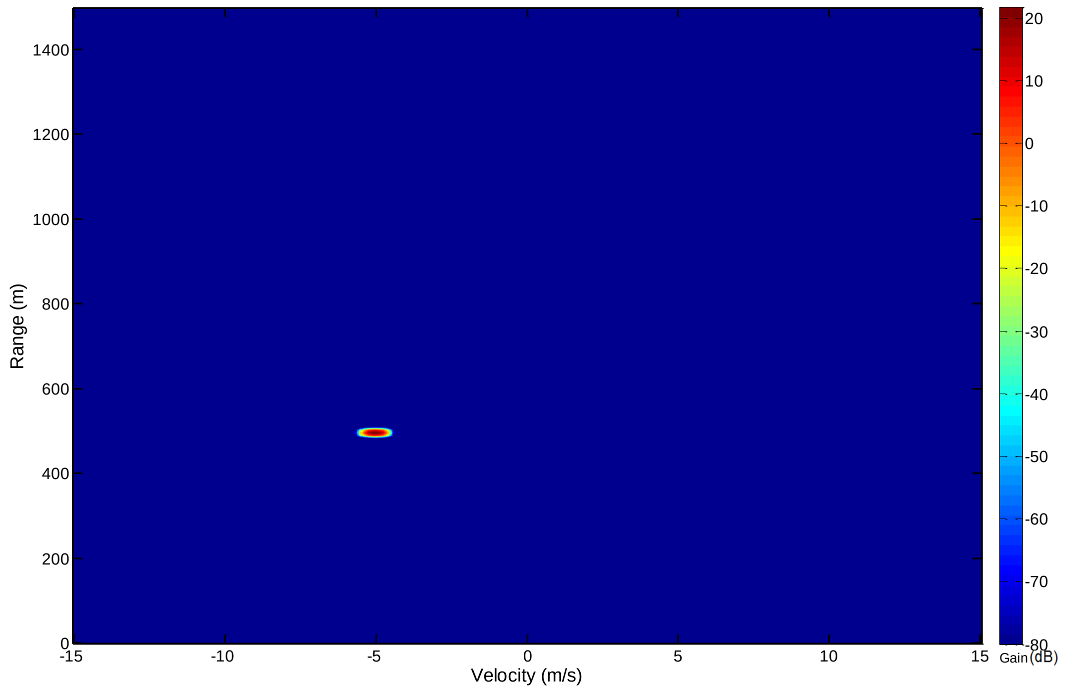


Figure 34. Example of a range-Doppler profile for a target starting $500\ m$ away from radar and travelling away at $5\ m/s$ with no acceleration. $f_0 = 10\ GHz, B = 50\ MHz, r_{alias-free} = 1500\ m, v_{alias-free} = 15\ m/s. T_{dwell} = 0.1\ s,$ Kaiser windowing.

Figure 34 represents a RDP for the time period only between 0s and $0.1s$. Next, through simple iteration techniques, we generated a time-series of RDPs. When played in a movie, it is possible to see the distance vs. time and velocity vs. time kinematics underlying the target. Note that a vectorized computational approach could have been used instead of an iterative approach for each RDP dwell interval. This process was designed with iteration to reduce memory constraints, thus allowing much more stressing radar resolutions. The overall work flow diagram is shown below.

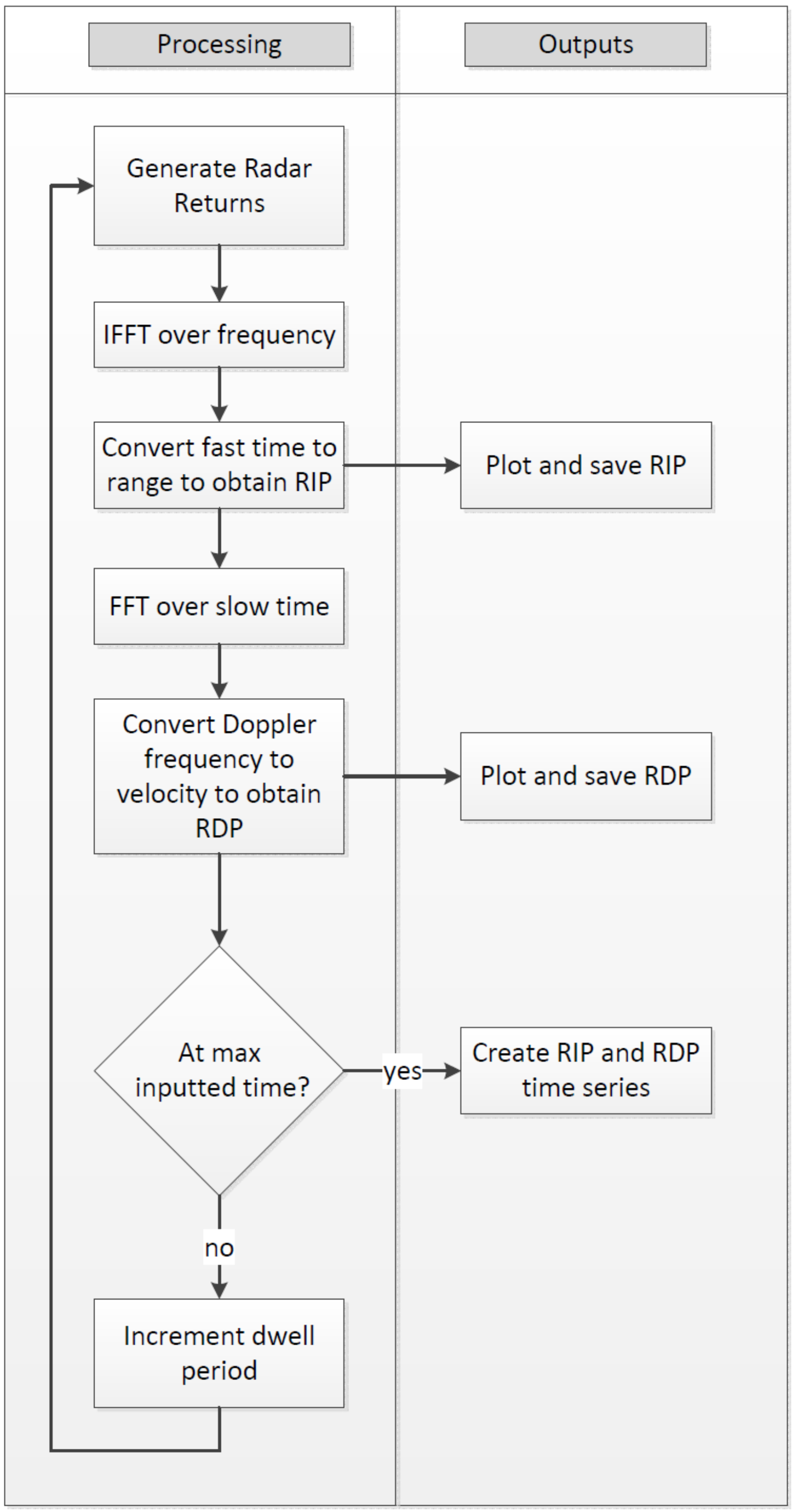


Figure 35. Range-Doppler processing work flow diagram.

2.2.2 One-dimensional ocean model.

Modeling the ocean is a crucial element of our radar scattering simulation for small boat detection. The ocean is particularly important because the power from radar returns of the ocean masks the radar return of small boats. As will be discussed in Section 2.2.3, ocean propagation toward or away from the radar is the most important case to study; thus, it is sufficient to model only a single dimension of the ocean. Before radar returns from the radar can be generated, it is first necessary to get a better understanding of ocean phenomenology and develop a model of the ocean surface.

The goal of creating an ocean model was to develop an algorithm to stochastically create a simulated ocean surface based on a set of input parameters. Thereafter, we used this ocean surface to extract point targets and incorporate with the radar simulation. Because we planned to integrate this model with radar simulations, some radar parameters were necessary so that the ocean was generated for compatible time indices. Besides these parameters, the ocean model is a stand-alone function independent of radar processing.

There are nine inputs into our algorithm comprised of the:

- ocean wind speed(s),
- starting distance of the ocean,
- ocean length,
- ocean sampling resolution,
- ocean propagation direction,
- simulation duration,
- alias-free maximum velocity of radar system,
- center frequency of the radar, and
- toggle that determines if the ocean should be plotted.

The ocean wind speeds are used to produce the spectral composition of the ocean. The starting distance, length of the ocean, and the ocean sampling resolution were used to create a discretized distance vector. Time was discretized based on the center frequency, alias-free velocity, and the simulation duration. The propagation direction of the ocean specifies whether the waves propagate up-sea or down-sea i.e. toward or away from the radar. The user also has the option of plotting the results using a toggle variable in order to visually inspect the ocean surface.

The first step toward the development of an ocean surface model was to use the spectral composition of a fully developed sea to generate waves. More specifically, we modeled the ocean using a sum of randomly phased sinusoids with scaled amplitudes to match the Pierson-Moskowitz spectrum. Before we explain the ramifications of this summation, we explain the function used to create a single frequency component of the ocean. The height of an ocean surface for a single ocean frequency as a function of time and distance is given by:

$$\boldsymbol{\eta}[x, t_s] = A\cos(\omega t_s - k_d x + \boldsymbol{\phi}) \tag{2 - 45}$$

where x is the horizontal distance to a point on the ocean surface, t_s is the time instance of the ocean surface, ω is a single angular frequency, A is the amplitude or height of the wave, k_d is dispersive wave number, and $\boldsymbol{\phi}$ is a phase shift. Note that it is only necessary to generate the ocean surface for instances of slow time. In this report, we use boldface to represent random variables and processes. According to Kinsman (2002), the energy of a wave is proportional to the amplitude squared (p. 153). Thus, the amplitude A is proportional to the square root of the power of the Pierson-Moskowitz spectrum at the single frequency ω, as seen in Equation (2 - 46). We use this proportionality as equality, and later, we will describe how we correct this approximation through scaling.

$$A \propto \sqrt{G(\omega)} \tag{2 - 46}$$

The wave number k_d is given by the dispersion relation:

$$k_d = \frac{\omega^2}{g} \tag{2 - 47}$$

The phase shift $\boldsymbol{\phi}$ is a uniform random variable between 0 and 2π as shown in Equation (2 - 48). This random variable will introduce a random starting phase of the wave and is necessary when adding multiple ocean frequencies together to add variations between different generations of the ocean surface.

$$\boldsymbol{\phi} = U(0, 2\pi) \tag{2 - 48}$$

Equation (2 - 45) is analogous to an amplitude scaled, time shifted plane wave. In terms of the ocean, the equation shows that when t is constant and d is variable, η represents a single frequency ocean surface at a fixed time across all distance. On the other hand, when d is constant and t is variable, η represents the ocean surface height as a function of time at a single location. The last example would be similar to the movement of a buoy in time.

Now that we have explained the equation of ocean surface height for a single ocean frequency, we explain how we created an ocean with many spectral components. The ocean surface as a function of time and distance and spectral components from the Pierson-Moskowitz spectrum is described by the linear sum of sinusoids equation below.

$$\boldsymbol{\eta}[x, t_s] = \sum_{i}^{N} A_i \cos(\omega_i t_s - k_i x + \boldsymbol{\phi_i}) \tag{2 - 49}$$

where A_i, ω_i, k_i, and $\boldsymbol{\phi_i}$ are now indexed for each frequency and change for each i. N equally spaced frequency components are used. These indexed terms represent the same values and are described the same as they are in Equations (2 - 46) through (2 - 48). There are two primary limits that we must address. The first is finding a maximum frequency limit and the second is finding the necessary number of equally spaced frequencies N to model.

We chose to limit the maximum frequency sampled in the Pierson-Moskowitz spectrum to ninety-nine percent of the total power in the spectrum. This limit was found analytically. By taking the integral of the PM spectrum equation, we derived the power in the angular frequency band between zero and the angular frequency ω, as shown in Equation (2 - 50). Note that $G(\omega)$ refers to the PM spectrum from Equation (2 - 20) and ω_m refers to maximum frequency based on wind speed shown in Equation (2 - 21).

$$\int_0^{\omega} G(\omega)\ d\omega = \frac{1.62x10^{-3} g^2}{{\omega_m}^4} e^{-1.25\left(\frac{\omega_m}{\omega}\right)^4} \tag{2 - 50}$$

Next, we found the total power in the ocean by setting ω equal to ∞; the results are shown in Equation (2 - 51).

$$\int_0^{\infty} G(\omega)\ d\omega = \frac{1.62x10^{-3} g^2}{{\omega_m}^4} \tag{2 - 51}$$

A useful method for limiting the spectral composition of the ocean is to find the frequency which results in ninety-nine percent of the total power of the spectrum when used as an upper bound in Equation (2 - 50). To find this frequency, denoted $\omega_{99\%}$, we used Equation (2 - 52).

$$0.99 = \frac{\int_0^{\omega_{99\%}} G(\omega)\ d\omega}{\int_0^{\infty} G(\omega)\ d\omega} = e^{-1.25\left(\frac{\omega_m}{\omega_{99\%}}\right)^4} \tag{2 - 52}$$

Substituting in Equations (2 - 50) and (2 - 51) into Equation (2 - 52) and solving for $\omega_{99\%}$ results in Equation (2 - 53). This equation represents the maximum frequency that we needed to model to obtain ninety-nine percent of the power as a function of wind speed in m/s.

$$\omega_{99\%} = \omega_m \left(\frac{-1.25}{ln(0.99)}\right)^{\frac{1}{4}} = \frac{2.772\,g}{V_{wind}} \tag{2 - 53}$$

The plot of this function, as shown in Figure 36, indicates that the maximum frequency containing ninety-nine percent of the power decreases inversely proportionally to the wind speed. Also notice that the frequency drops significantly until approximately 5 knots, where it begins dropping at a much slower rate. Later, we use this fact to simplify processing.

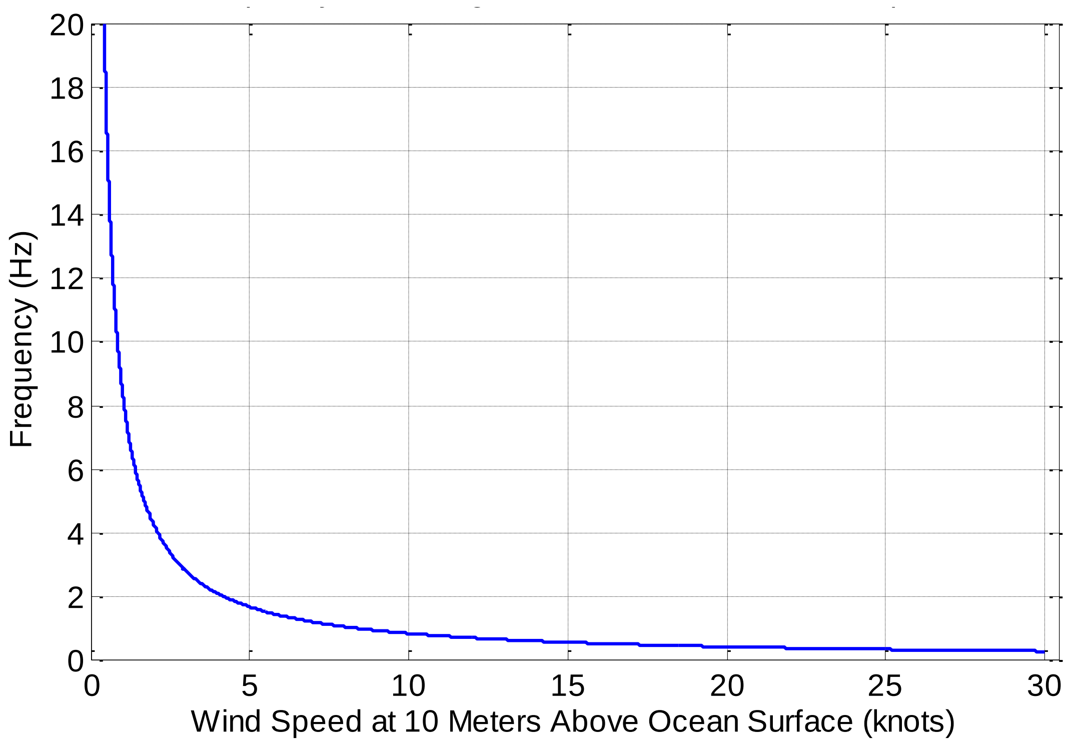


Figure 36. Frequency containing ninety-nine percent power of Pierson-Moskowitz spectrum versus wind speed.

The relationship between the frequency containing ninety-nine percent power and wind speed can be intuitively verified by inspecting a plot of the PM spectrum in power vs. wind speed, as shown in Figure 37. The analysis of this figure leads to the fact that there is more power located at lower frequencies for higher wind speeds; therefore, the frequency containing ninety-nine percent of the power for higher wind speeds is relatively lower than the frequency containing ninety-nine percent power for lower wind speeds, as Figure 36 suggests.

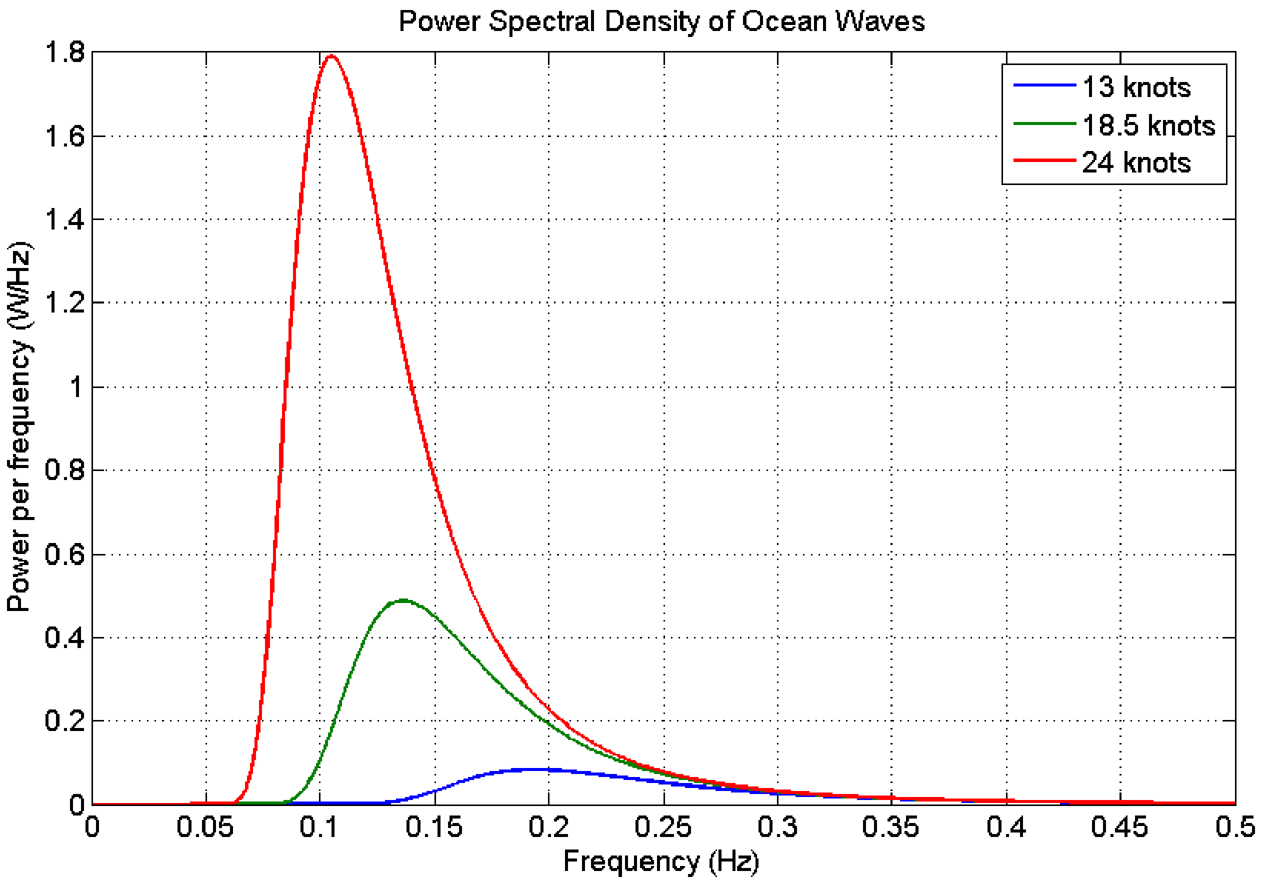


Figure 37. Pierson-Moskowitz power spectral density of ocean in watts. Plotted for three wind speeds of 13, 18.5, and 24 knots.

Using the frequency limit, we completed our ocean surface definition by modifying Equation (2 - 49) to:

$$\boldsymbol{\eta}[x, t_s] = \sum_{i=1}^{N_{\omega_{99\%}}} A_i \cos(\omega_i t_s - k_i x + \boldsymbol{\phi_i}) \tag{2 - 54}$$

where $N_{\omega_{99\%}}$ is the index of the limiting frequency.

After we determined the frequency limit for our ocean surface definition, we realized that that there was a distance at which the ocean repeated within the generated ocean distance if not enough PM sample points were chosen. This ocean repetition looked extremely unnatural. To solve this problem, an arbitrarily large number could be chosen for the number of sampling points; however, this resulted in long computation times. To determine the number of frequencies necessary to produce an ocean model that does not contain repetition, we used numerical measurements. Although we know that we can solve for the period of a summation of sinusoids using the least common multiple method, the method relies on known frequencies which have been sampled in our case. We started by generating an ocean with varied numbers of sampling frequencies, visually measured the repetition distances, and recorded the measurements. Next, we repeated this process for several ocean wind speeds, and created a scatter plot of these data. We noticed that there was a square root relationship between the ocean repetition distance, denoted as $r_{repetiton}$, and number of sampling points used. However, the relationship was not exactly square root; in fact, we noticed that it was of the form:

$$N_{required}\left[r_{repetition}\right] = a + b\sqrt{r_{repetiton}} \qquad (2 - 55)$$

We used a second order polynomial fitted function to determine the values for the fit parameters a and b for a wind speed of twenty-four knots. The empirical measurements as well as the polynomial fitted equations are shown in Figure 38. Next, we expanded the above equation by adding dependence of wind speed, which was determined through pattern matching methods (V_{wind} is specified in knots here). In addition, we added a safety factor of 1.2.

$$N_{required}\left[r_{repetition}, V_{wind}\right] = \left\lceil 1.2 \cdot \frac{24}{V_{wind}}\left(a + b\sqrt{r_{repetiton}}\right)\right\rceil \qquad (2 - 56)$$

We use Equation (2 - 56) to select the minimum number of sampling points required based on the ocean length input, which ensured that the number of sampling points chosen was neither too large nor too small.

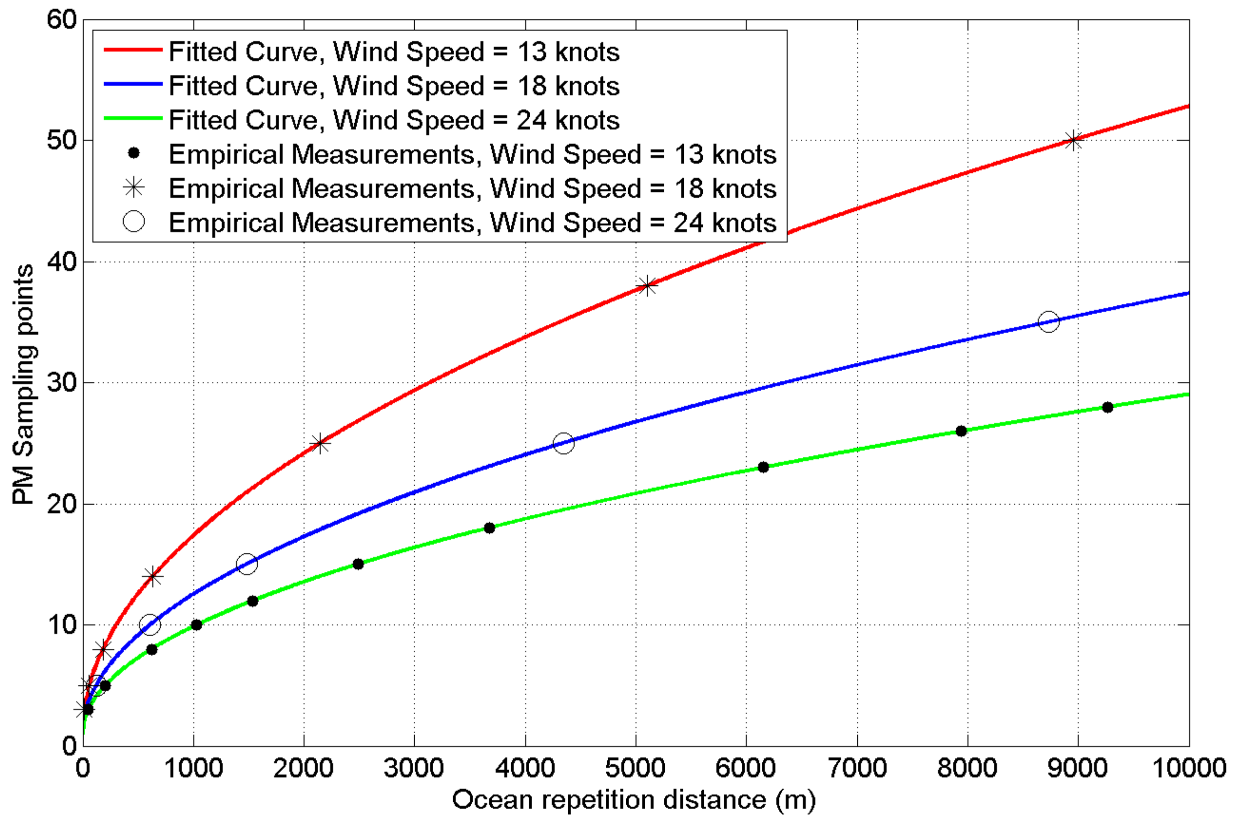


Figure 38. Pierson-Moskowitz sampling points as a function of ocean repetition distance.

After we could describe the ocean surface in terms of a sampled limit of PM spectral components using Equation (2 - 54), it was necessary to scale the ocean height. As expressed earlier, scaling is essential because Equation (2 - 46) is an approximation. Recall that the Beaufort scale correlates wind speed ranges to significant wave height ranges. Using the midpoints of each wind speed range and wave height range, we developed a fourth order polynomial fitted equation that describes the dependence of wind speed on significant wave height. The plot of the midpoints and polynomial fitted equation can be seen in Figure 39 and the equation is related via:

$$H_{S(Beaufort)} = 7.41x10^{-4}\, V_{wind}{}^{4} - 8.40x10^{-5}\, V_{wind}{}^{3} + 5.20x10^{-3}\, V_{wind}{}^{2} + 6.14x10^{-2}\, V_{wind} - 3.12x10^{-2} \qquad (2 - 57)$$

where V_{wind} is the wind speed above the ocean.

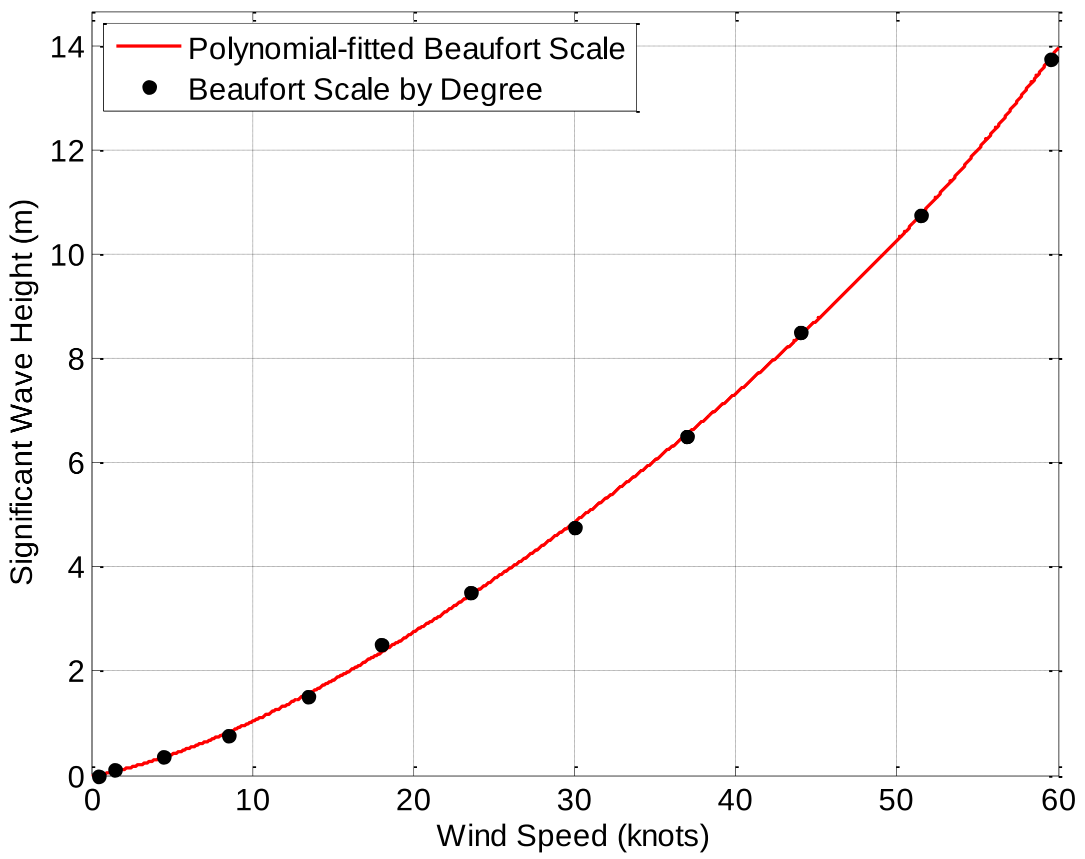


Figure 39. Beaufort Scale. Black points represent the midpoint of each degree of the Beaufort scale. The red line indicates a fourth order polynomial fitted line that describes the significant wave height by wind speed.

In order to scale our ocean model to the Beaufort scale, it was necessary to determine the significant wave height generated from our ocean model. Calculating significant wave height was possible using Equation (2 - 22). According to the National Data Buoy Center (2011), the significant wave height is measured by buoys on the ocean surface, which implies that the standard deviation should be calculated over time (National Oceanic and Atmospheric Administration). The ergodic theorem permits the standard deviation to be also calculated over distance (Papoulis, 2002, p. 276). To obtain accurate results, we chose to perform a two-dimensional standard deviation operation across distance and time. This process is described by Equation (2 - 58).

$$\boldsymbol{\sigma_{\eta}} = \sigma_{2D}(\boldsymbol{\eta}[d, t_s]) \qquad (2 - 58)$$

Using Equation (2 - 22) and (2 - 58) we can find the significant wave height of our ocean model as shown in Equation (2 - 59).

$$\boldsymbol{H}_{S(ocean\,model)} \approx 4\boldsymbol{\sigma}_{\boldsymbol{\eta}} \tag{2 - 59}$$

Finally, using the empirical significant wave height $H_{S(Beaufort)}$ and the computed ocean model significant wave height $H_{S(ocean\,model)}$, we found the ratio to scale the modeled ocean to the ocean data. The equations used to find the ratio and scale the modeled ocean are described in Equations (2 - 60) and (2 - 61).

$$H_{S(scale)} = \frac{H_{S(Beaufort)}}{\boldsymbol{H}_{S(ocean\,model)}} \tag{2 - 60}$$

$$\boldsymbol{\eta}_{\boldsymbol{scaled}}[x, t_s] = H_{S(scale)} \cdot \boldsymbol{\eta}[x, t_s] \tag{2 - 61}$$

Figure 40 shows the result of scaling the surface of a modeled ocean to fit the significant wave height expected from a given wind speed. Figure 40a is the modeled ocean before scaling and Figure 40b is the ocean after scaling. Note the difference in y-axis scales.

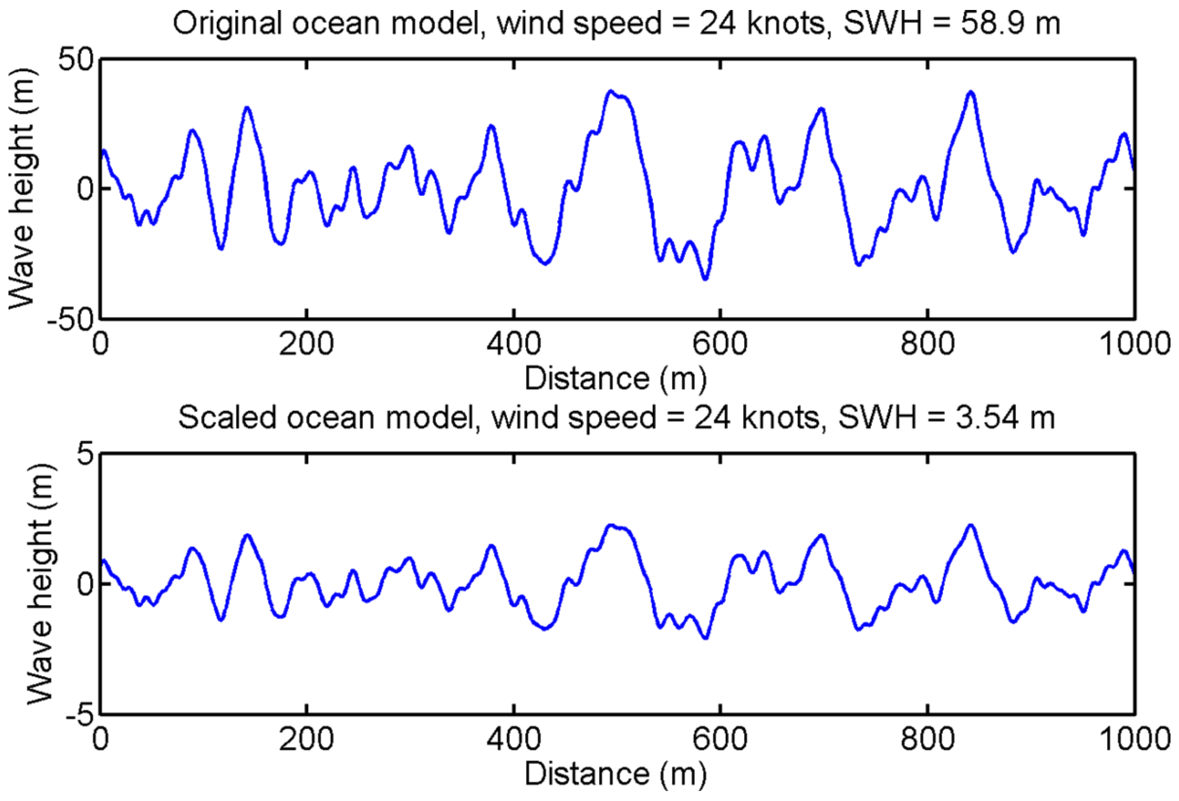


Figure 40. Scaling wave height comparison. Figure 40a shows an ocean before scaling for a wind speed of 24 knots with a significant wave height of 58.9 m. Figure 40b shows the ocean scaled for 24 knots with a significant wave height of 3.54 m. Note the difference in y-axis scales.

One method of code verification for our ocean model was to take a two-dimensional DFT of $\boldsymbol{\eta}$. This process converts the time axis to angular frequency and the distance axis to wave number, which is a spatial frequency. A plot of the two-dimensional DFT as well as the expected dispersion relation is shown in Figure 41. The expected dispersion relation, from Equation (2 - 17), as a function of wavenumber is given by:

$$\omega = \sqrt{k_d g} \tag{2 - 62}$$

Figure 41 shows a set of high amplitude vertical regions that represent each of the frequencies of the sampled PM spectrum. We will refer to these regions as bars.

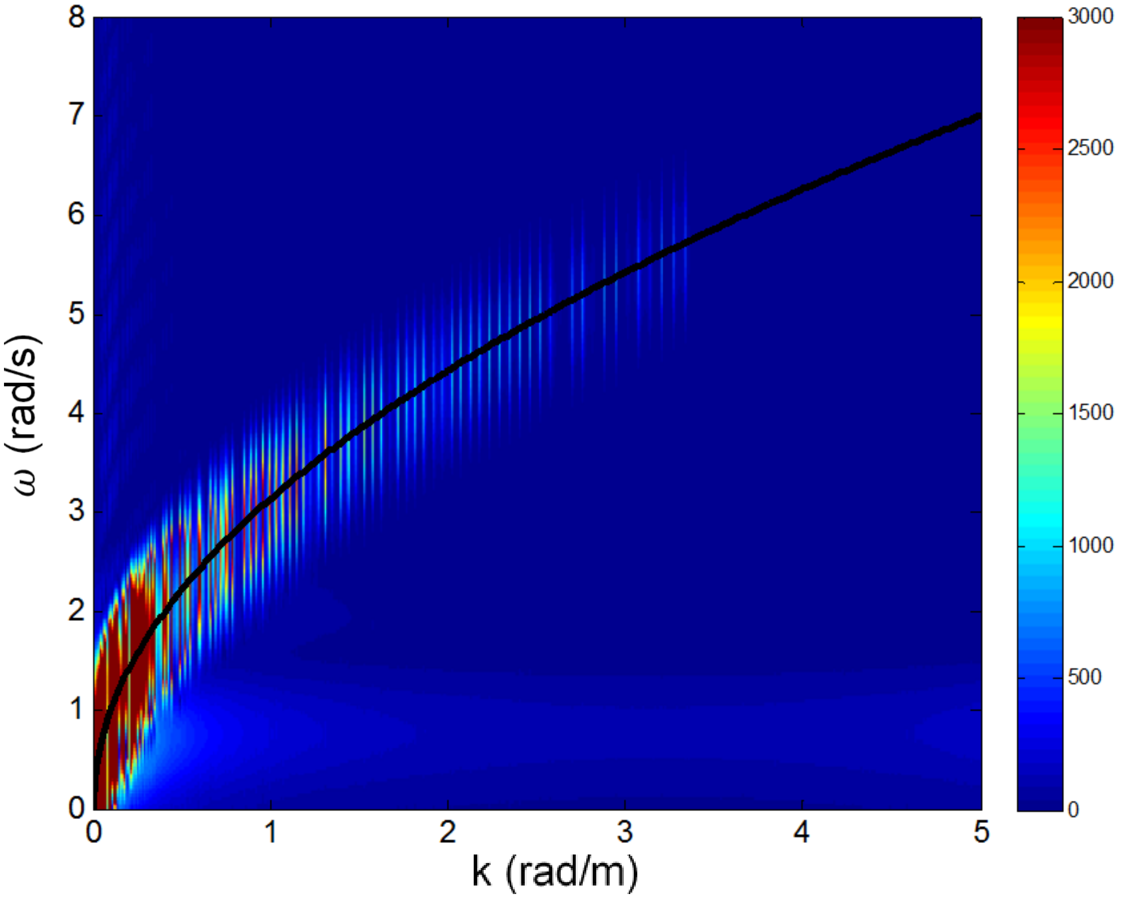


Figure 41. Two-dimensional FFT of the ocean surface in terms of the spatial and radial frequency. The bars represent frequencies following the dispersion relation. The overlaid black line represents the expected dispersion relationship.

Figure 41 verifies four aspects of our ocean model. First, the bars follow the dispersion relation represented by the black line, which proves that the dispersion relation was applied correctly. Second, bars are limited at a frequency that corresponds to $\omega_{99\%}$ for that ocean surface and proves that the cutoff frequency calculation is correct. Third, the intensity of the bars show that the power falls off as the frequency increases which proves that the model follows the PM spectrum equation. Finally, the number of bars represent the number of frequencies ($N_{\omega_{99\%}}$) produced in our model and can be verified by counting. After analyzing these aspects using multiple test cases, we were able to verify that all four parameters followed the model we had chosen.

Now that we could describe the ocean surface in terms of a scaled, sampled limit of PM spectral components using Equation (2 - 54), it was possible to create point targets using the ocean surface, which are later used to integrate the ocean surface and radar. Figure 42 shows a visual representation of an ocean model over a distance of one kilometer with a wind speed of 24 knots and sampled every 1 m.

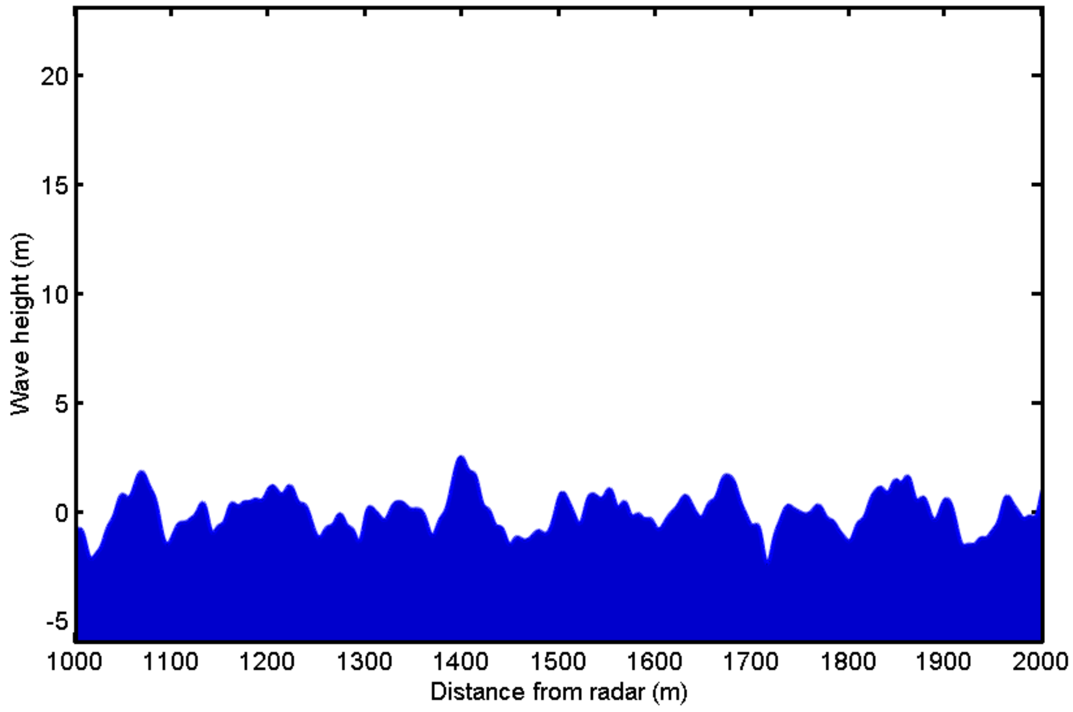


Figure 42. Ocean surface model. This ocean covers 1000 meters with a wind speed of 24 knots.

Figure 43 shows a visual representation of the same ocean model over a distance of forty meters with a wind speed of 24 knots. Notice that this figure has a one to one axis ratio and is more physically realistic.

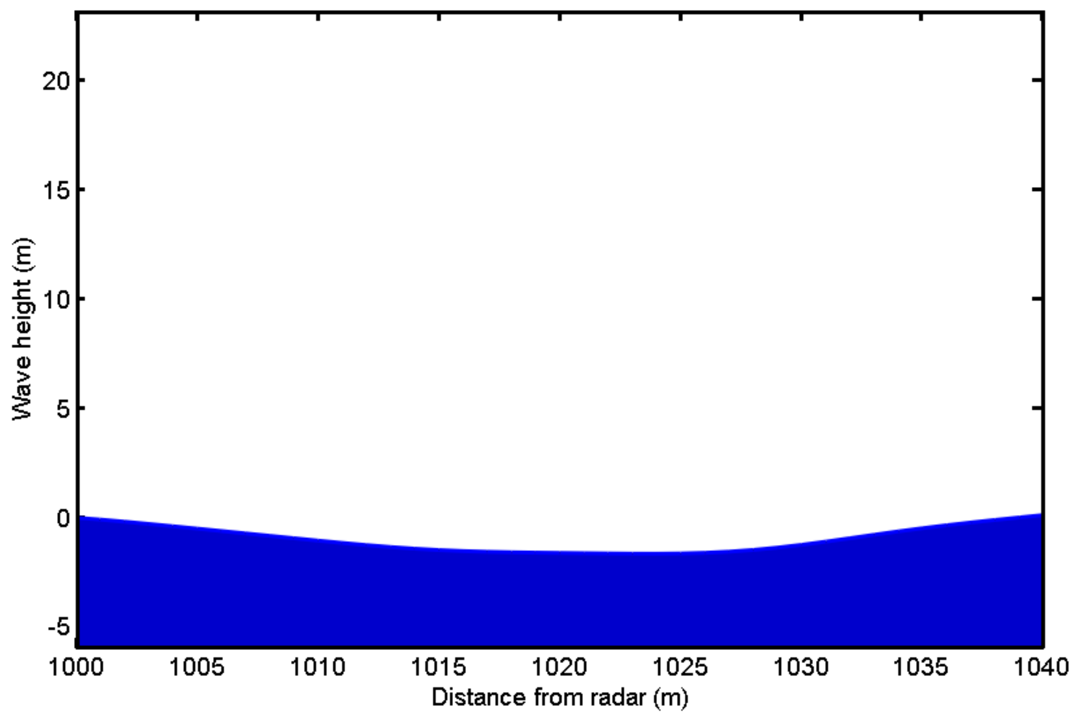


Figure 43. Ocean surface model. This ocean covers 40 meters with a wind speed of 24 knots. This is a one to one aspect ratio.

2.2.3 Integration and calibration.

The non-scanning radar described in Section 2.2.1 can be integrated with the one-dimensional ocean model described in Section 2.2.2 to produce our first deliverable, the non-scanning, one-dimensional ocean radar scattering simulation. An in-depth discussion of integration and calibration methods follow.

We present the overall work flow for the non-scanning one-dimensional ocean scattering simulation in Figure 44. This flow is presented first to help delineate the complexities of the simulation. The general method was to compute the RTI for both the ocean and the targets independently, and then to sum them and find the RDP of the total. One particularly important aspect to notice is that processing was conducted independently for the ocean and the targets and are later summed in time. Moreover, the radar returns of the entire ocean duration were generated and calibrated at once, but the radar returns of the targets were generated for dwell intervals.

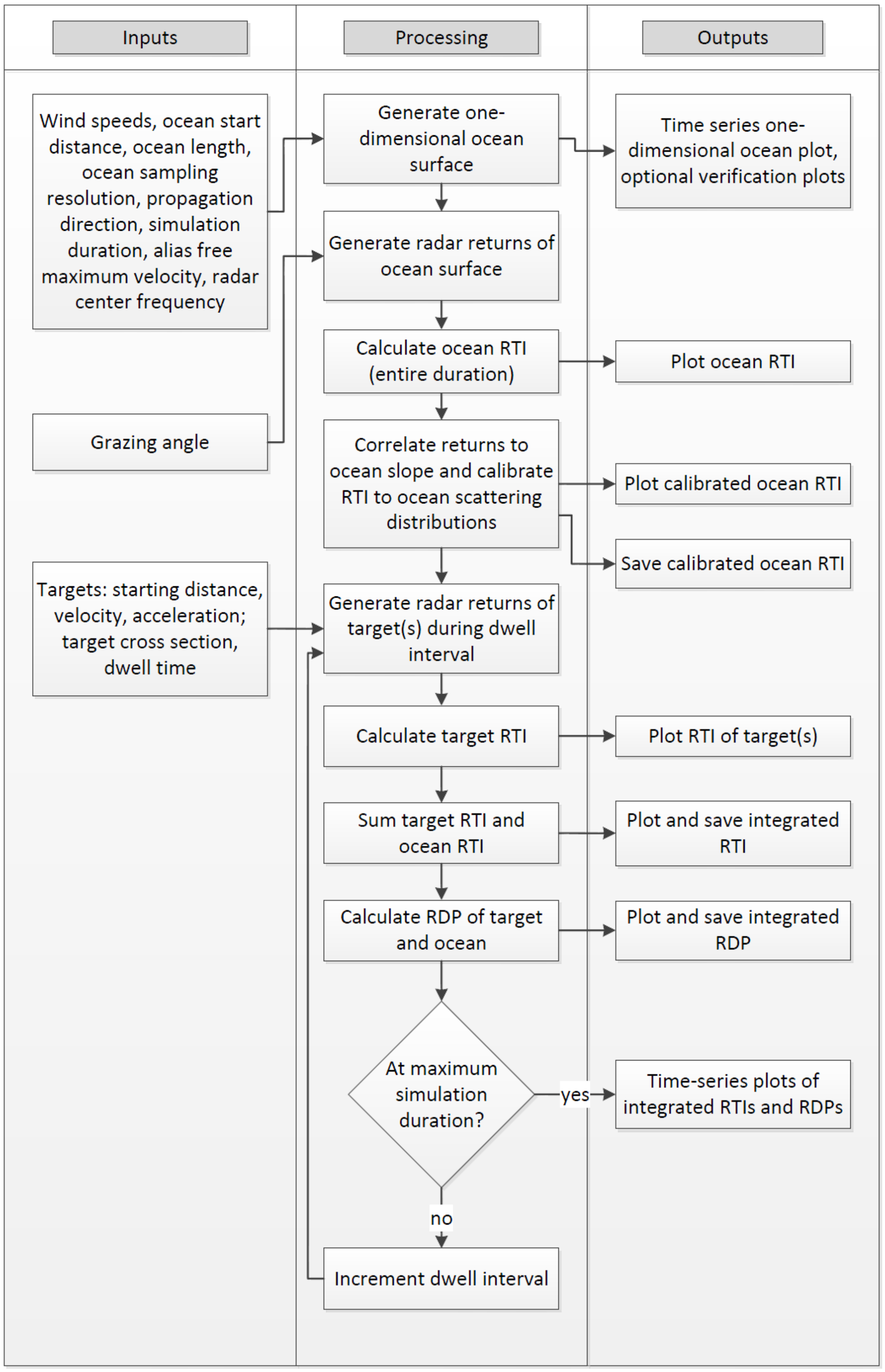

Inputs
Processing
Outputs
Wind speeds, ocean start distance, ocean length, ocean sampling resolution, propagation direction, simulation duration, alias free maximum velocity, radar center frequency
Grazing angle
Targets: starting distance, velocity, acceleration; target cross section, dwell time
Generate one-dimensional ocean surface
Generate radar returns of ocean surface
Calculate ocean RTI (entire duration)
Correlate returns to ocean slope and calibrate RTI to ocean scattering distributions
Generate radar returns of target(s) during dwell interval
Calculate target RTI
Sum target RTI and ocean RTI
Calculate RDP of target and ocean
At maximum simulation duration?
yes
no
Increment dwell interval
Time series one-dimensional ocean plot, optional verification plots
Plot ocean RTI
Plot calibrated ocean RTI
Save calibrated ocean RTI
Plot RTI of target(s)
Plot and save integrated RTI
Plot and save integrated RDP
Time-series plots of integrated RTIs and RDPs

Figure 44. Overall work flow diagram for the non-scanning one-dimensional ocean model. The left pane shows user inputs, the middle pane shows algorithmic flow, and the right pane shows simulated outputs.

Two requirements led us to design our computation architecture around the use of independent target and ocean returns. The first requirement was to create robust, easy to use code. Independently computed returns for the targets and ocean allow the architecture to be more compartmentalized and flexible at a minor speed cost. In addition, these characteristics were very important in our computational architecture design because 1) compartmentalized architectures are relatively less complex and more capable than un-compartmentalized architectures, and 2) algorithms can be more generic and modifiable. While these advantages do not benefit this first deliverable, the advantages will become very apparent in the transition to the second deliverable. The small loss in computational speed is not worth the added benefits of flexibility. The second requirement that led our architecture design was our sponsor's request that a calibrated ocean may be saved. A saved calibrated ocean is beneficial because it is often more useful to vary target parameters than ocean parameters. It is not worth the time to re-generate and re-interrogate an ocean every time target parameters need to be varied because of the relatively long time it takes to construct the ocean data. In addition, another added benefit to independent RTI computation is that it is often useful to see the radar response of just one of the two scattering sources.

The first step in this algorithm was to generate a one-dimensional ocean surface, which was completed using the results of Section 2.2.2. Large models are memory bound during computation, as will be discussed in Section 5.2. If memory constraints allow, multiple oceans can be generated at the same time. In addition, every algorithmic process presented below supports computation given multiple oceans. Simultaneous ocean calculation is advantageous because the user can be sure that the same ocean generation parameters are being used and the processing will take less time. We observe a processing performance increase in this case because the computations are vectorized across different oceans. In fact, the only computations that are not vectorized in this project are dwell intervals (to allow for larger models) and ocean calibration (because the functions used do not support vectorization).

The next step was to generate the radar scattering of the ocean surface, which was accomplished by considering every point in the ocean surface, $\boldsymbol{\eta}[x, t_s]$, as a point target. According to our discussion of radar processing and the quasi-static approximation in Section

2.2.1, we needed to determine only the range from each ocean point target to the radar as a function of slow time (for both range and Doppler processing, see Section 2.2.1.4). We distinguish the horizontal distance, x, from the range, r, in Figure 45. As this figure suggest, the radar is raised and is looking down at the ocean at a specified grazing angle. The static height, h, of the radar was computed via

$$h = \frac{oceanStart + oceanLength}{2} \tan(grazingAngle) \qquad (2 - 63)$$

This ensured an exact grazing angle at the middle of the ocean. We used this geometry to compute the range to each point target in slow time given the height of the radar, and the ocean surface:

$$r_\tau[t_s] = \sqrt{(h - \boldsymbol{\eta}[x, t_s])^2 + x^2} \qquad (2 - 64)$$

Note that radar returns were not calculated directly with Equation (2 - 43) at this point.

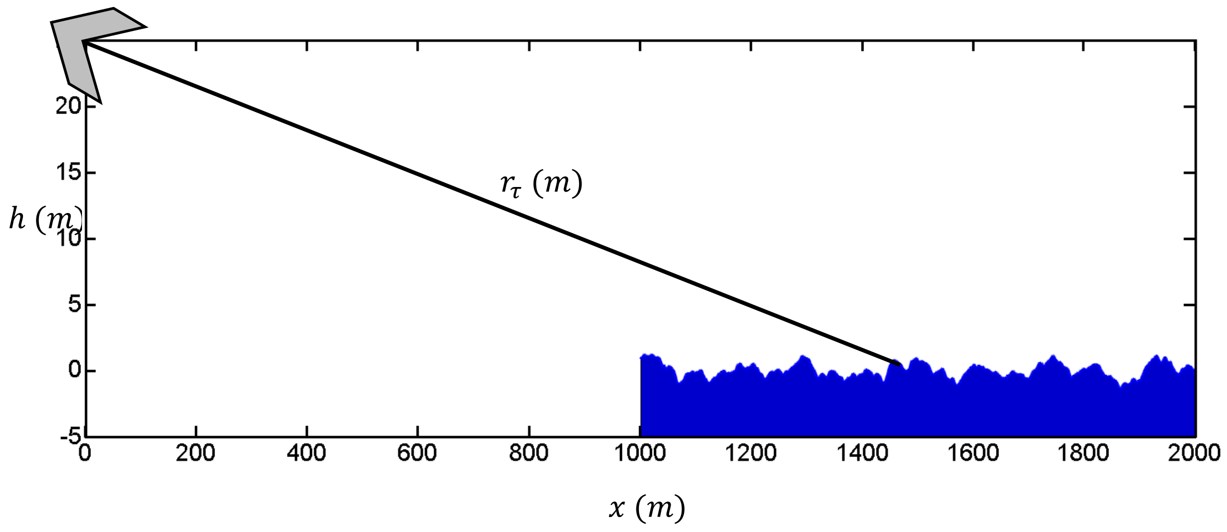


Figure 45. Non-scanning, one dimensional ocean radar return geometry.

After that, a modification is made to Equation (2 - 43) to correlate the ocean scattering strength to the slope of the ocean, as discussed in Section 2.1.3. To compute the slope of a point on the ocean, a simple first difference calculation using the previous and next distance sample, denoted as x_{-1} and x_{+1} respectively, was used. This process was extended to every point for every slow time:

$$s_\tau[t_s] = \frac{\boldsymbol{\eta}[x_{+1}, t_s] - \boldsymbol{\eta}[x_{-1}, t_s]}{x_{+1} - x_{-1}} \tag{2 - 65}$$

The previous or next distance sample is not valid for the end points. In these two cases, x replaces x_{-1} and x_{+1}, respectively. Notice that both positive and negative slopes are calculated using Equation (2 - 65). We noticed that the computed ocean scattering distributions (discussed next) fit better to the empirical distribution using positive and negative slopes compared to using modified slopes (positive only, absolute value, etc.). Using this slope, Equation (2 - 43) was modified:

$$V[\omega, t_s] = W_\omega[\omega] W_{t_s}[t_s] e^{-2jk \cdot R_s} \sum_{\tau=1}^{\mathrm{N_T}} s_\tau[t_s]\, e^{-2jkr_\tau[t_s]} \tag{2 - 66}$$

Such slope correlation was verified by visually checking for correlation, as shown in Figure 46. In our integration of radar and ocean, we considered the variable R_s in the equation above to be the range to the closest point of the ocean. In addition, we calculated $r_{alias-free}$ as the furthest point of the ocean. At this point, our model was capable of simulating radar returns of the ocean surface. These returns were range processed using an IFFT to obtain a RTI of the ocean surface.

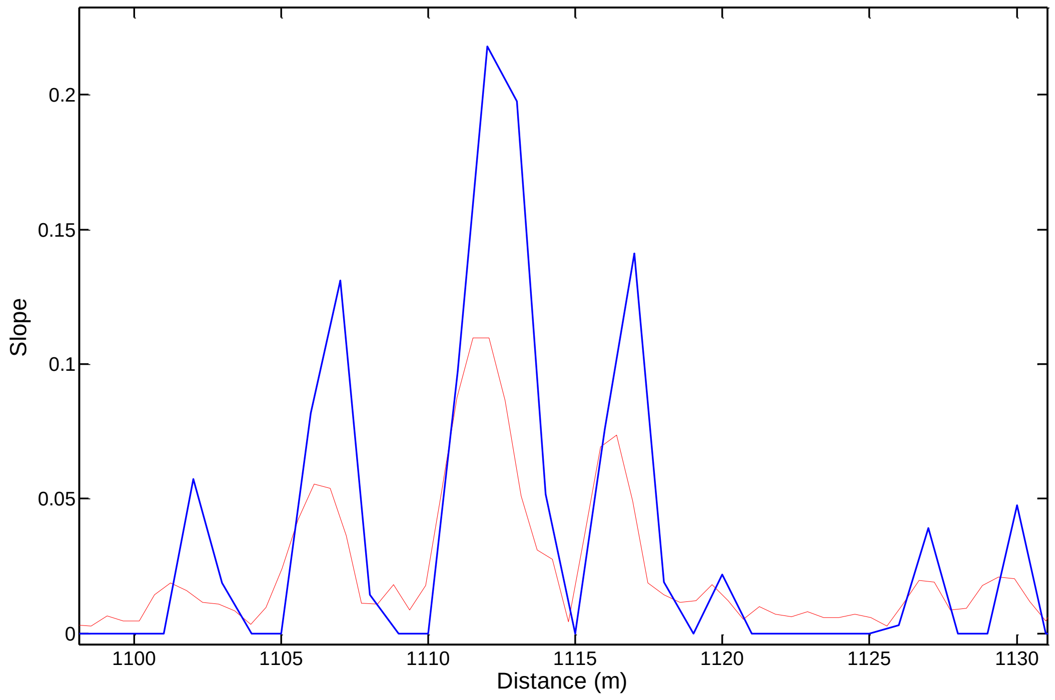


Figure 46. Comparison between slope (in blue) and radar return strength (in red).

The ocean scattering statistics of the RTIs of the ocean surface that our model produced did not correlate well to actual ocean clutter statistics, as shown in Figure 47. The distributions were plotted on a logarithmic scale to simplify processing. Note that the distribution presented is just one example: the computed ocean scattering distribution is a stochastic process. In addition, this is just one of the six empirical probability density functions (see Table 2). In order to make the calculated distribution fit to the empirical distribution reported by Long, the calculated distribution was calibrated.

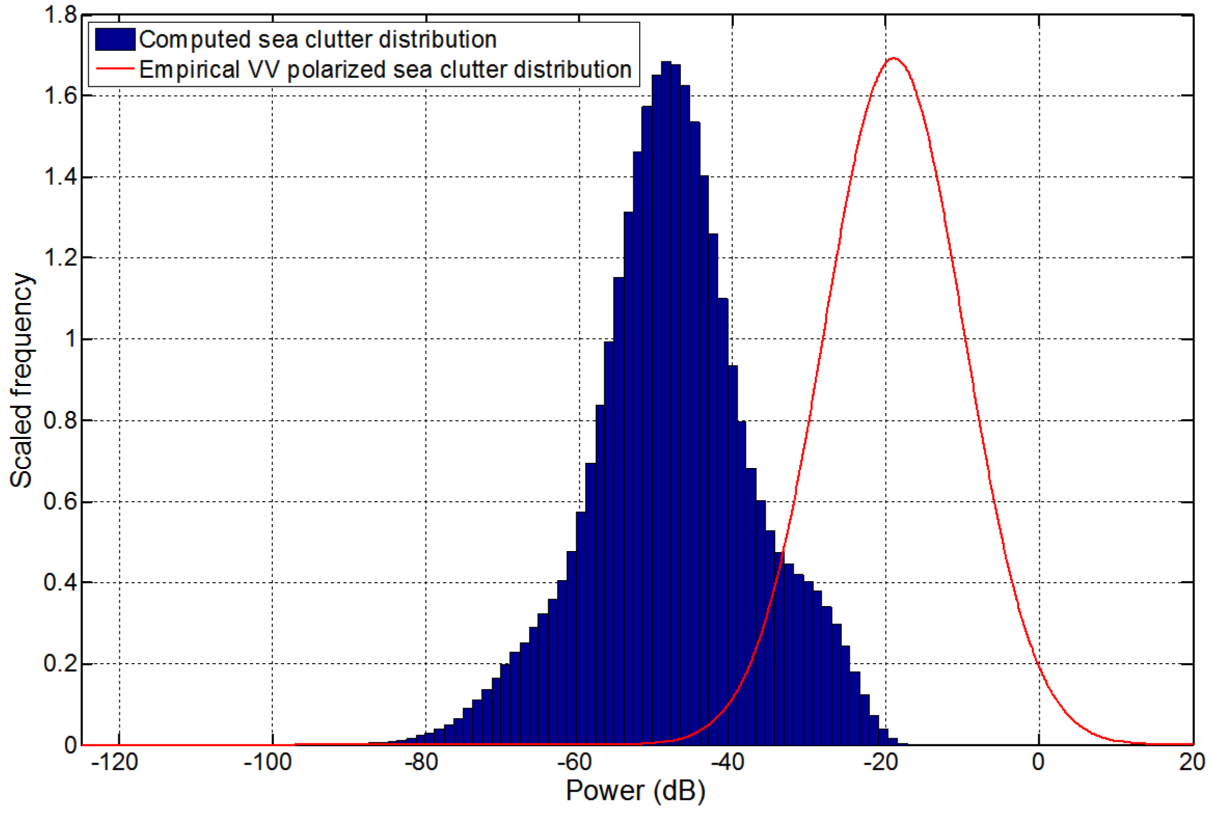


Figure 47. Computed vs. empirical ocean clutter distribution. The blue histogram shows a histogram of the computer sea clutter distribution and the red envelope represents the empirical clutter distribution reported by Long. The computed clutter distribution was generated with the following parameters: ocean generated from $1000\,m$ to $2000\,m$, $f_0 = 10\,Ghz$, $B = 1\,Ghz$, $AFMR = 1050\,m$, $grazing\ angle = 5°$, $prf = 20\,Hz$, downsea propagation, wind speed $= 13\,knots$.

The first step in the calibration algorithm was to calibrate the mode of the computed scattering distribution to the mode of the empirical distribution. In order to attain this result, the computed and empirical distribution modes were found, differenced, and then subtracted from each point of the ocean RTI in range and slow time. The mode of a log-normal distribution can be found according to (Wikipedia, 2012):

$$mode = e^{\mu-\sigma^2} \qquad (2 - 67)$$

For the empirical distribution, this was possible because mean, μ, and standard deviation, σ, were known; we denote this empirical mode as m_e. However, for the computed distribution, μ and σ are not known, so we found the mode more indirectly. Simply finding the most occurring power within histogram bins is problematic in two regards: first, the binning distorts the location of the actual mode; second, the computed distributions are sometimes skewed to one side, so the mode is not a good representation of center location.

A better center location was found using the following algorithm: first, the computed distribution was binned with very small bin intervals (on the order of 2000 bins). Because the distribution is stochastic, there are often large bin to bin variations. In order to reduce this variance, a fifth order infinite impulse response Butterworth low pass filter with cutoff frequency at one percent of the normalized frequency was applied both in the forward and reverse time directions to achieve zero phase filtering. The filter characteristics are shown in Figure 48. The startup transient to this filter is shown in Figure 49 in green. This transient does not significantly affect the envelope in the regions that are used to find the center location of the distribution. The filter produces low pass filtered histogram envelopes of the computed ocean scatter distribution. Next, the goal is to find two points in power on opposite sides of the peak of the envelope which correspond to $2/3$ of the maximum frequency. The fraction $2/3$ was chosen because it is far enough below the peak to be affected less by skewed distributions, while it is high enough above the non-Gaussian relation toward the bottom of the distribution. Two black arrows identify these two points in Figure 49. These two points were found using a find-closest algorithm, described in Appendix C. The average location of these two points was taken and is the estimated mode of the computed sea scattering distribution, denoted m_c.

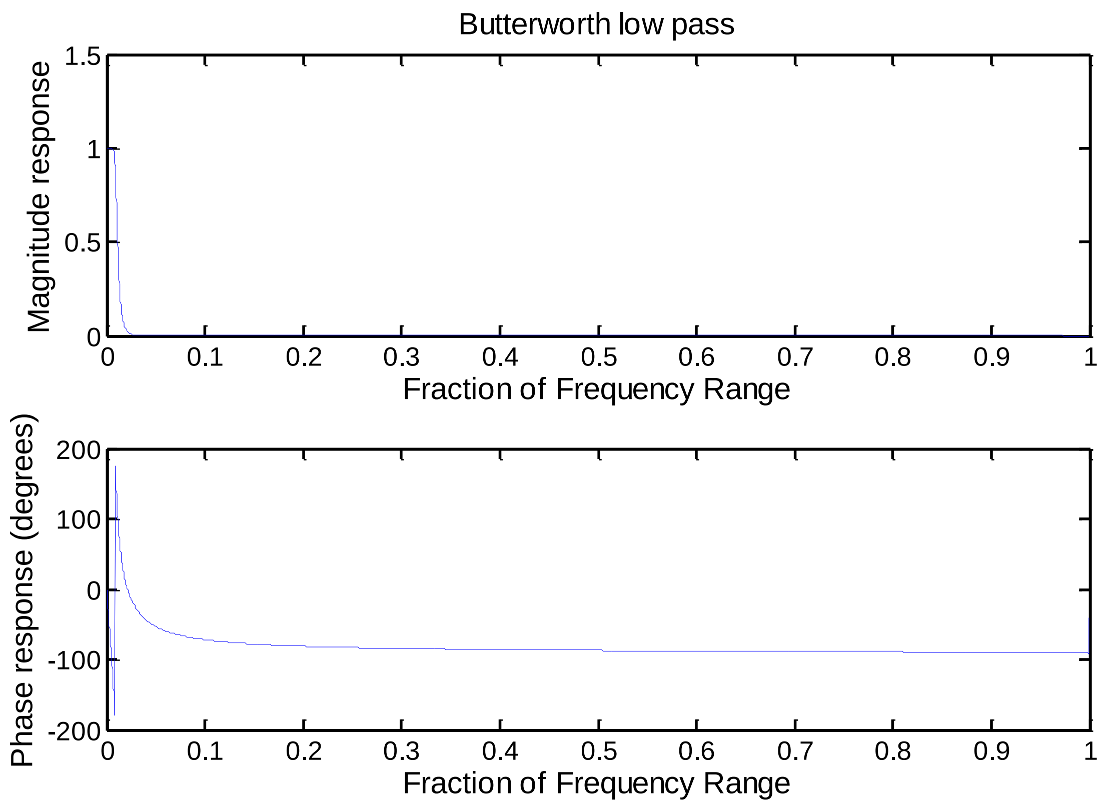


Figure 48. Magnitude and phase response of Butterworth lowpass filter.

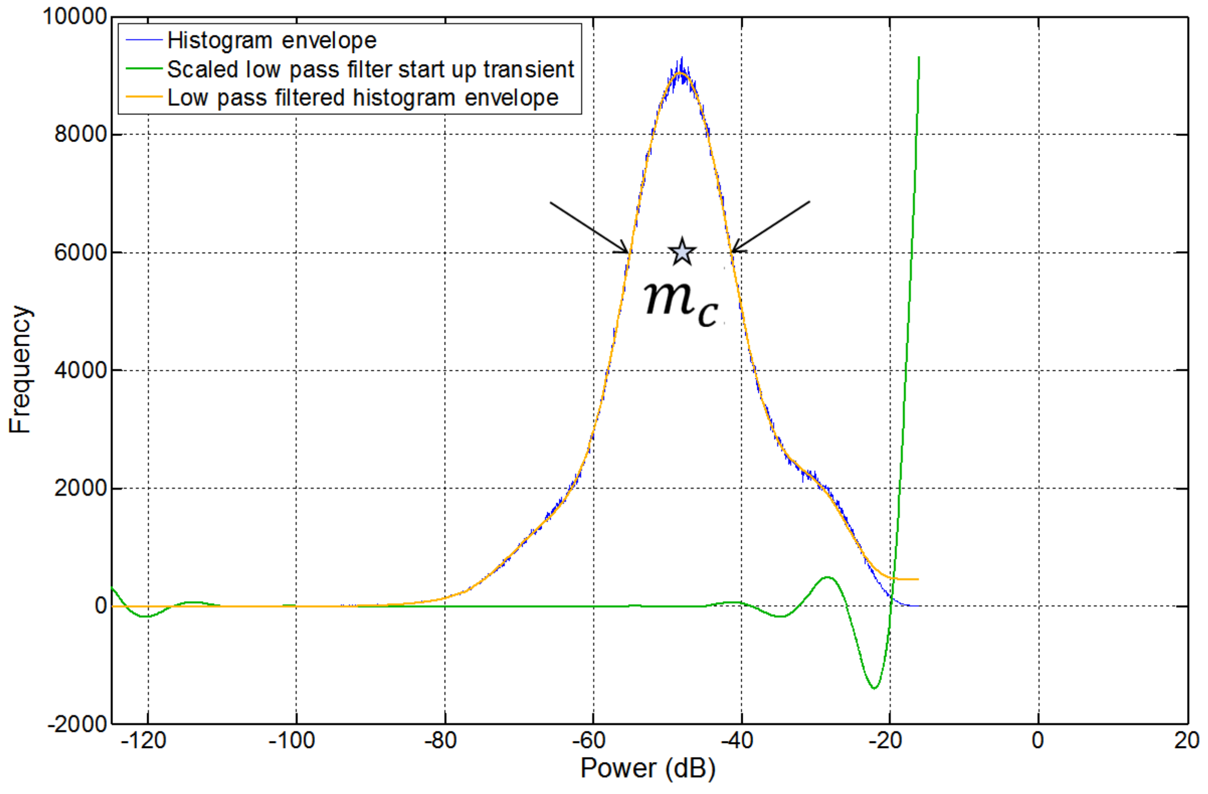


Figure 49. Computed distribution center location determination. The star represents the location of the center of the distribution.

The difference between the computed and empirical modes was taken according to:

$$\Delta m = m_c - m_e \tag{2 - 68}$$

This difference is the amount of required shift. The mode-shifted RTI, written as RIP_{ms}, is:

$$RTI_{ms}[r, t_s] = RTI[r, t_s] - \Delta m \tag{2 - 69}$$

The result is shown in Figure 50.

Inspection of Figure 50 shows that the calibration algorithm was not done: the widths of the computed and expected distributions were not the same (in this case they are fairly close, but consider the shape of the HH polarized case, where the distribution width spans up to 100 dB). Thus, it was necessary to dilate the mode shifted, computed ocean scattering distribution. This was accomplished by taking the ratio of lengths indicated in green in Figure 50. More specifically, the same $2/3$ of max frequency method was used to find a point to the left of the peak for both distributions. Because the modes were already known, the absolute values of the differences between the modes and these new left peak points are the indicated lengths. Each value in the mode shifted RTI is multiplied by this dilation factor, called g, to produce the dilated, mode shifted, RTI, denoted as $RTI_{ms,d}$:

$$RTI_{ms,d}[r, t_s] = g \cdot RTI_{ms}[r, t_s] \tag{2 - 70}$$

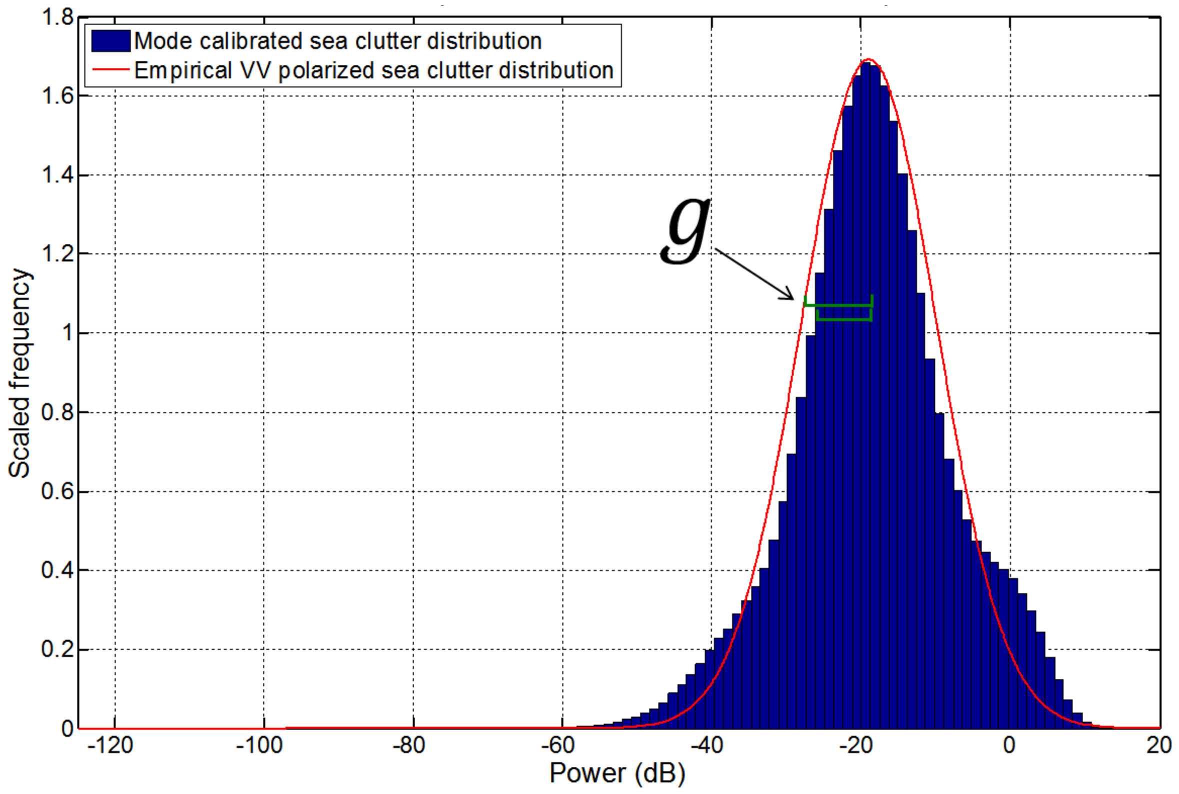


Figure 50. Mode-shifted, computed vs. empirical ocean clutter distribution.

The result is shown in Figure 51. Clearly, the dilation factor multiplication unaligned the modes. To further calibrate the mode shifted, dilated RTI, the mode-shifting algorithm described above was leveraged again to re-standardized the mode of the distributions. By denoting this second mode shift as Δm_2 and the mode shifted, dilated, mode shifted sea clutter range time intensity as RTI_{ms,d,ms_2}, we write:

$$RTI_{ms,d,ms_2}[r, t_s] = RTI_{ms,d}[r, t_s] - \Delta m_2 \qquad (2 - 71)$$

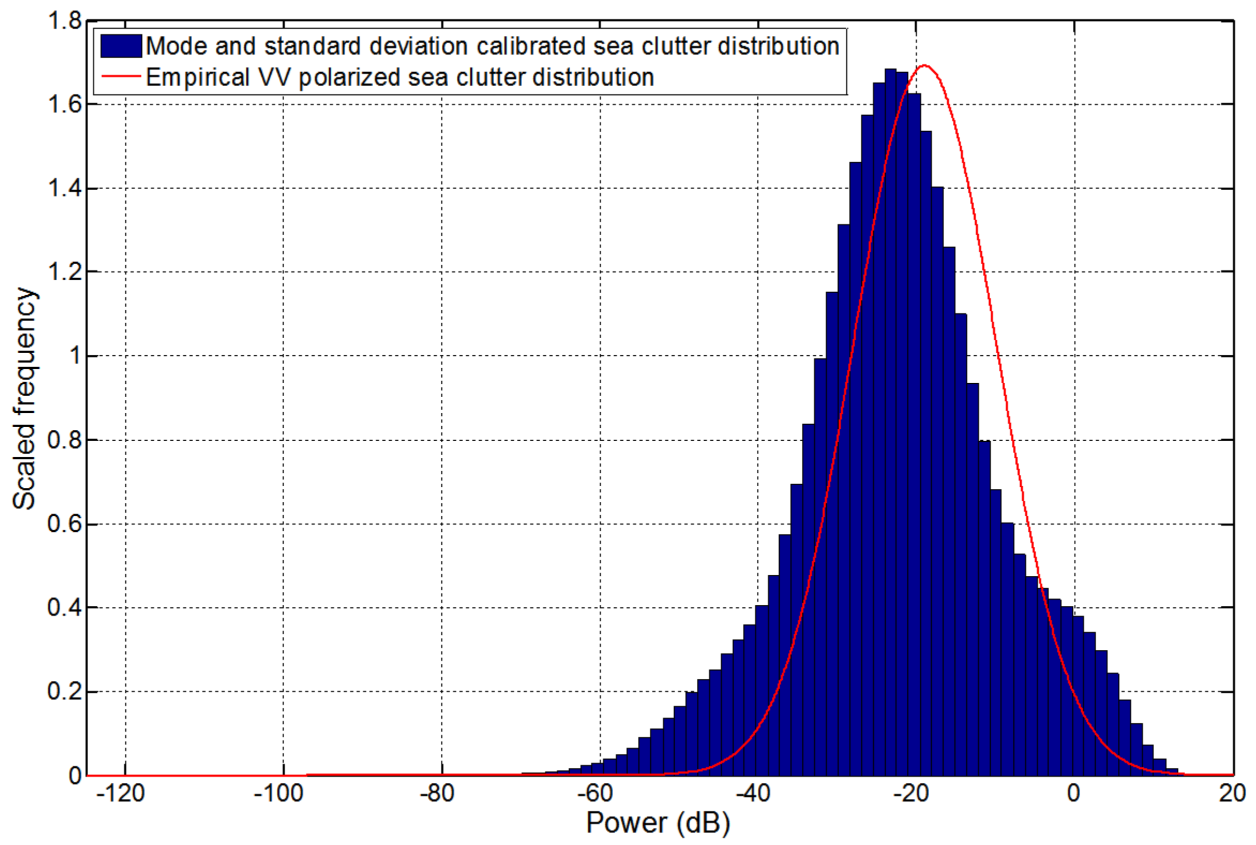


Figure 51. Mode-shifted, dilated, computed vs. empirical ocean clutter distribution.

We can express the entire calibration routine as:

$$RTI_{ms,d,ms_2}[r,t_s] = \left(g \cdot (RTI[r,t_s] - \Delta m)\right) - \Delta m_2 \qquad (2 - 72)$$

Note that this process can be simplified by first dilating and then shifting the computed distribution. Future revisions can make this modification. The final, calibrated RTI is shown in Figure 52. Clearly, the two distributions are very well aligned.

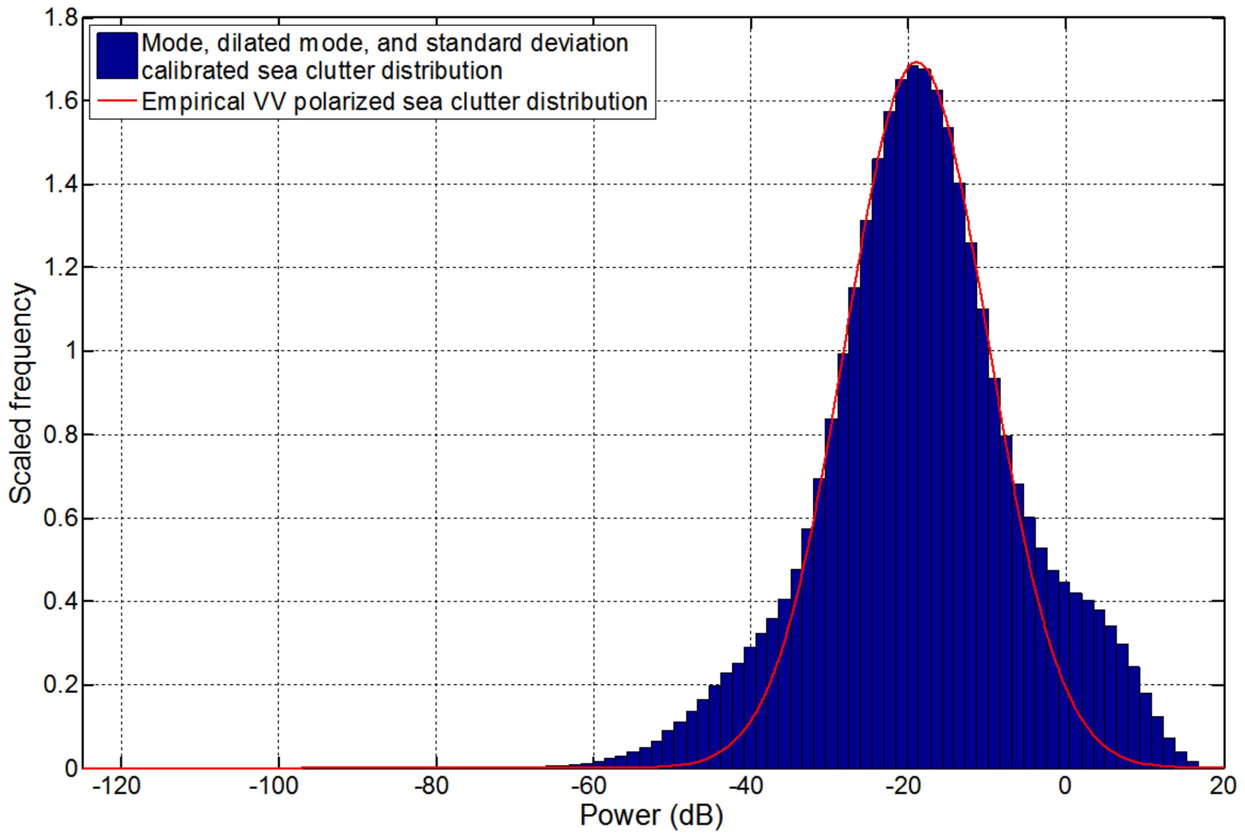


Figure 52. Mode-shifted, dilated, mode-shifted computed vs. empirical ocean clutter distribution.

Next, RTIs of simulated targets were computed as outlined in Section 2.2.1.4. We calculate the range of the boat as if it were on the ocean surface. That is, we apply Equation (2 - 64) to incorporate the ocean height at the target location for target range processing. In addition, we further scale the RTI according to Equation (2 - 23) where the area A_f is given by:

$$A_f = \frac{\alpha\ c\ r\ B_w}{2B} \qquad (2 - 73)$$

Note that there is no area or beam width for the one-dimensional ocean. In this case (and in the second deliverable), the B_w is arbitrarily set. In the third deliverable, the specified B_w is chosen.

To merge the RTI of simulated targets and the calibrated RTI of the ocean, simple addition was used. Addition is valid because RTIs are fundamentally Fourier transforms, which is an integral function. Because integrals are linear in their function space, the sum of the integral of two functions is the same as the integral of the sum of the functions. Analogously, the sum of two RTIs is the same as the RTI of the sum of point targets:

$$RTI_{targets+ocean}[r, t_s] = RTI_{targets}[r, t_s] + RTI_{ocean}[r, t_s] \qquad (2 - 74)$$

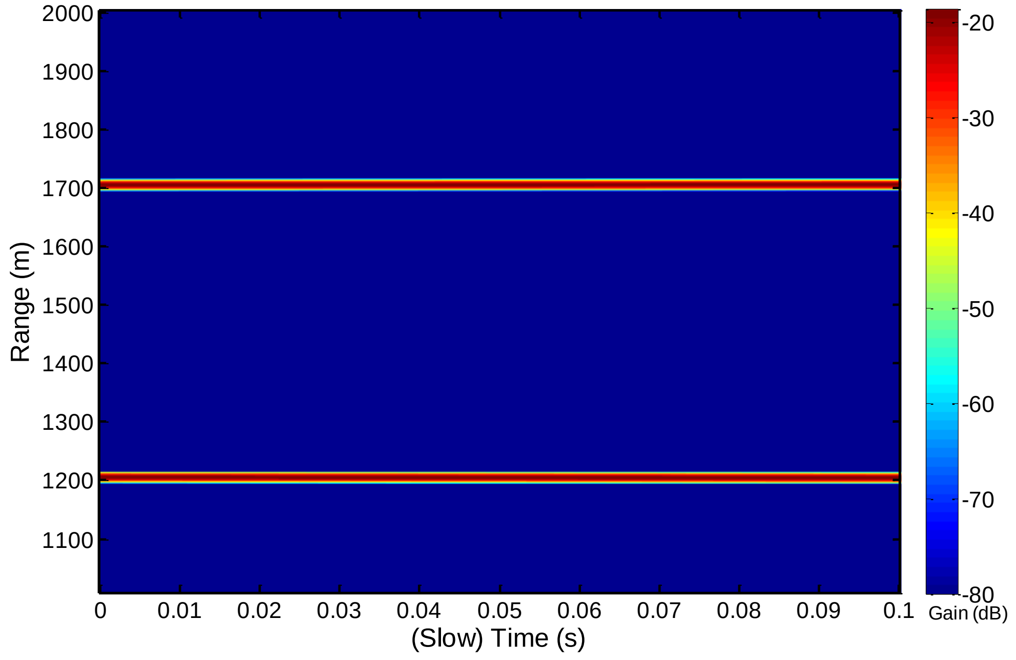
Finally, the RTI of the targets and ocean can be Doppler processed to produce range Doppler profiles. Furthermore, a time series of RDPs can be created in the same manner as discussed in Section 2.2.1.4. Some examples of these plots are shown in the following results section.

2.3 Results and Discussion

In this chapter, we have presented the background and methods for a non-scanning, one-dimensional ocean scattering simulation. Under time and computation constraints, we were very successful in developing a useful, easy to use, robust, and computationally effective simulation. Overall, this simulation also proved to be an excellent platform to experiment with parallelism and GUI because of its stability and ease of scalability. In this section, we first recount the major subcomponent results and then present the capabilities of the non-scanning, one-dimensional ocean scattering simulation. As we present results, we discuss the ramifications of our work. We particularly focus on the discussion of accuracy and verification of our model.

First, we presented the radar return equation as a function of frequency and slow time according to Equation (2 - 43) and its associated results: the RTI and RDP time series. An example RDP time series with two accelerating point targets is included in the accompanying attachment to this report. Figure 53 shows a different example of the RTI and RDP than those shown in Section 2.2.1.4. We note that such processing results are easily validated via comparison to known trajectories and to similar plots found in the literature. We believe that our method of producing these plots balanced complexity and constraints very well because we chose a modeling approach that produced verifiable results while limiting complexity. For example, we reduced complexity through approximation in processing which had negligible effects on the end result. Using a gate representation of the chirp waveform and pulse compression via Fourier transform instead of the chirp waveform and matched filter exemplifies such approximations. Another example of an approximation that led to less complexity was our decision to not model the range dependence of radar return power (see Equation (2 - 37)). We neglected modeling this phenomenon because the dependence of range on radar return power is commonly filtered out in radar processing.

(a)



(b)

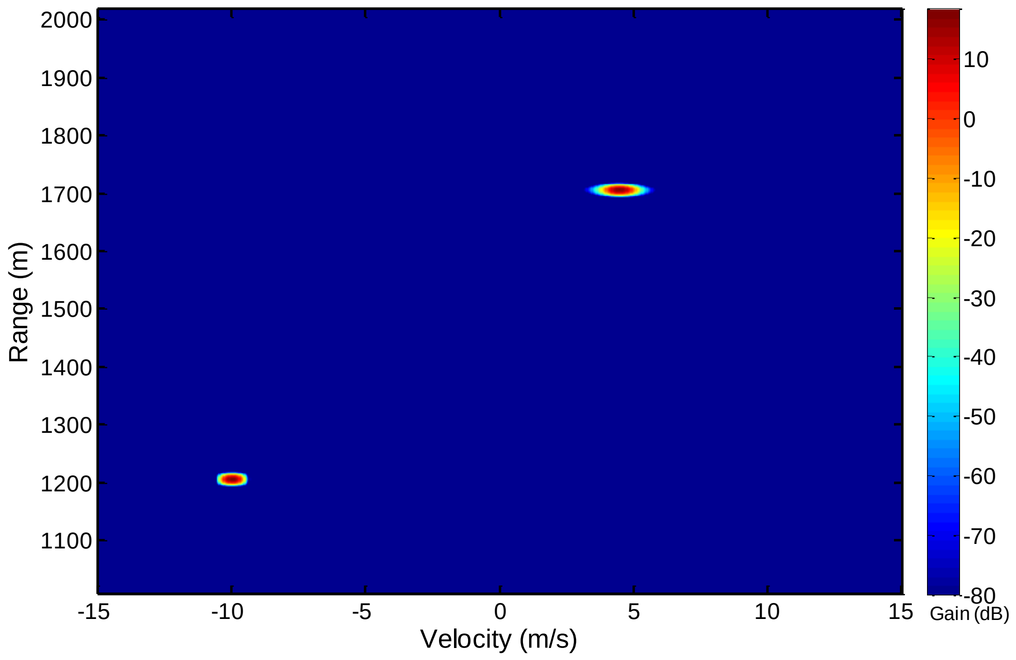


Figure 53. Example of a range time intensity plot (a) and range-Doppler profile (b) for two targets. One target starts $1200\ m$ away from the radar and is travelling away at $10\ m/s$ with no acceleration. The other target starts $1700\ m$ away from the radar, travels toward the radar at $5\ m/s$, and has acceleration of $10\ m/s^2$ away from the radar. Other parameters include: $f_0 = 10\ GHz, B = 50\ MHz, grazing\ angle = 5°, d_{alias-free} = 1000\ m, v_{alias-free} = 15\ m/s, T_{dwell} = 0.1\ s$, Kaiser windowing.

After developing a radar processing simulation, we developed a one-dimensional ocean model. Sample ocean surfaces for wind speeds of 13, 18.5, and 24 knots are shown in Figure 54. A time series movie of a sample ocean surface can be found in the accompanying attachment to this report.

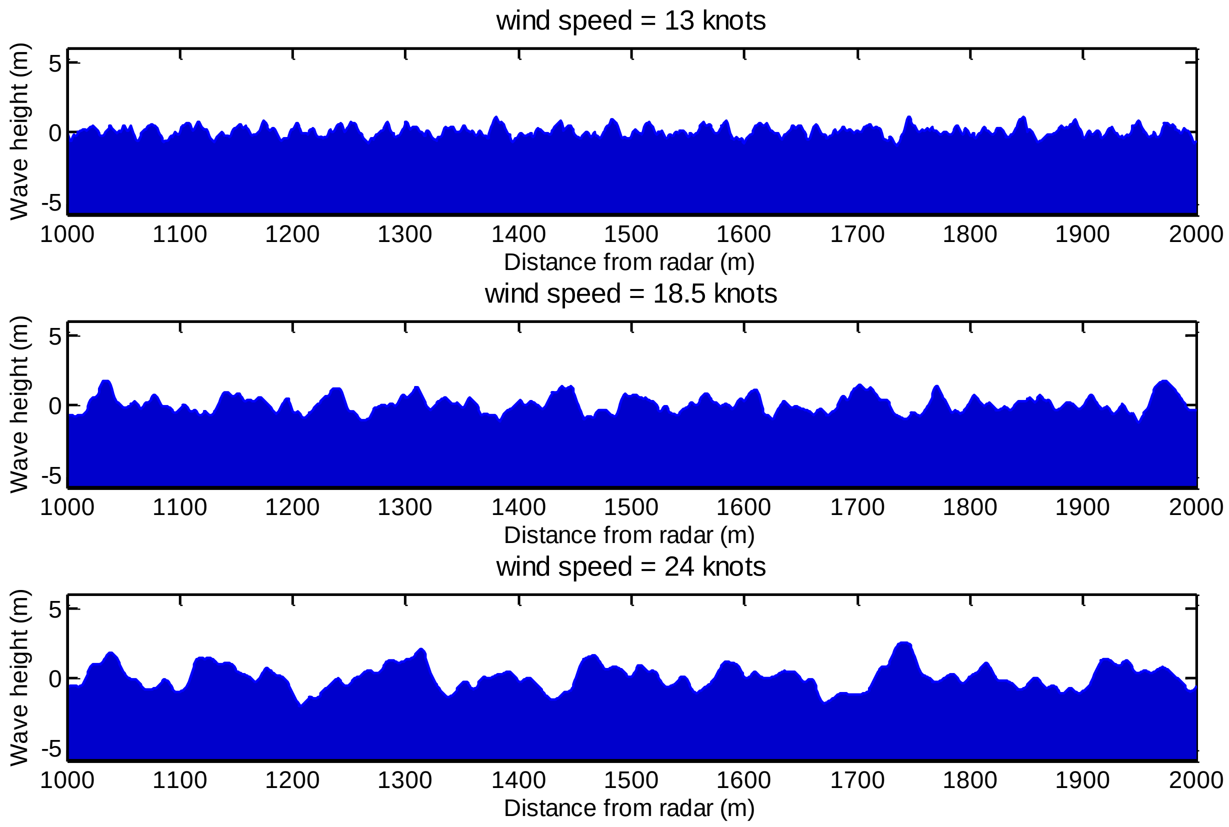


Figure 54. Comparison of three ocean surfaces with 13, 18.5, and 24 knot wind speeds, respectively. The distance resolution is 1 m and the ocean length is $1000\ m$.

We verified two aspects of our ocean model. The first verification was the distribution of wave heights. We confirmed that the distribution of wave heights matched the distribution described by the Beaufort scale. This match was easily verified because we directly scaled the significant wave height (see Equation (2 - 61)). A plot of the distributions for wind speeds of 13, 18.5, and 24 knots are shown in Figure 55. To generate this figure, a histogram with 10,000 bins was applied to an ocean with a length of 500 km and centimeter distance resolution was generated. We notice that the distribution is Gaussian. Recall that the wave height function is a sum of a series of randomly phased sinusoids. The Central Limit Theorem states that the distribution of the sum of a series of independent, identically distributed random variables is approximately Gaussian (Papoulis, 2002).

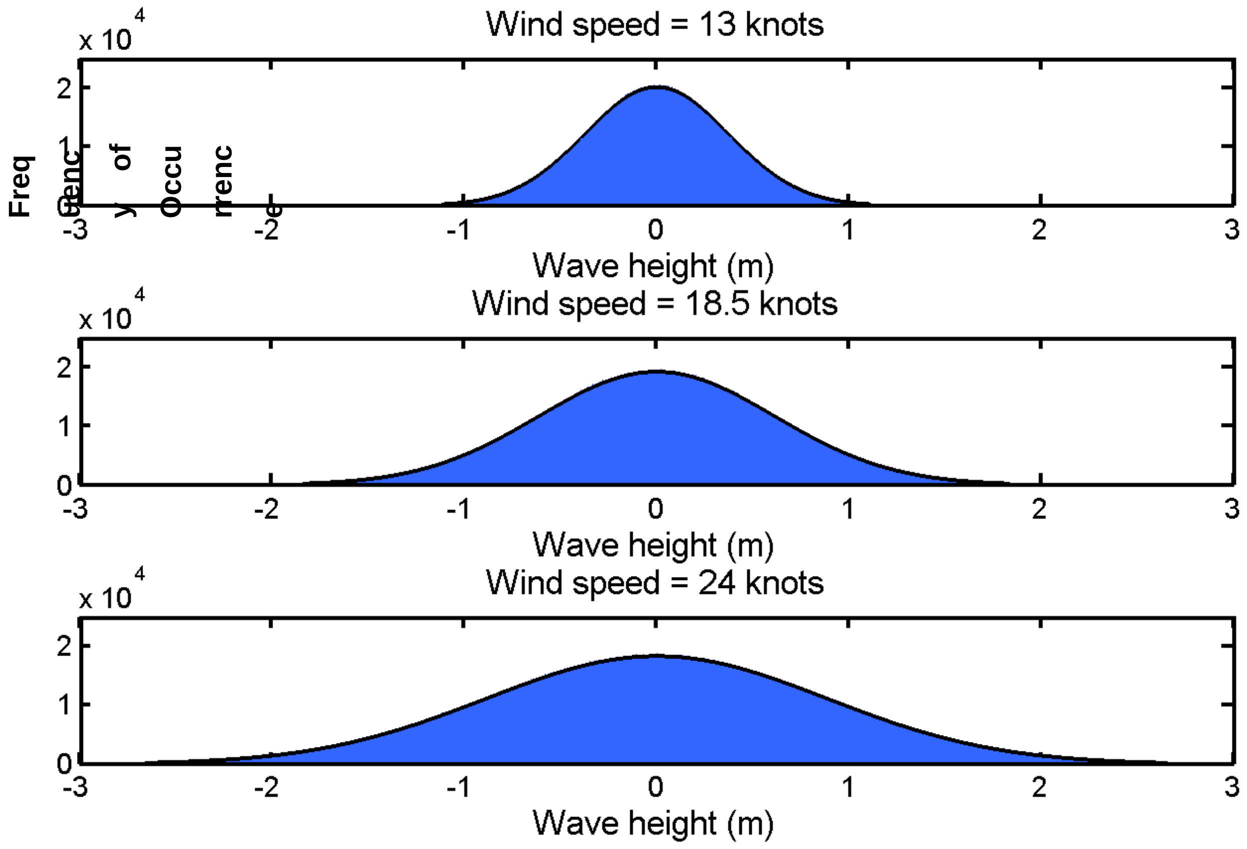


Figure 55. Wave height distribution of three ocean surfaces with 13, 18.5, and 24 knot wind speeds, respectively. The distance resolution is 1 cm and the ocean length is 500 km. The blue region represents the histogram of the wave heights of the ocean and the overlaid black envelope represents fitted Gaussian distributions.

We note that this scaling was necessary due to the Pierson-Moskowitz amplitude scaling approximation that we used. The first approximation was that we evenly sampled the Pierson-Moskowitz spectrum and derived the power of each sinusoid from the square root of the sampled value. This square root of the sampled value was an approximation itself. A more accurate representation would have considered a scaling factor, density, and the acceleration due to gravity (Kinsman, 2002). Ideally the ocean model would not require scaling. To form a more accurate model, we recognize the Pierson-Moskowitz spectrum is a density function; thus, the power of each sinusoid is more accurately represented by the area surrounding the frequency sample (a Riemann sum method or similar). Such a method would more accurately portray the wave heights. As a result, it may not be necessary to scale the wave heights to the Beaufort scale. We also verified that the power spectral density of our ocean surface matched the Pierson-Moskowitz spectra using a Welsh periodiogram. Three oceans were used composed of a distance of 10 km with 1m resolution and 4000 seconds with 1 s resolution. The Welsh periodiogram was defined by eight sections of approximately 2200 ocean distance samples, with 50% overlap, Hamming windowing, and taken for each time instant. It was then averaged across time to produce the result. The power spectral density (PSD) of the ocean was scaled to the PM spectrum. The results are shown in Figure 56. The scale between the actual and expected PSDs is insignificant because only the shape is important for our model. Notice that the power spectral density (PSD) of the ocean is seemingly comprised of many impulse functions due to the sampling of the PM spectra as presented in Section 2.2.2.

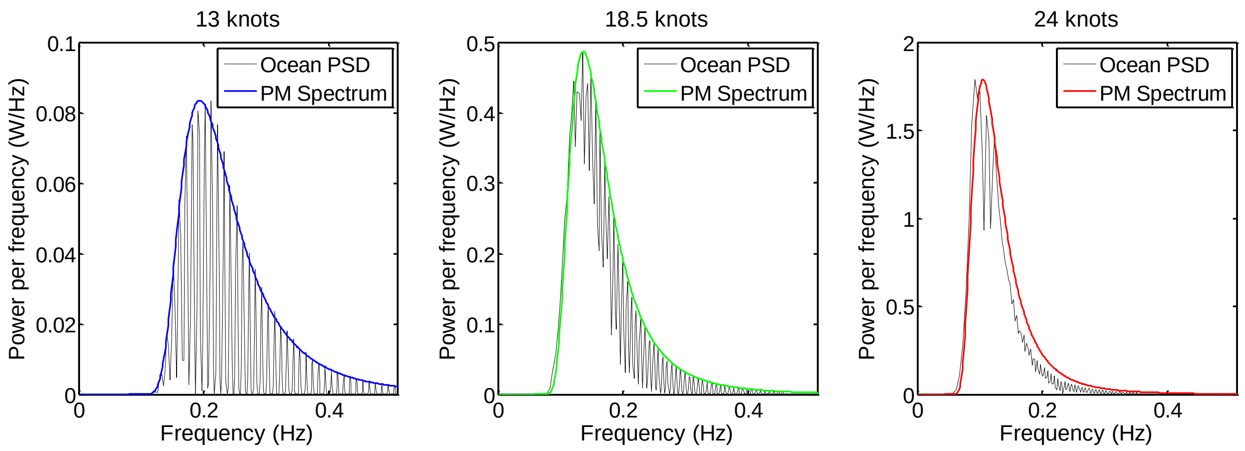


Figure 56. Power spectral density of the ocean model versus PM spectrum for wind speeds of 13, 18.5, and 24 knots.

We believe that this model approximates the surface and the propagation of the ocean to the required standard for later integration with radar processing; thus, it is a desirable balance of complexity and constraints for this project. More specifically, we could have developed a more rigorous ocean model that, for example, included capillary wave interactions (greater than 48 Hz ocean waves). However, the extra amount of time to develop and compute such model would not be significantly advantageous because our computational constraints limit the distance resolution of the ocean (more distance resolution leads to more point targets, which drastically increases computation time and, more importantly, memory required). With such limited distance resolution, the simulation would not be able to resolve capillary waves.

Once the ocean model and radar simulation were complete, they were combined to find the RTI of the ocean surface. After that, the RTI was slope scaled and calibrated to ocean scattering distributions found in literature. We compare un-calibrated and calibrated ocean surface RTI plots below in Figure 57. We note that mode shifting in the calibration algorithm is equivalent to adjusting the gain of the radar or scaling up all clutter cross sections. We believe that the resultant calibrated RTI is an accurate representation of the ocean scatter from an ocean surface, according to the distributions we found in the literature. However, the very fact that our scattering distribution did not fit distributions found in literature and that it was necessary to scale our model suggests a limitation in our simulation. Specifically, we did not model the effects of polarization. We expect that had we modeled these effects, our generated ocean scattering distributions would have had much better fit to those found in literature. Future work should consider such effects. In turn, the method we chose to model polarization (scaling to known distributions) is dependent on the accuracy of the data found in the literature. Another contributing factor that made it necessary to calibrate the ocean scattering distributions is that the Pierson-Moskowitz spectra is only an approximation because it is result of a curve fitted to empirical data. A power spectral density that is better matched to realistic ocean waves may produce ocean scattering distributions that are more log normal. Unfortunately, a major limitation is that there are little data available in the public domain. Furthermore, we note that the linear interpolation that we assumed for the data reported by Long may be invalid. Specifically, we noticed that the ocean scattering distributions appeared less log-normally distributed as the wind speed increased. Thus, to make our simulation useful to our sponsor in the future, we developed an easy way to modify the calibration distributions.

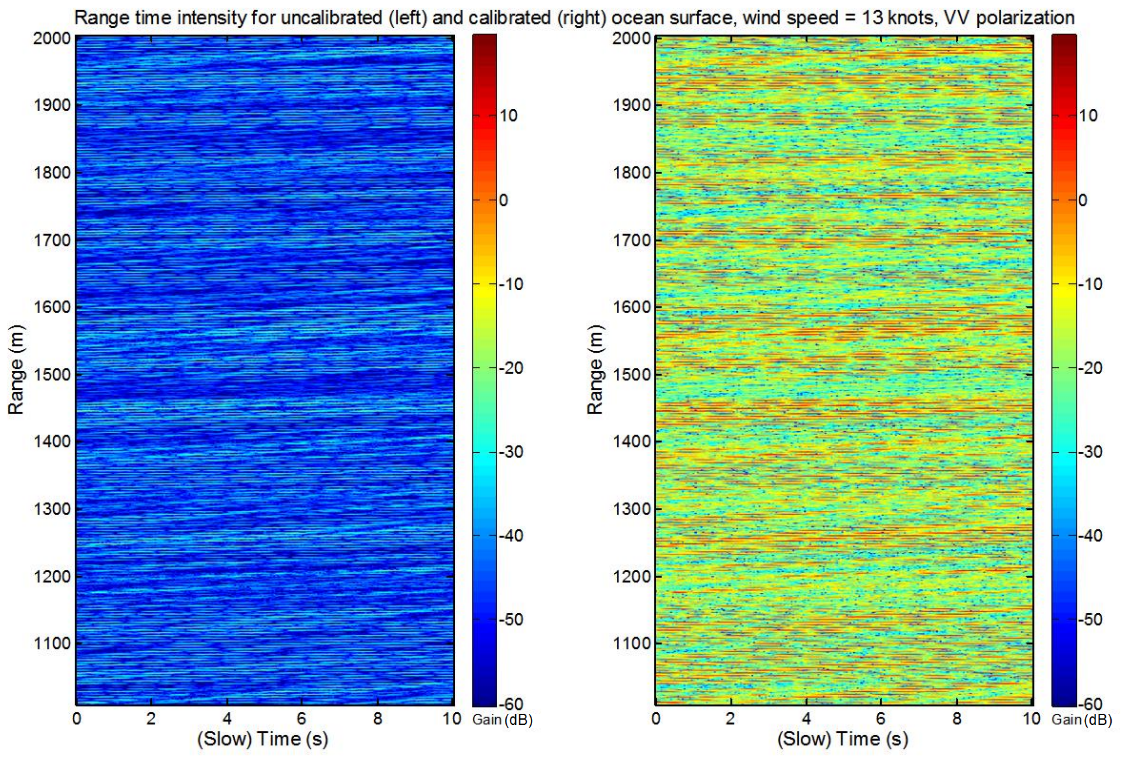


Figure 57. Comparison between uncalibrated (left) and calibrated (right) RTIs of the ocean surface. Parameters include: $f_0 = 10\ GHz, B = 50\ MHz, grazing\ angle = 5°, d_{alias-free} = 1000\ m,$ $v_{alias-free} = 15\ m/s\ , wind\ speed = 13\ knots,$ VV polarized.

Finally, we modeled targets on the ocean surface. The target height was set to the height of the ocean at the location of the target. In addition, the cross section of the targets was set to the mean cross section of the radar scatter. Future work should explore cross section modeling in more depth. A kinematic distance vs. time plot of two sample targets is shown in Figure 58. A sample RTI plot of the ocean surface with these two targets is shown in Figure 59. A sample RDP of the same two targets on the ocean surface is shown in Figure 60. A time series movie of this RDP can be found in an accompanying attachment to this report. We see that most of the power of the RDP is located at low velocities. Thus, it is more difficult to detect a slow boat as compared to a fast moving boat. Such time series of RDP of targets and the ocean surface will help our sponsor develop algorithms to better detect small boats on the ocean surface.

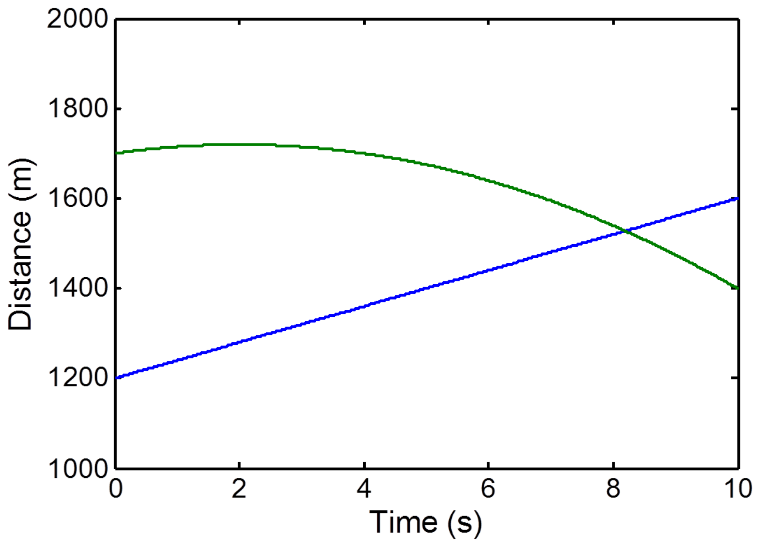


Figure 58. Distance vs. time for two sample targets. One target starts $1200\ m$ away from the radar and is travelling away at $10\ m/s$ with no acceleration. The other target starts $1700\ m$ away from the radar, travels toward the radar at $5\ m/s$, and has acceleration of $10\ m/s^2$ away from the radar.

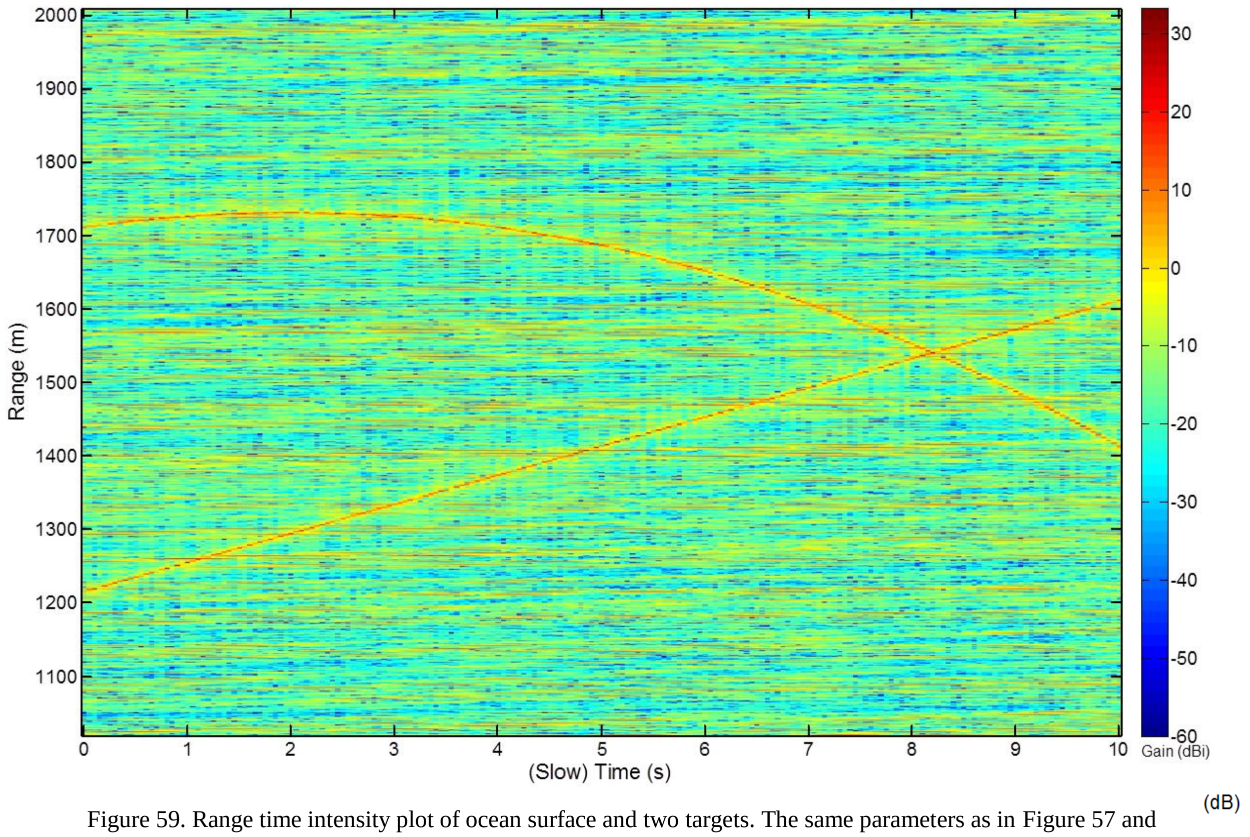


Figure 59. Range time intensity plot of ocean surface and two targets. The same parameters as in Figure 57 and Figure 58 were used. Notice the similarity between this figure and Figure 58.

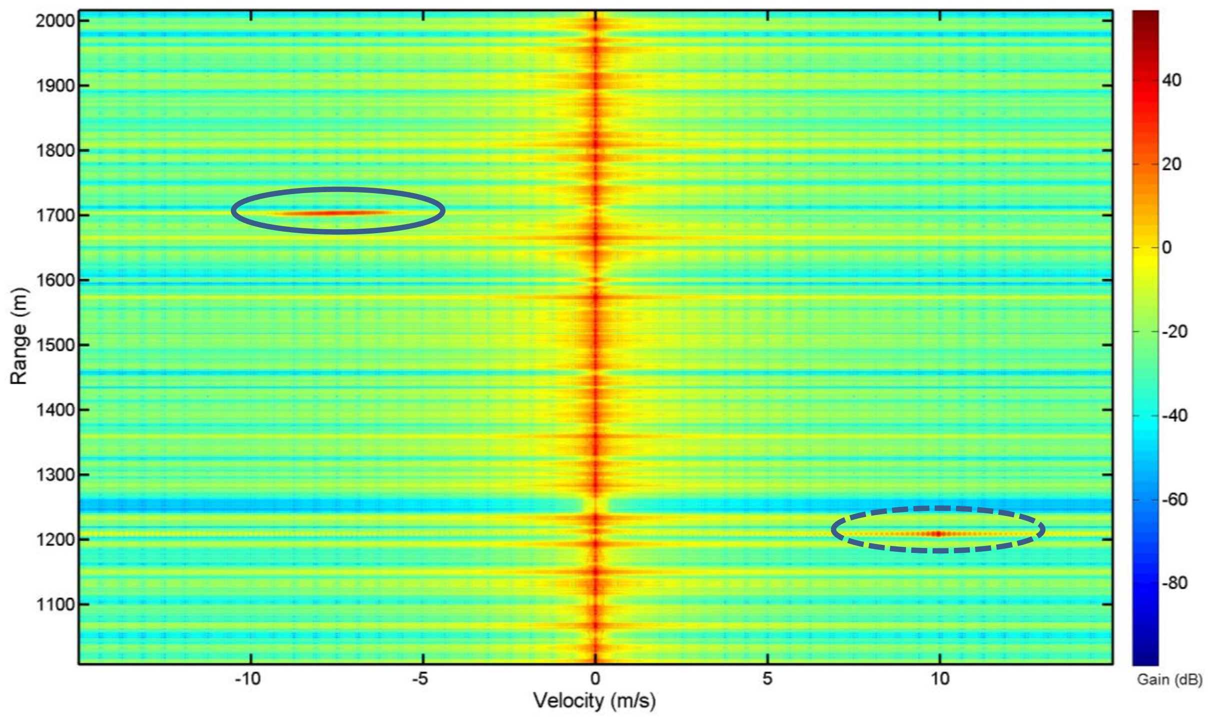


Figure 60. Range Doppler profile of ocean and two targets. Parameters include: $f_0 = 10\ GHz, B = 50\ MHz,$ $grazing\ angle = 5°, d_{alias-free} = 1000\ m, v_{alias-free} = 15\ m/s, wind\ speed = 24\ knots, t = 0\ to\ 0.5\ s,$ VV polarized. The same targets as in Figure 58 were used. The solid circle highlights an accelerating target while the dashed circle highlights a target with constant velocity.

Finally, we note that the method used to generate the motion of the point targets on the ocean surface is flawed for Doppler generation. We came to this realization after noticing that the computed velocity information of the ocean surface was inconsistent with expected ocean surface velocity information. Specifically, our model predicts velocities under 1 m/s, while we expect velocities up to 2 m/s (D. Blejer, personal communication, September 27, 2012). Recall that we generated ocean surface point targets which vary in height as a function of time. This description perfectly lends itself to ocean surface shape and characteristics and range processing. However, it inaccurately measures the Doppler information of the ocean because the point targets are stationary in distance (yet, not stationary in range – see the discussion surrounding Figure 45). Thus, we only calculated the vertical velocities of the ocean, and not the propagation components. We have adjusted our model through velocity scaling to more accurately portray the Doppler phenomenology of the ocean. A more accurate, yet more complicated model may be a Lagrangian mechanics particle model with use of the material derivative (Currie, 1974, pp. 6-8). Future work can investigate Legrangian mechanics further.

3 Non-Scanning, Quasi-Two-Dimensional Ocean Radar Scattering Simulation

In order to better model the real-world, it is useful to simulate the radar returns of a target on a two-dimensional surface, visually represented in Figure 61. A one-dimensional ocean model limits target modeling to one-dimensional movement. With a two-dimensional ocean model, it is possible to more accurately model target movement. In addition, the extra degree of spatial freedom allows boat wake modeling, as discussed in Sections 3.1.2 and 3.2.2.

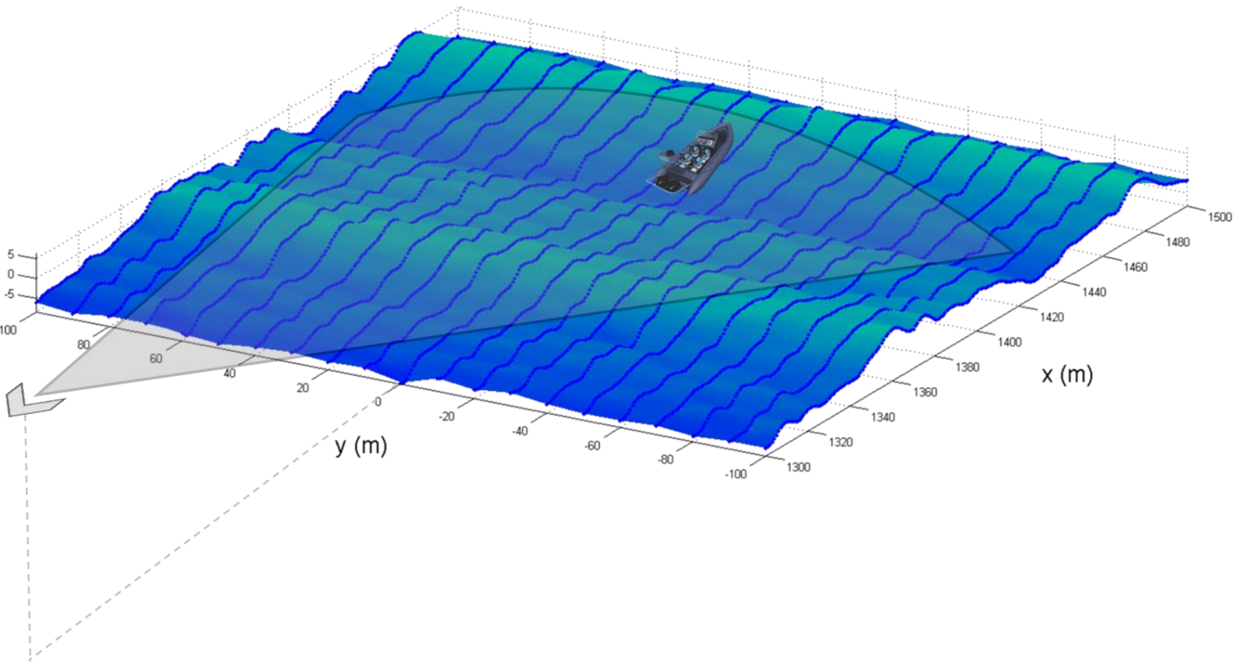


Figure 61. Visual representation of non-scanning radar, quasi-two-dimensional ocean, and target integration.

3.1 Background

Following is the background that pertains to modeling a quasi-two-dimensional ocean model and small boat wake model. We then discuss use of the Kelvin wake cusp wave crest as a method for modeling the wake of a boat moving at constant velocity.

3.1.1 Quasi-two-dimensional ocean model.

Similar to the one-dimensional ocean model discussed in Sections 2.1.2, there are many existing two-dimensional ocean models. One of the most popular and accurate models is the Phillips model, which supports multi-directional ocean propagation directions (Tussendorf, 2001; Mitchell, 2005). In this model, the relative power of each of these propagation directions is a function of wind speed and wind direction. While implementing such a two-dimensional ocean model would be ideal, the Phillips model would also add complexity that is out of the scope of this project. An ocean model that is suitable for our scope is the quasi-two-dimensional ocean model, which considers ocean propagation in one direction. The unidirectional propagating ocean represents the most extreme ocean clutter scenarios. Because the radar returns of the ocean are the strongest in the up-sea and down-sea cases, the one-dimensional propagation of the quasi-two-dimensional ocean models these cases very well (D. Blejer, personal communication, August 8, 2012). If a detection algorithm works well for the quasi-two-dimensional ocean, it will likely work well for multi-directional propagating two-dimensional oceans.

3.1.2 Small boat wake model.

Wakes are defined as patterns produced by moving objects in a fluid. This project refers to wake as the water patterns developed by moving boats. In this context, there are four primary types of wake: turbulent wake, Kelvin wake, narrow-V wake, and internal wake. Although each of these types of wakes cause detectable patterns in radar, turbulent, narrow-V, and internal wakes are not considered in our model because the Kelvin wake has the largest effect on radar returns, and thus is most likely to be mistaken for a small vessel. The complexity of the model can be reduced by only considering Kelvin wake.

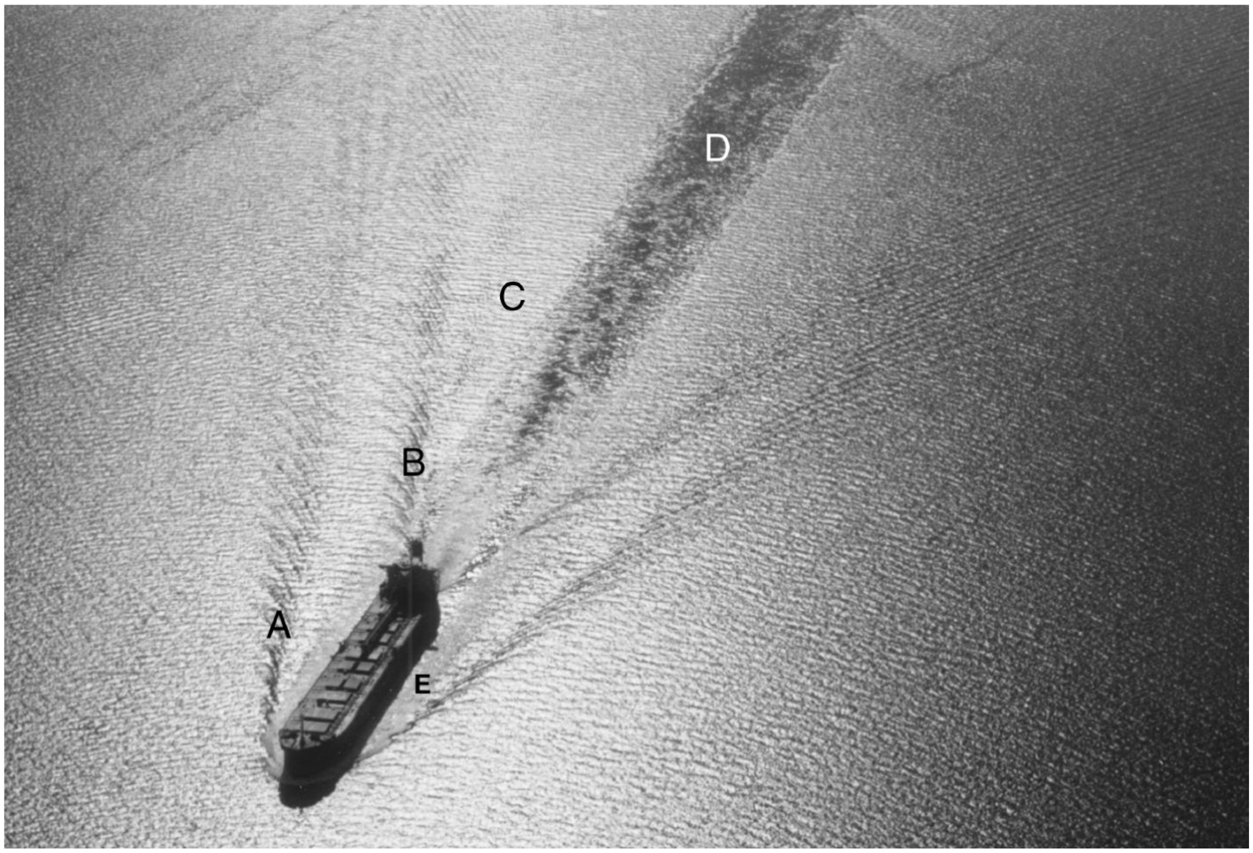


Figure 62. Example of Kelvin wake and turbulent wake. A) Kelvin envelope from bow, B) Kelvin envelope from stern, C) Kelvin wake transverse wave, D) turbulent wake, and E) turbulent region along side of ship (Jackson & John, 2004, p. 285).

Under the assumption that a boat has a constant velocity, constant direction, and is in deep water, Kelvin wake is predictable. A Kelvin wake consists of transverse and divergent waves that are produced by the movement of a boat through water. As seen in Figure 63, these waves form a 38.94 degree V shaped pattern behind the boat called the Kelvin envelope. The 38.94 degree angle is valid for a wide range of boat velocities and shapes due to the dispersion relation of deep water waves (Whitham, 1999, p. 409). Therefore, in our model, we always use this angle. There are two Kelvin envelopes that are produced by each boat, one from the bow and one from the stern of the boat. The Kelvin envelope produced from the bow of the boat is much larger in amplitude than the Kelvin envelope from the stern of the boat; thus, we focus on modeling only the former.

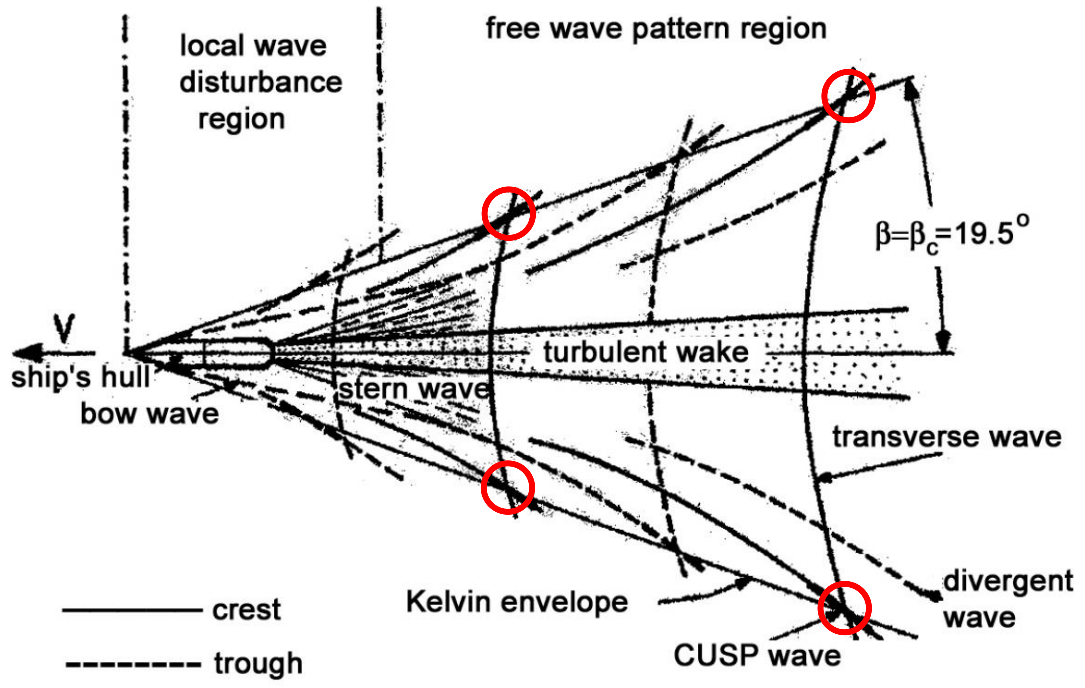


Figure 63. Kelvin wake representation. Circles represent location of the crests of the cusp waves (Hennings, Romeiser, Alpers, & Viola, 1999, p. 2522).

Furthermore, Kelvin envelopes are composed of two Kelvin arms, formed by the arms of the V-shaped pattern. Thus, each arm is angled 19.47 degrees from the center of the Kelvin envelope. The Kelvin arms are formed by cusp waves, which are the superposition regions of transverse and divergent wave fronts. The highest peaks formed by such superpositions are referred to as cusp wave crests, as shown by the circles in Figure 63.

The cusp wave crests are particularly important to our radar target model. Because the cusp wave crests generate the largest positive perturbations on the ocean surface, they produce the largest radar returns. Thus, to reduce complexity, we consider it sufficient to only model cusp wake crests in order to model boat wakes.

Cusp waves are always angled at a 35.26 degree offset from the direction of travel of the boat, and face outward on each Kelvin arm: 35.26 degrees is the propagation angle of the cusp wave. The orientation of the cusp waves can be seen in Figure 64. The cusp waves occur at intervals along the Kelvin arms based on the velocity of the boat.

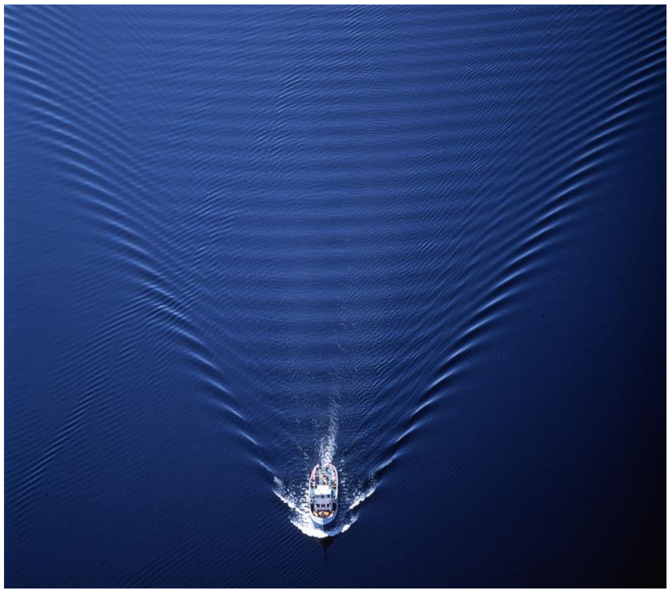


Figure 64. Example of Kelvin wake ("Particles and Waves", n.d.).

The angle of each Kelvin arm and the angle of the cusp wave propagation direction do not depend on velocity; however the cusp wave phase speed and wavelength do depend on velocity. According to Hennings, Romeiser, Alpers, and Viola (1999), these values are described by,

$$V_{cusp} = V_{boat} \cos \phi_{cusp} \qquad (3 - 1)$$

$$\lambda_{cusp} = \frac{2\,\pi\, V_{cusp}{}^{2}}{g} \qquad (3 - 2)$$

where V_{cusp} is the phase speed of the cusp wave, λ_{cusp} is the wavelength of the cusp wave, V_{boat} is the velocity of the boat in meters per second, g is the acceleration of gravity, and ϕ_{cusp} is the propagation angle of the cusp wave (p. 2520).

3.2 Methods

First, we discuss the extrapolation of a one-dimensional ocean through orthogonal duplication resulting in a quasi-two-dimensional ocean. Then, we add random height variability to the ocean and present surface smoothing. The addition of boat wake to the ocean surface and final integration of the radar, boat wake, and ocean into a single model are also presented.

3.2.1 Quasi-two-dimensional ocean model.

The quasi-two-dimensional ocean was generated using an algorithm that converts the one-dimensional simulated ocean model into a quasi-two-dimensional propagating ocean. The model requires the output of the one-dimensional ocean model. We developed three options for the quasi-two-dimensional ocean model as requested by the sponsor. This model uses the stored one-dimensional ocean height as a function of distance and time. These data and cross-sea input parameters define the quasi-two-dimensional ocean model. Note that in our quasi-two-dimensional ocean model, we refer to the x dimension as the up/down sea direction and we refer to the y dimension as the cross-sea direction.

There are ten inputs to our algorithm comprised of:

- Cross-sea range
- Cross-sea distance resolution
- Wave type toggle
    - uniformly extrapolated
    - uniformly extrapolated with additive independent height variance
    - uniformly extrapolated with additive correlated height variance
- Amount of random height variance
- Time smoothing interval
- Cross-sea smoothing interval
- Up/Down-sea smoothing interval
- Plotting toggle
- Ocean select
- Small ocean plot toggle
- Small ocean plot bounds

The cross-sea range and distance resolution inputs are used to create a discretized cross-sea distance vector. Toggle wave type input allows the user to select from the three following ocean options: uniform waves (Figure 66), uniform waves with added independent random height variance (Figure 67), and uniform waves with added correlated random height variance (Figure 68). The amount of random variance desired in the ocean is specified as a fraction of significant wave height. Time, cross-sea, and up/down-sea smoothing intervals specify time or distance over which smoothing is performed. The plotting toggle allows the user to selectively plot the two-dimensional ocean. Ocean select specifies which of the previously generated one-dimensional oceans to use as a basis for the quasi-two-dimensional model. The small ocean plot toggle and plot bounds allow the user to plot a smaller region of the full ocean as well as define the smaller region bounds. Small region plots allow for the user to verify the behavior of the ocean from a zoomed in perspective and allow the simulation to stay within memory constraints.

The first of the wave type options produces a uniform wave ocean model. This ocean model is the foundation for all three surface options of the quasi-two-dimensional ocean model. The uniform wave ocean model is initially produced from the one-dimensional ocean by replicating the one-dimensional ocean along the cross-sea (y) dimension. Each replication of the one-dimensional ocean model is spaced as defined by the cross-sea distance resolution. Figure 65 illustrates the replication of the one-dimensional ocean along the y-axis, parallel to the x-axis. We define the origin of the ocean at the center of the y extents and at the initial x location. In MATLAB, we performed this operation by repeating $\boldsymbol{\eta}[\mathrm{x}, \mathrm{t_s}]$ row-wise (first dimension) where $\boldsymbol{\eta}[\mathrm{x}, \mathrm{t_s}]$ was a two dimensional matrix with x in one dimension and $\mathrm{t_s}$ in the other dimension. This process is described by:

$$\boldsymbol{\eta}_{\mathbf{2D}}[\mathrm{x}, \mathrm{y}, \mathrm{t_s}] = \begin{bmatrix} \boldsymbol{\eta}[\mathrm{x}, \mathrm{t_s}] \\ \boldsymbol{\eta}[\mathrm{x}, \mathrm{t_s}] \\ \vdots \\ \boldsymbol{\eta}[\mathrm{x}, \mathrm{t_s}] \end{bmatrix} \tag{3 - 1}$$

where the number of rows is equal to the number of elements in the discretized cross sea distance vector.

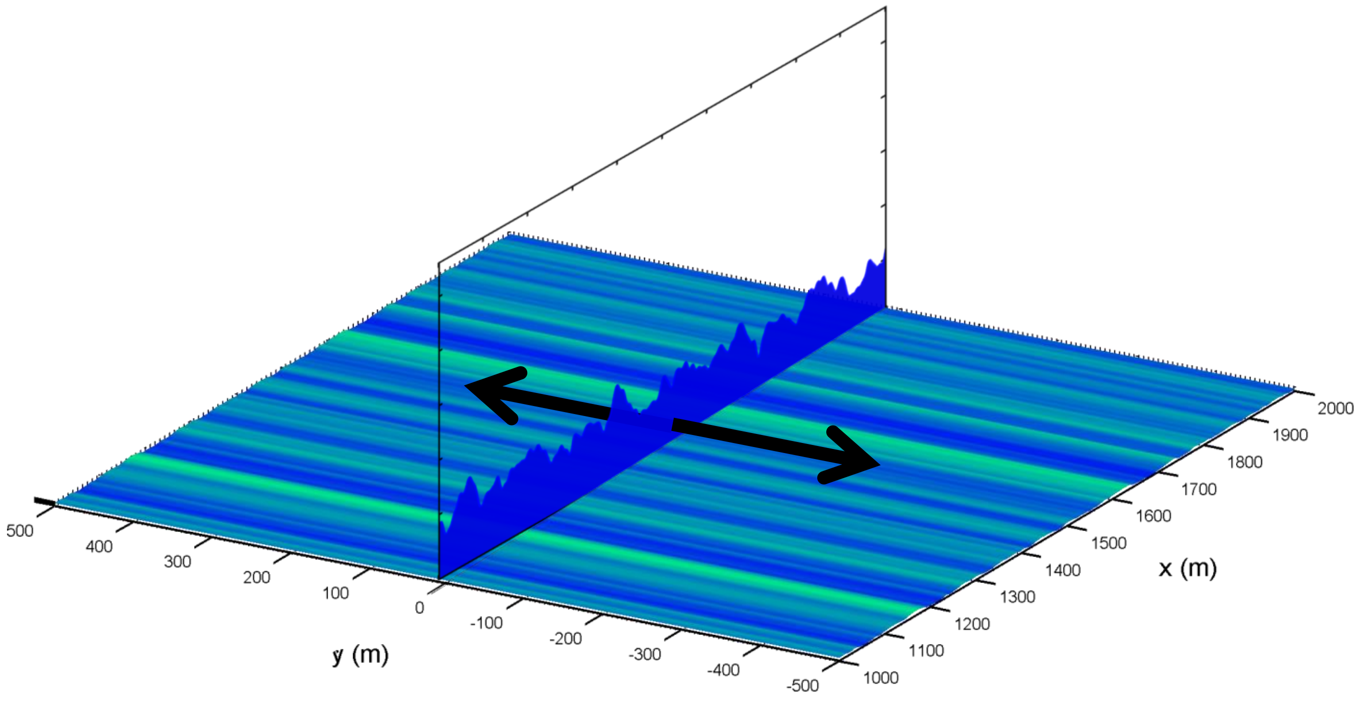


Figure 65. Representation of replication of one-dimensional ocean surface in the cross-sea direction. The black arrows show the directions of replication.

The result of such replication causes the ocean surface to have uniformly propagating waves in the up-sea and down-sea directions without variation of ocean height in the cross-sea direction. The radar is located at some height at the origin. Figure 66 shows a 100 square meter close up of a uniform wave two-dimensional ocean model.

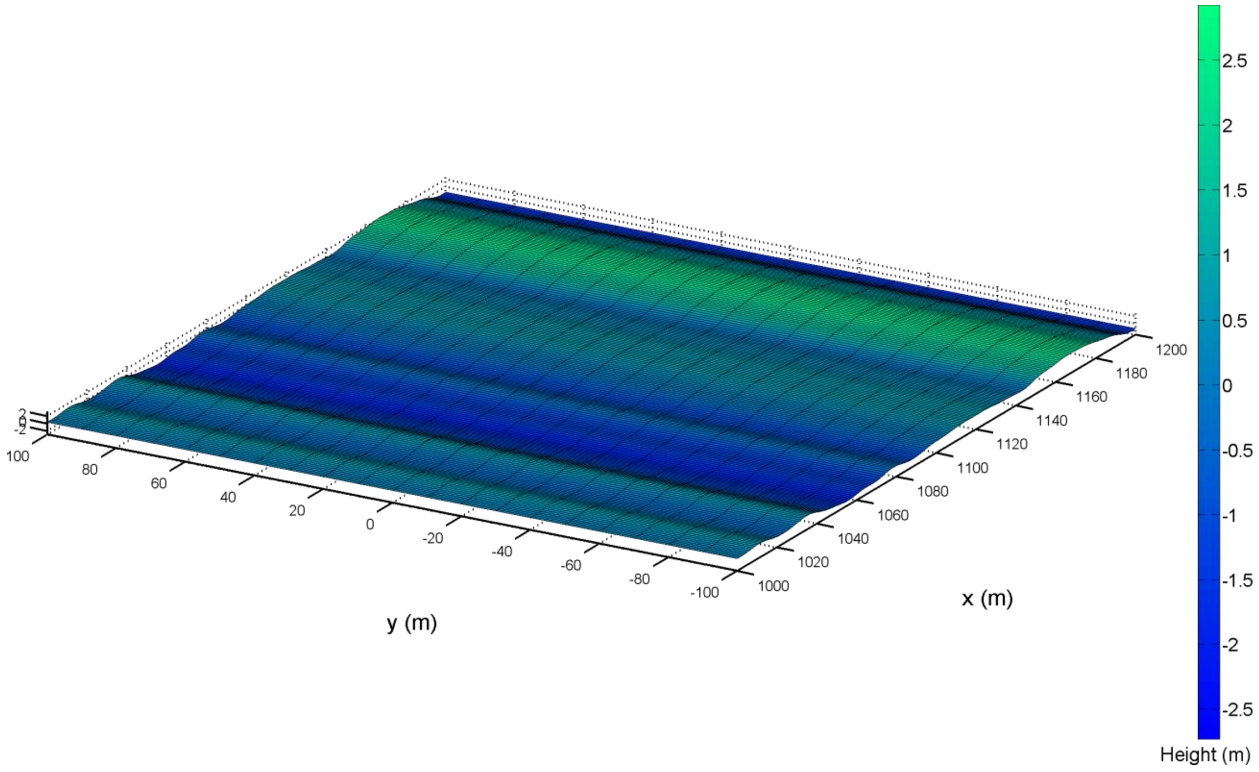


Figure 66. Plot of 200-square meter region of interest (ROI) plot of a uniform wave quasi-two-dimensional ocean surface. Parameters include: $x\ resolution = 1\ m, y\ resolution = 10\ m, wind\ speed = 13\ knots$.

The uniform wave quasi-two-dimensional ocean served as the foundation for the second of the wave type options, achieved by adding independent random height variance. Adding variance to the $\boldsymbol{\eta}_{\mathbf{2D}}$ ocean model was accomplished by adding an independent random Gaussian number to the height of each point of the ocean in time. An equation that represents a quasi-two-dimensional ocean model with variance is characterized by the following equations:

$$\boldsymbol{\eta}_{\mathbf{2D_V}}[\mathrm{x}, \mathrm{y}, \mathrm{t_s}] = \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \quad for\ all\ x, y, and\ t_s \tag{3 - 2}$$

$$\boldsymbol{\mu} = \boldsymbol{\eta}_{\mathbf{2D}}[\mathrm{x}, \mathrm{y}, \mathrm{t_s}] \tag{3 - 3}$$

$$\boldsymbol{\sigma} = p \cdot \boldsymbol{H_s} \tag{3 - 4}$$

where $N(\mu, \sigma^2)$ represents a Gaussian distribution with mean μ and variance σ^2. In Equation (3 - 2), the mean of the distribution is given by the two-dimensional ocean heights in time and the standard deviation is given by the term $p \cdot H_s$, a fraction of the significant wave height. Notice that the mean of the Gaussian variable is a random process. To add variance in MATLAB, we generated a three dimensional random Gaussian matrix with the same dimensions as $\boldsymbol{\eta}_{\mathbf{2D}}$, then performed an element-by-element addition. To produce the Gaussian variance matrix, we first generated a standard normal distribution matrix, and then scaled this matrix by the standard deviation. Figure 67 shows an example of a uniform wave quasi-two-dimensional ocean model with variance.

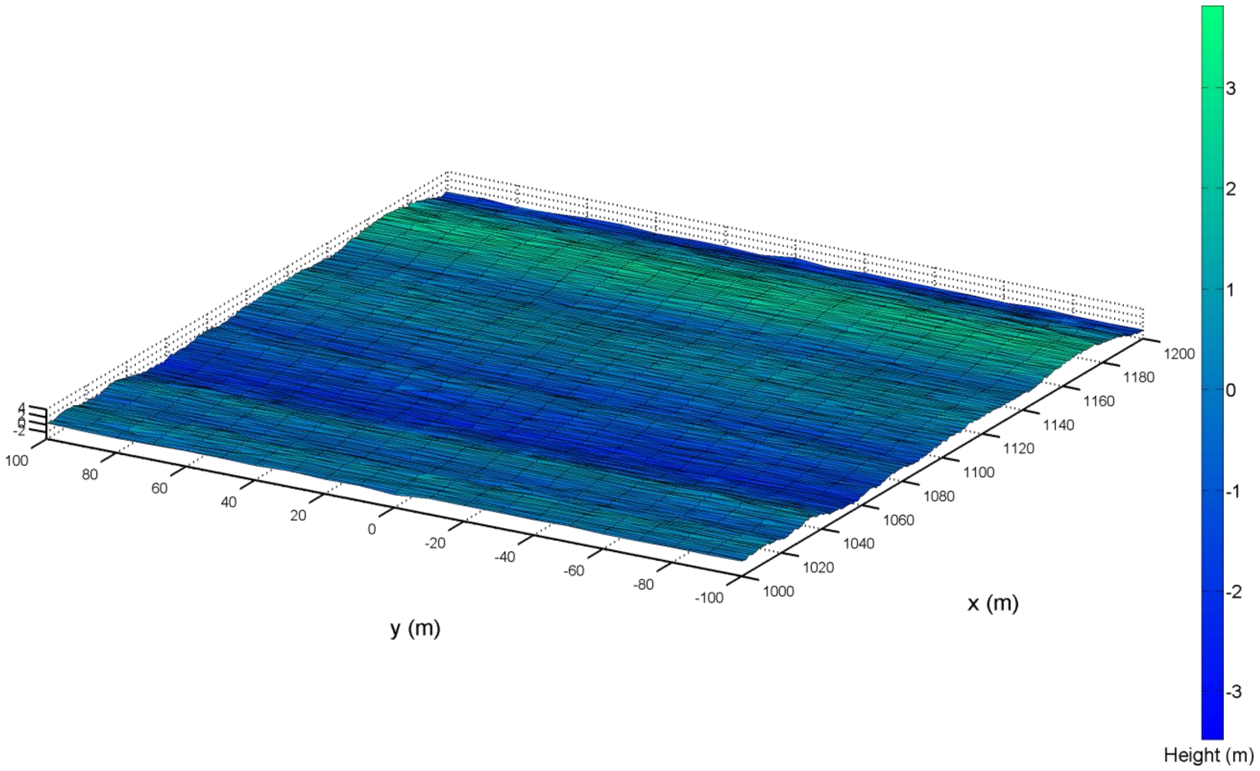


Figure 67. Plot of 200-square meter ROI of a uniform wave quasi-two-dimensional ocean surface with variance. Parameters include: $x\ resolution = 1\ m, y\ resolution = 10\ m, wind\ speed = 13\ knots, \sigma = H_s/15$.

The third and last of the wave type options produces a uniform wave two-dimensional ocean model with variance and smoothing. Note that we do not use $\boldsymbol{\eta_{2D_V}}$ as the basis for the smoothed ocean. Instead, we generate an interpolated random Gaussian matrix and sum it with $\boldsymbol{\eta_{2D}}$. The interpolation inherent in the Gaussian matrix produces the effect of smoothing. The inputs up/down-sea, cross-sea, and time smoothing intervals define the dimensions of a three-dimensional random Gaussian matrix, denoted as **RGM**. There is one random height generated for every interval distance or time. Thus, the dimensions of **RGM** are defined by the number of intervals within the ranges of x, y, or t, respectively.

$$\mathbf{RGM} = \mathrm{N}(0, \boldsymbol{\sigma}^2) \tag{3 - 5}$$

$$\boldsymbol{\sigma} = p \cdot \boldsymbol{H_s} \tag{3 - 6}$$

The Gaussian random matrix is then three dimensionally cubic spline interpolated for each x, y, and t_s. The resultant matrix of the interpolation has the same dimensions as $\boldsymbol{\eta}_{\mathbf{2D}}$, and is represented by:

$$\mathbf{RGM_{interp}} = interpolation_{3D,spline}(\boldsymbol{RGM}) \quad for\ all\ points\ in\ x, y, and\ t_s \tag{3 - 7}$$

This interpolated random Gaussian matrix is added to $\boldsymbol{\eta}_{\mathbf{2D}}$ producing:

$$\boldsymbol{\eta}_{\mathbf{2D_{VS}}}[\mathrm{x}, \mathrm{y}, \mathrm{t_s}] = \boldsymbol{\eta}_{\mathbf{2D}}[\mathrm{x}, \mathrm{y}, \mathrm{t_s}] + \mathbf{RGM_{interp}} \tag{3 - 8}$$

Without smoothing, waves typically are very choppy because they change for each x and y position, and at each time instant. With smoothing, the ocean surface does not vary as quickly and appears more realistically. Note that $\boldsymbol{\eta}_{\mathbf{2D_{VS}}}$ has correlated noise, whereas $\boldsymbol{\eta}_{\mathbf{2D_V}}$ has uncorrelated noise. The spline interpolation parameters manipulate the correlation of the noise. Figure 68 shows an example of a uniform wave two-dimensional ocean model with variance and smoothing.

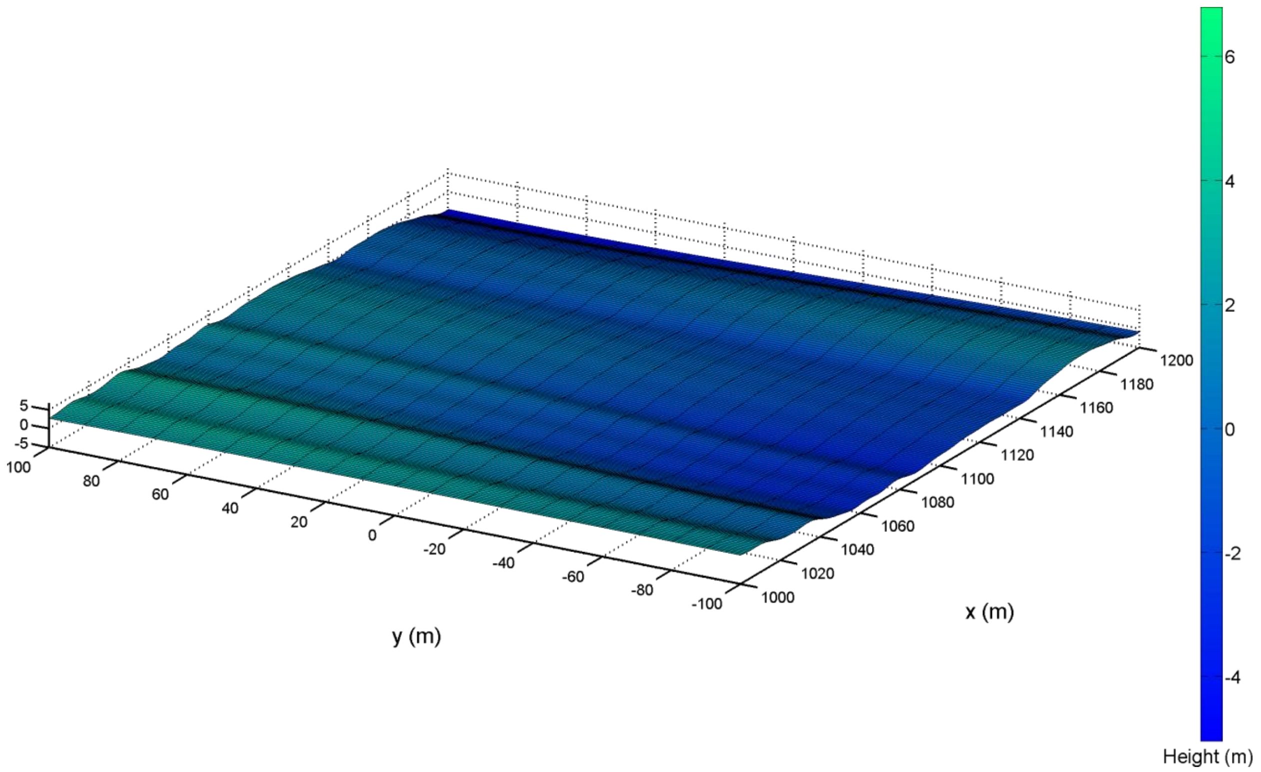


Figure 68. Plot of 200-square meter ROI of a uniform wave quasi-two-dimensional ocean surface with variance and smoothing. Other parameters include: $x\ resolution = 1\ m, y\ resolution = 10\ m, wind\ speed = 13\ knots$, $\sigma = H_s/2$, $x\ smoothing\ interval = 100\ m$, and $y\ smoothing\ interval = 200\ m$.

Figure 69 shows how the smoothing process affects the ocean over 1000 square meters of an ocean with the same input variables as the ocean in Figure 68. Notice that the smoothing affect is more apparent in the latter plot because the y smoothing interval was relatively large relative to the size of the region of interest.

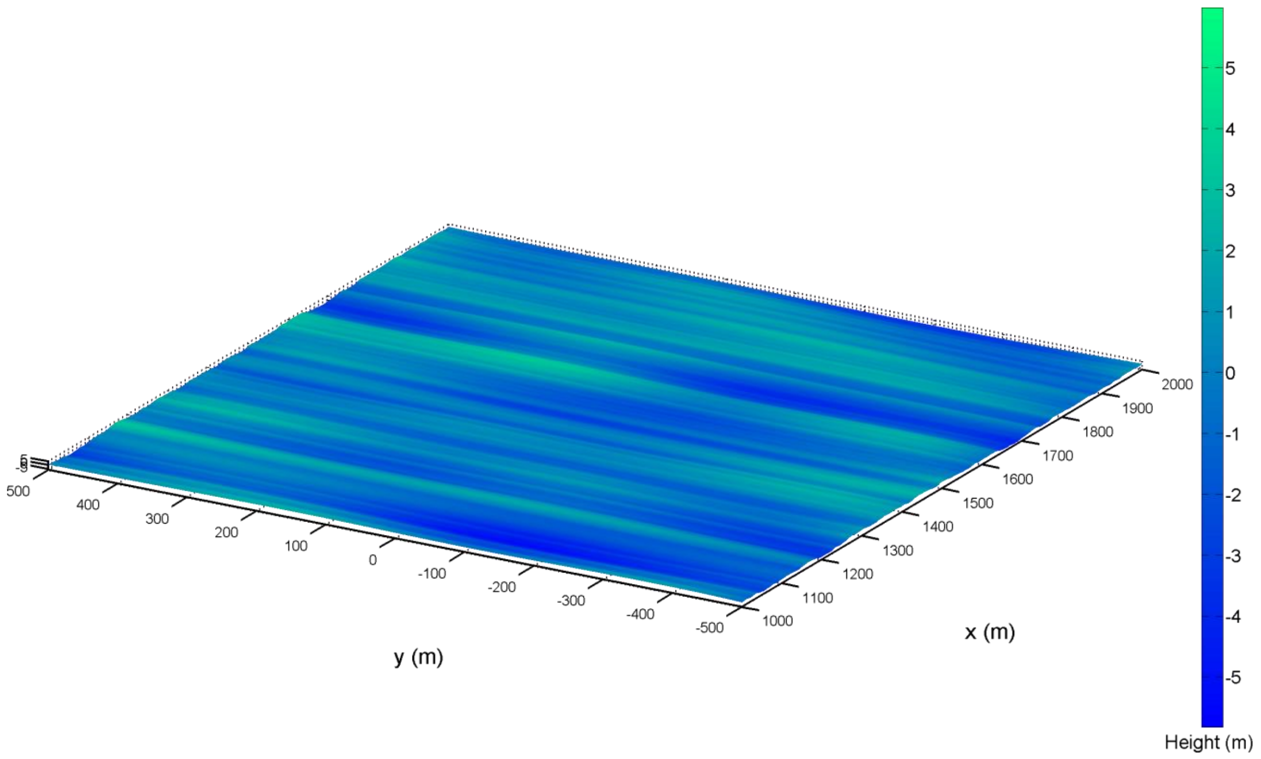


Figure 69. Large ocean plot of the uniform wave quasi-two-dimensional ocean model with variance and smoothing. Other parameters include: $x\ resolution = 1\ m, y\ resolution = 10\ m, wind\ speed = 13\ knots$, $\sigma = H_s/2$, $x\ smoothing\ interval = 100\ m$, and $y\ smoothing\ interval = 200\ m$.

3.2.2 Boat wake model.

The goal of the following methods was to develop an algorithm that generates point targets that represent cusp wave crests based on given boat and radar information. The boat parameters produced the locations of cusp wave crests while the radar information was used to determine the spatial bounds within which cusp wave crests were generated. In order to implement a wake model, we first determined the cusp wave crest geometry, then generated point targets within the radar bounds, and finally, added amplitude decay to the cusp wave crests.

There are three inputs to our algorithm comprised of the:

- Boat vector(s),
- Slow time vector, and
- Radar viewing bounds.

The boat vector(s) contained the initial positions, velocities, and directions for a set of boats. The algorithm is designed to generate the cusp wave crests for each slow time within a given radar dwell time such that the timing information matches the radar timing information. The slow time vector represents this interval of time. The radar viewing bounds are very important to the computational speed of our algorithm. If the target or any of its associated wake is outside of the radar viewing bounds, it is not necessary to compute the radar returns of those point targets because the radar returns will not significantly affect the received energy.

3.2.2.1 Determining cusp wave crest geometry.

The first step in the boat wake generation process was to determine the cusp wave crest geometry. We first present the geometry for a boat traveling toward or away from the radar, then we extend the wake model to simulate a boat traveling in any direction. Using the propagation direction of the cusp wave and the Kelvin arm angle for a boat traveling away from the radar, we constructed a diagram to determine the distance relationship between cusp wave crests, as seen in Figure 70. We refer to the left Kelvin arm as the arm that extends behind the port side of the boat and the right Kelvin arm as the arm that extends behind the starboard side of the boat.

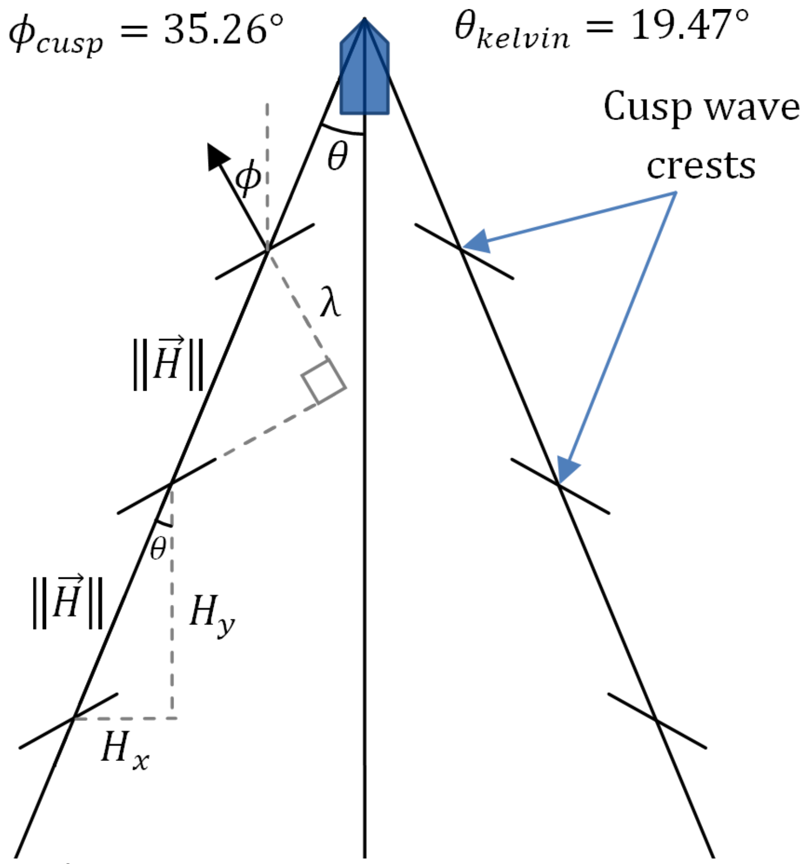


Figure 70. Geometry of cusp wave crest locations. The upside down V represents the Kelvin envelope. The vertex of the Kelvin envelope begins at the bow of the boat, represented by the blue pentagon. The angled lines on the Kelvin envelope represent the localized intersections between the Kelvin envelope and the cusp waves. The locations of these intersections represent the locations of the cusp wave crests of the Kelvin wake.

The interval at which the cusp wave crests repeat on each Kelvin arm depends on the velocity of the boat. For a given constant boat velocity, each cusp wave is separated by a constant offset described by the vector $\vec{H}$ along the Kelvin arm. This distance was determined using the wavelength of the cusp wave from Equation (3 - 2), the propagation angle of the cusp wave, denoted as ϕ_{cusp}, and the angle of the Kelvin arm, represented by θ_{kelvin}:

$$\|\vec{H}\| = \frac{\lambda_{cusp}}{\cos(\theta_{kelvin} + \phi_{cusp})} \tag{3 - 9}$$

It was important that we could find Cartesian components that composed the vector $\vec{H}$ to be able to replicate the cusp wave crest on our ocean efficiently. The distance between each cusp wave was broken into x-y components of $\vec{H}$ that describe the change in distance between the location of one cusp wave crest and the next. These values were used to determine the position of each subsequent cusp wave crest by using multiples of H_x and $\mathrm{H_y}$ from a start position. In Figure 71, point A represents the position of one cusp wave crest location while point B represents the next cusp wave crest location along the Kelvin arm.

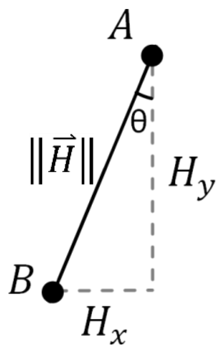


Figure 71. Geometry representing offset between two cusp wave crests. Point A represents the current cusp wave crest location. Point B represents the next cusp wave crest position. H is the distance between the two cusp wave crests.

Trigonometry can be applied to determine the H_x and H_y values. For the simple case where the boat is traveling toward or away from the radar, the components are symmetric and can be described by Equations (3 - 10) and (3 - 11):

$$H_x = \|\vec{H}\| \sin\theta_{kelvin}, \tag{3 - 10}$$

$$H_y = \|\vec{H}\| \cos\theta_{kelvin}. \tag{3 - 11}$$

Next, we established a method for calculating cusp wave crest locations for a boat traveling at any angle. In the previous case, we were able to exploit the fact that the Kelvin envelope was symmetrical about a coordinate axis. Thus, each Kelvin arm had the same angle with respect to the coordinate axis and the same values for $\mathrm{H_x}$ and $\mathrm{H_y}$. However, this symmetry does not apply when the boat is traveling at an arbitrary angle.

To extend this model to simulate the wake of boats traveling in an arbitrary direction, we use the following transformations:

Table 3. Transformations for cusp wave offsets for left and right Kelvin arms.

	Left Kelvin Arm (LKA)	Right Kelvin Arm (RKA)
Θ	$\Phi + (180 - \theta_{kelvin})$	$\Phi + (180 + \theta_{kelvin})$
H_x	$\|\vec{H}\| \cos\Theta_{LKA}$	$\|\vec{H}\| \cos\Theta_{RKA}$
H_y	$\|\vec{H}\| \sin\Theta_{LKA}$	$\|\vec{H}\| \sin\Theta_{RKA}$

where Φ denotes the angle at which the boat is traveling in a polar coordinate system such that the origin is located at the radar. The transformed angle of each Kelvin arm, is denoted as Θ, defined by $\Phi + 180° \pm \theta_{kelvin}$ where positive θ_{kelvin} is the "right" Kelvin arm and vice versa. The left and right Θ can then be substituted into Equations (3 - 10) and (3 - 11) to produce the correct offsets in a Cartesian coordinate space.

3.2.2.2 Generating wake point targets.

The overall goal for the wake model was to generate wake point targets for use in the radar simulation, within the radar interrogation bounds. Because the radar returns generated from targets outside of the radar bounds do not significantly affect the RTI and RDP, we do not include these targets in our model. In turn, our model requires less computation due to the decrease in the number of point targets.

We chose an iterative algorithm to add successive point targets at the cusp wave crest locations. Using this method, we determined four cases to account for:

- Boat and both Kelvin arms within radar bounds,
- Boat outside of radar bounds with both Kelvin arms in radar bounds,
- Boat outside of radar bounds with one Kelvin arm in radar bounds, and
- Boat and both Kelvin arms outside of radar bounds.

The last two cases are difficult to account for with an iterative algorithm because there are no point targets along the Kelvin arm(s) to successively add. For these cases, the Kelvin arm does not intersect the radar interrogation bounds.

In order to simplify these cases for our iterative process, we developed a method to create a subset of points within a region of interest (ROI) that includes the radar bounds. Each Kelvin arm is processed individually within this algorithm. The process involved establishing an ROI around the radar bounds, determining the number of point targets that are possible within this ROI for each Kelvin arm, and then iteratively checking and adding points that are contained within the radar bounds. Another method of finding candidate cusp wave crest locations is by finding the intersections of the Kelvin arms with the radar bounds initially; however, this latter approach is more complex computationally than the ROI method. Additionally, the iterative process does not consider Kelvin arms that do not intersect the ROI.

We used a circular ROI which was a highly effective shape because it was generalized to both the phased array and non-scanning radar cases. Figure 72 depicts how our algorithm accounts for each Kelvin arm. Observe Boat A which is located outside of the ROI where neither Kelvin arm enters the ROI. Boat A does not contain a subset of cusp wave crests that are within the bounds of the ROI so it was not considered. Boat B has one Kelvin arm that enters the ROI but does not enter the radar bounds. This Kelvin arm has a subset of possible cusp wave crests that are contained in the ROI. The green line in the figure below represents the locus of possible cusp wave crest locations. These locations are iteratively checked to determine if they are within the radar bounds. For this Kelvin arm, the intersecting line within the ROI does not intersect the radar bounds, so no points are saved. The Kelvin arm from Boat B that does not enter the ROI will not be considered. Boat C has one Kelvin arm that enters the ROI. The subset of possible cusp wave crests that exist within the ROI are iteratively checked. In this case, the Kelvin arm intersects both the radar bounds and the ROI and the points along it are saved. The red line represents this intersection.

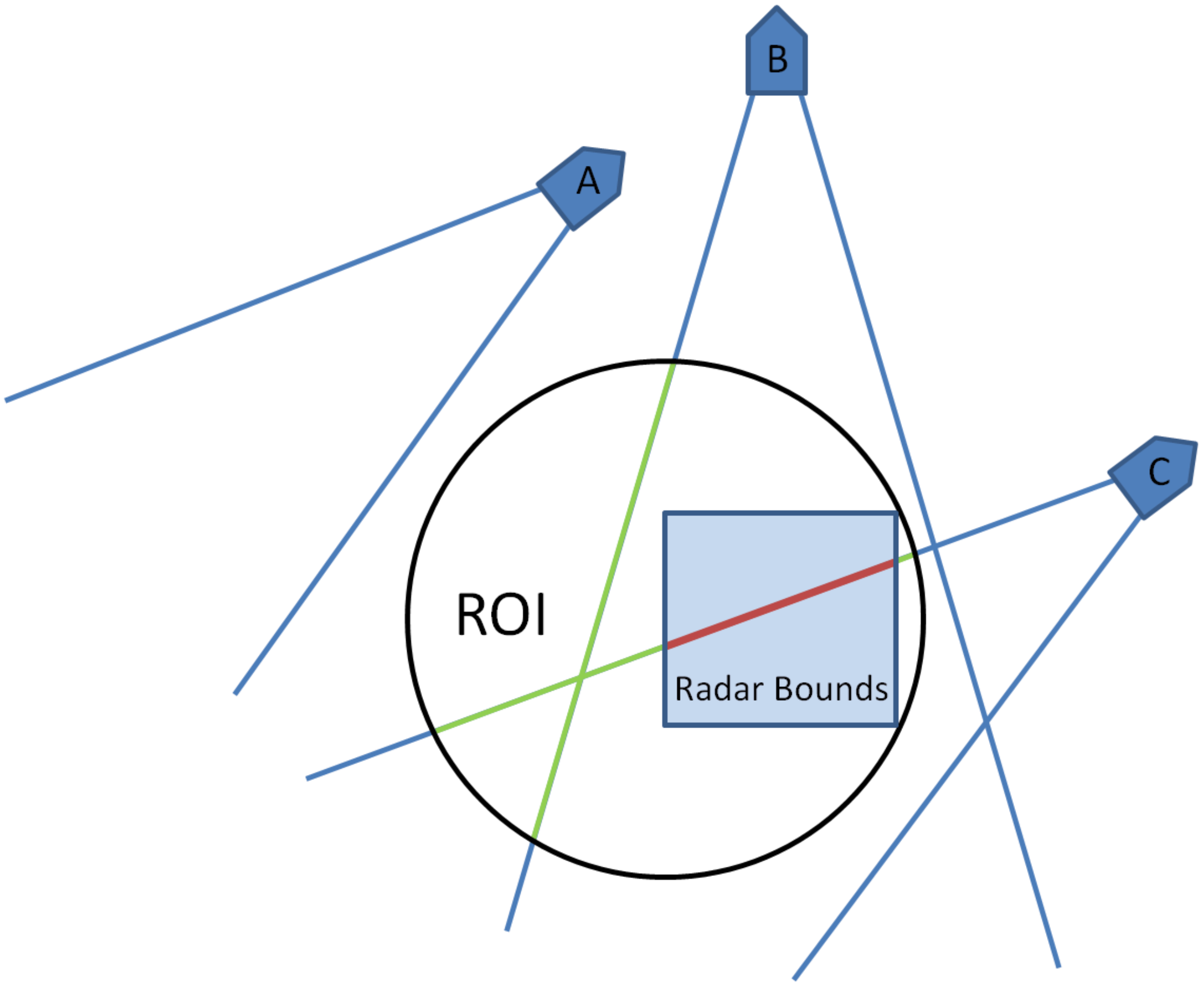


Figure 72. ROI and radar bounds example. Boat A has two Kelvin arms outside the ROI. Boat B has one Kelvin arm inside the ROI and outside the radar bounds. Boat C has one Kelvin arm that enters both the ROI and the radar bounds. Blue lines represent the locus of cusp wave crests that are not iteratively processed. Green lines represent the locus of cusp wave crests that are iteratively processed by not saved (not within radar bounds). Red lines represent the locus of cusp wave crests that are iteratively processed and saved because they are within the radar bounds. The radar is in the center of the ROI

Next we present our algorithm to determine the subset of cusp wave crest point targets for each Kelvin arm. First, we define the position of the bow of the boat as the origin and construct a circle at the radar. This geometry is shown in Equation (3 - 12), where x_0 and y_0 represent the initial x and y positions of the boat and r_{ROI} represents the radius of the circular ROI:

$$r_{ROI}{}^2 = (x + x_0)^2 + (y + y_0)^2. \tag{3 - 12}$$

We substituted Equations (3 - 13) and (3 - 14) into Equation (3 - 12) to convert this function from Cartesian to polar form:

$$x = r\cos\theta \tag{3 - 13}$$

$$y = r \sin\theta \qquad (3 - 14)$$

Next, we simplified the polar form of Equation (3 - 12) to produce the quadratic equation shown in Equation (3 - 15). Applying the quadratic formula to Equation (3 - 15) and solving for r results in equation (3 - 16).

$$r^2 + 2r(y_0 \sin\theta + x_0 \cos\theta) - {r_{ROI}}^2 + {x_0}^2 + {y_0}^2 = 0 \qquad (3 - 15)$$

$$r(\theta) = -y_0 \sin\theta - x_0 \cos\theta \pm\sqrt{(y_0 \sin\theta)^2 + (x_0 \cos\theta)^2 + 2x_0 y_0 \sin\theta\cos\theta + {r_{ROI}}^2 - {x_0}^2 - {y_0}^2} \qquad (3 - 16)$$

The analysis of Equation (3 - 16) allowed us to determine a subset of possible point targets, as explained below. The results of this equation must be interpreted differently depending on if the boat is located inside or outside of the ROI. We will refer to the positive square root function of Equation (3 - 16) as $r_{pos}(\theta)$ and the negative square root function as $r_{neg}(\theta)$.

If the boat is located inside the ROI, $r_{pos}(\theta)$ represents the distance from the boat to an edge of the ROI. Inputting the angle of $\vec{H}$ into $r_{pos}(\theta)$ determines the length of Kelvin arm that exists within the ROI, denoted as L_{ROI}. The floor of the division of L_{ROI} by $\|\vec{H}\|$ results in the number of point targets that exist within the ROI (denoted as N_{ROI}) for a Kelvin arm:

$$N_{ROI} = \left\lfloor \frac{L_{ROI}}{\|\vec{H}\|} \right\rfloor = \left\lfloor \frac{r_{pos}(\angle\vec{H})}{\|\vec{H}\|} \right\rfloor \qquad (3 - 17)$$

Starting from the boat position and iteratively moving down the Kelvin arm until reaching N_{ROI} by adding $\vec{H}$ successively, we can check to see if each point target is within bounds of the radar. The set of valid point targets within the radar bounds can then be input into the radar simulation.

Figure 73 shows the distance from a boat to the edge of the ROI. The black dot represents the position of the boat within the ROI and the x represents the center of the ROI (location of the radar). Both Figure 73 and Figure 74 show red dots which represent the projections from the boat along angles 0, 90, 180, and 270 degrees that were used for basic code verification.

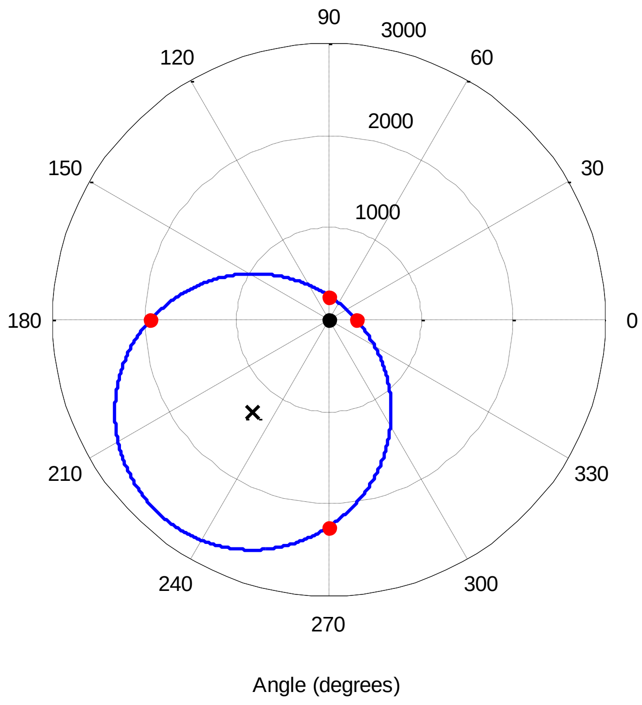


Figure 73. Distance from a boat to the edge of the ROI in polar form. The boat is shown centered at (0,0) whereas the radar is located at the x position.

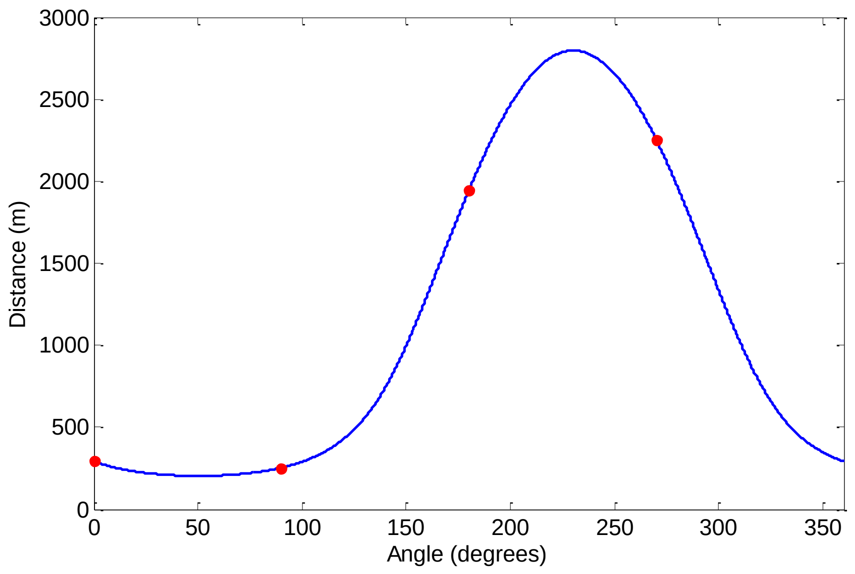


Figure 74. Distance from a boat to the edge of the ROI over 360 degrees. The ROI has a radius of 1500 m with the boat being located inside of the ROI at position (825, 1000). The red points are located at 0, 90, 180, and 270 degrees.

If the boat is located outside of the ROI, there are two conditions associated with the results of $r_{pos}(\theta)$ and $r_{neg}(\theta)$ after inputting the angle of $\vec{H}$. If $r_{pos}(\angle\vec{H})$ and $r_{neg}(\angle\vec{H})$ are imaginary or negative, the Kelvin arm does not enter the ROI. If $r_{pos}(\angle\vec{H})$ and $r_{neg}(\angle\vec{H})$ are positive, the Kelvin arm enters the ROI. In this case, the length of the Kelvin arm that exists in the ROI , denoted as L_{ROI}, is given by:

$$L_{ROI} = r_{pos}(\angle\vec{H}) - r_{neg}(\angle\vec{H}) \tag{3 - 18}$$

The first point target that exists within the ROI occurs at $n_{ROI\,start}$ successive point target positions away from the boat, and is given by:

$$n_{ROI\,start} = \left\lceil \frac{r_{neg}(\angle\vec{H})}{\|\vec{H}\|} \right\rceil \tag{3 - 19}$$

The last point that exists within the ROI occurs at $n_{ROI\,end}$ successive point target positions away from the boat, and is given by:

$$n_{ROI\,end} = \left\lfloor \frac{r_{pos}(\angle\vec{H})}{\|\vec{H}\|} \right\rfloor \tag{3 - 20}$$

The number of point targets that exist in the ROI along this locus is:

$$N_{ROI} = n_{ROI\,end} - n_{ROI\,start} + 1 \tag{3 - 21}$$

A general equation to find the position of a successive point target away from the boat, denoted as P_{PT}, with the successive point target number, denoted as n_{PT}, is defined by:

$$P_{PT} = n_{PT} \cdot \vec{H} \tag{3 - 22}$$

With these equations, we are able to iteratively check to see if each point target is within bounds of the radar. If the point target is within bounds of the radar, it can be later input into the radar simulation.

We consider two scenarios for radar bounds that must be iteratively checked: square (for non-scanning radar, see Section 2.1.1) and circular sector (for phased array radar, see Section 4.1). Using the square radar bounds, we included only cusp wave crest point targets that exist within the bounds of the quasi-two-dimensional ocean. We chose to position the origin of ROI circle at the same location as the origin of the quasi-two-dimensional ocean model. With the position of the ROI and the bounds of the ocean, we define the radius of the ROI to be the length from the origin to the furthest ocean surface position from the origin, given by:

$$r_{ROI} = \sqrt{\max(|y|)^2 + \max(x)^2} \tag{3 - 23}$$

We then must check every cusp wave crest point target within the ROI along each Kelvin arm and add the point targets that fit the requirements:

- $\min(y) \le P_{PT,y} \le \max(y)$

- $\min(x) \leq P_{PT,x} \leq \max(x)$

where $P_{PT,y}$ is the y position and $P_{PT,x}$ is the x position of the cusp wave crest point target.

Using the circular sector bounds, we include only cusp wave crest point targets that exist within the bounds of the phased-array radar viewing angle of $\pm 45°$. Because a circular sector is a portion of a complete circle, we define the r_{ROI} to be:

$$r_{ROI} = d_{alias-free} \tag{3 - 24}$$

where $d_{alias-free}$ is $\sqrt{{r_{alias-free}}^2 - h^2}$. We then must check every cusp wave crest point target within the ROI along each Kelvin arm and add the point targets that fit the requirements:

- $-45° \leq \angle P_{PT} \leq 45°$
- $\|P_{PT}\| \leq d_{alias-free}$

It is not necessary to check for the second requirement because the process of limiting the point targets to within the ROI inherently insists by Equation (3 - 24) that the magnitude of the position is within the alias-free maximum distance.

Figure 75 shows four boats each traveling at various velocities and angles and exactly where there cusp waves would be located using a phased-array radar.

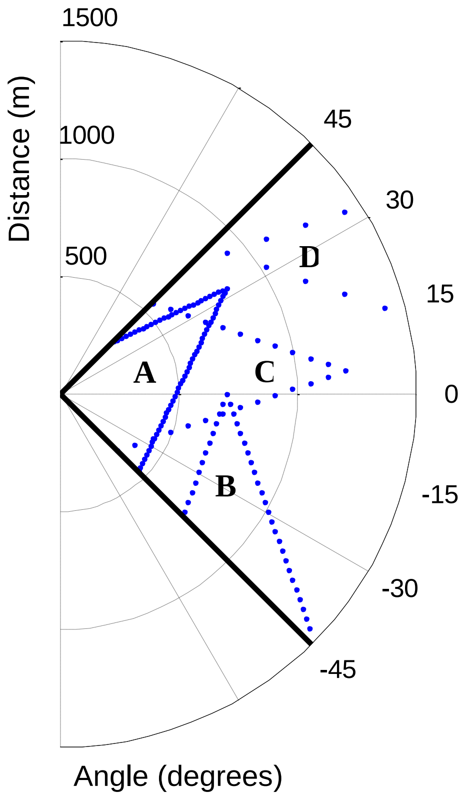


Figure 75. Model of multiple boats and their wakes traveling at various velocities. Boat traveling A) 10 knots at 45 degrees, B) 15 knots at 90 degrees, C) 20 knots at 0 degrees, and D) 30 knots at 180 degrees. The black bold lines represent the scanning bounds for the radar.

The next step that is necessary is preparing for the propagation of these cusp waves for each slow time of the radar. Each cusp wave propagates at the velocity of the boat, so the position of every cusp wave crest also propagates at the same velocity. The standard kinematic equations for position shown in Equations (3 - 25) and (3 - 26) can be used to determine the position at each subsequent slow time:

$$x = x_0 + v_x t_s \tag{3 - 25}$$

$$y = y_0 + v_y t_s \tag{3 - 26}$$

Finally, we added decay to the cusp wave crests to make them more realistic for our model. According to Landau and Lifshitz, dissipation of ocean waves follows the equations (as cited in Chryssostomidis & Liu, 2011):

$$amplitude\ decay = e^{-\gamma t} \tag{3 - 27}$$

$$\gamma = 2vk^2 = \frac{2v\omega^4}{g^2} \tag{3 - 28}$$

The above equation was used to add decay to the cusp wave crest amplitudes in our boat wake model. Because the cusp waves travel at the group velocity, we substituted Equation (3 - 29) into Equations (3 - 27) and (3 - 28). Simplifying the result led to Equation (3 - 30).

$$\omega = \frac{2\pi V_{cusp}}{\lambda_{cusp}} \tag{3 - 29}$$

$$amplitude\ decay = e^{-\left(\frac{32\pi^4 V_{cusp}{}^5}{\lambda_{cusp}{}^4 g^2}\right)t} \tag{3 - 30}$$

An example of the decay of the boat wake coefficients in time is shown in Figure 76.

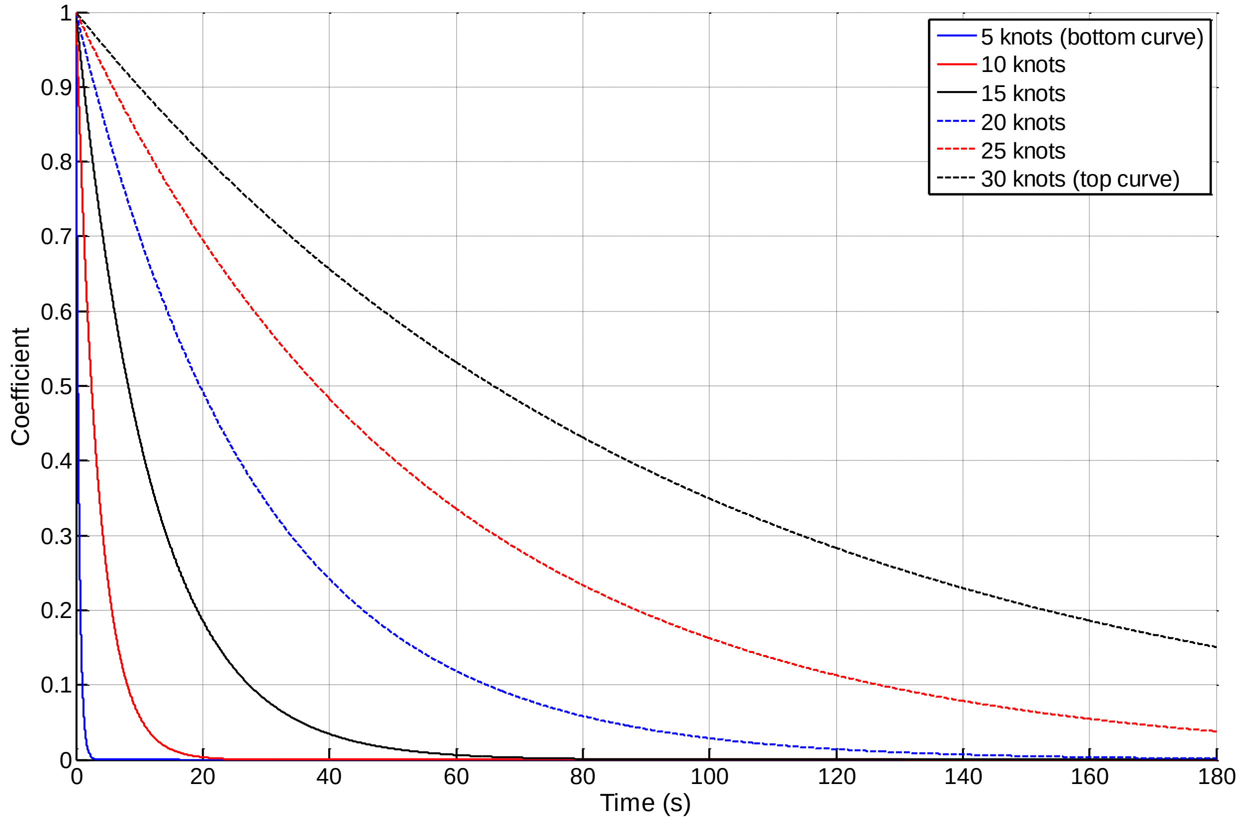


Figure 76. Amplitudes of dampening boat wake coefficients for various velocities in time.

The cusp wave crest amplitude coefficient is a scale factor between 0 and 1. Cusp wave crests are discrete points along these curves. Each cusp wave crest was considered an indexed time period away from the boat described by equation:

$$t_i = \frac{\|\vec{H}\| cos(\theta_{kelvin}) \cdot i}{V_{boat}} \tag{3 - 31}$$

where i is the index number of the cusp wave crest behind the boat. Substituting Equation (3 - 31) into Equation (3 - 30) yields:

$$indexed\ amplitude\ decay = e^{-\left(\frac{32\pi^4 V_{cusp}{}^5}{\lambda_{cusp}{}^4 g^2}\right)\left(\frac{\|\vec{H}\| \cos(\theta_{kelvin})\cdot i}{V_{boat}}\right)} \qquad (3 - 32)$$

Each cusp wave crest was given an associated index and the resulting coefficients of Equation (3 - 32) were applied to the amplitudes of the cusp wave crests in time.

The very act of modeling Kelvin wake cusp wave crests adds substantial computation because each one needs to have its radar returns generated. To reduce the number of cusp wave crests, and thus computation, we limit the generation of cusp wave crests to those which have coefficients above 0.005.

Figure 77 shows the results of these methods using non-scanning radar with a boat traveling toward the radar at 20 knots. The left plot is a RDP of the boat and wake and the right plot is the actual positions of the point targets. The bow of the boat is located at the point of the V shaped pattern. A time series of this example is located in the accompanying attachment to this report. Notice that the cusp wave crests across from one another on opposite Kelvin arms are not resolved in the RDP because they occupy the same range and velocity bins.

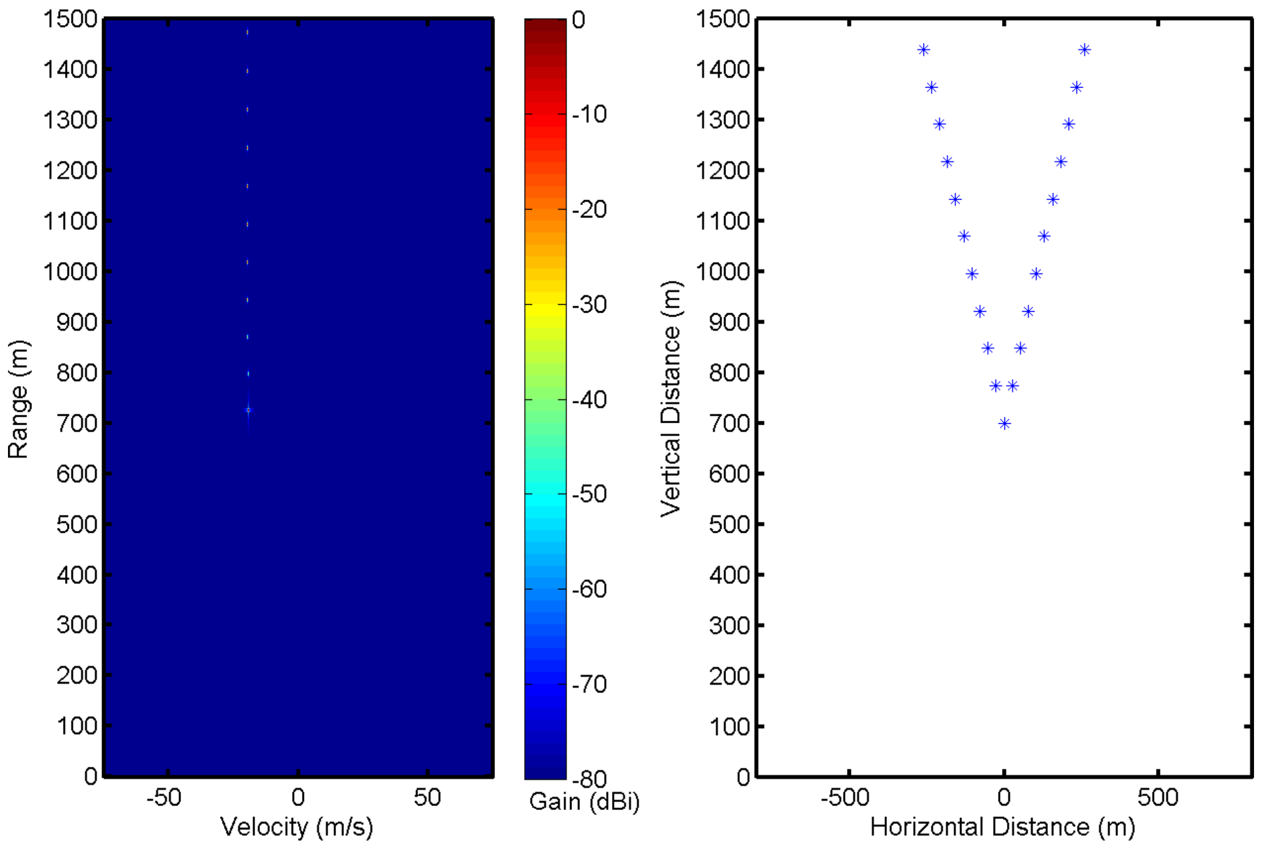


Figure 77. Range Doppler profile for a boat with cusp waves traveling at 20 knots toward the radar. Other parameters include: $f_0 = 10\ GHz, B = 100\ MHz, r_{alias-free} = 1500\ m, v_{alias-free} = 70\ m/s, d_0 = 700\ m,$ $V_{boat} = 10.288\ m/s\ (20\ knots)$.

3.2.3 Integration.

To integrate the quasi-two-dimensional ocean, non-scanning radar, targets, and their wakes, the same computing architecture from the non-scanning, one-dimensional ocean radar scattering simulation was used as a base. This section explains the important differences and modifications that were essential to implement for MATLAB integration. The chosen architecture allowed easy addition of more complicated processes and also allowed us to leverage previous functionality. Figure 78 shows the modified program architectural flow, with green boxes that indicate major new inputs, processing steps, and outputs, and with blue, dashed boxes that indicate modified processes.

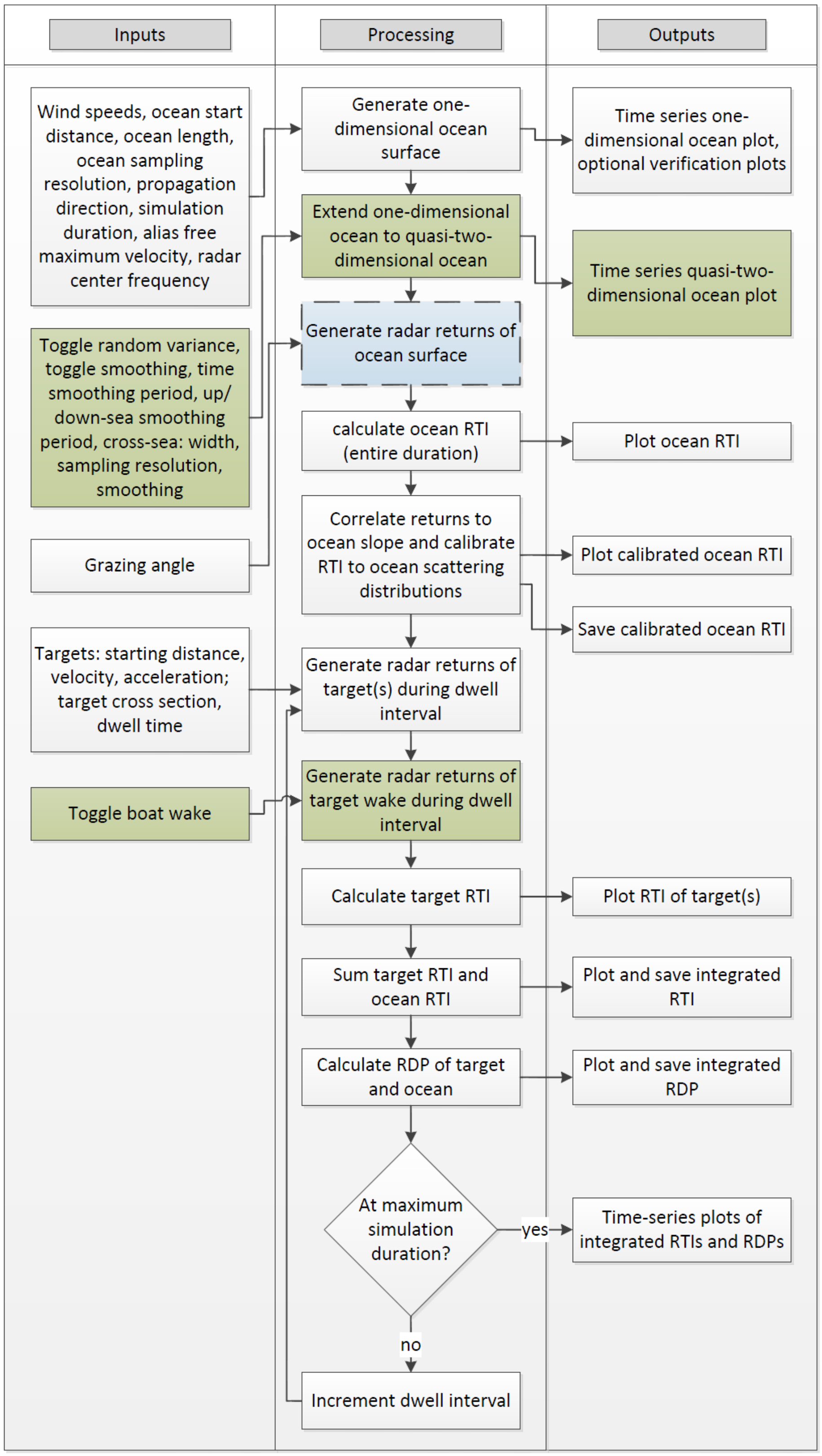

Inputs
Processing
Outputs
Wind speeds, ocean start distance, ocean length, ocean sampling resolution, propagation direction, simulation duration, alias free maximum velocity, radar center frequency
Toggle random variance, toggle smoothing, time smoothing period, up/down-sea smoothing period, cross-sea: width, sampling resolution, smoothing
Grazing angle
Targets: starting distance, velocity, acceleration; target cross section, dwell time
Toggle boat wake
Generate one-dimensional ocean surface
Extend one-dimensional ocean to quasi-two-dimensional ocean
Generate radar returns of ocean surface
calculate ocean RTI (entire duration)
Correlate returns to ocean slope and calibrate RTI to ocean scattering distributions
Generate radar returns of target(s) during dwell interval
Generate radar returns of target wake during dwell interval
Calculate target RTI
Sum target RTI and ocean RTI
Calculate RDP of target and ocean
At maximum simulation duration?
yes
no
Increment dwell interval
Time series one-dimensional ocean plot, optional verification plots
Time series quasi-two-dimensional ocean plot
Plot ocean RTI
Plot calibrated ocean RTI
Save calibrated ocean RTI
Plot RTI of target(s)
Plot and save integrated RTI
Plot and save integrated RDP
Time-series plots of integrated RTIs and RDPs

Figure 78. Overall work flow diagram for the non-scanning quasi-two-dimensional ocean model. The left pane shows user inputs, the middle pane shows algorithmic flow, and the right pane shows simulated outputs. The green boxes represent new and the blue, dashed boxes represent modified processes in comparison to Figure 44.

The first major change in the architecture was the addition of the quasi-two-dimensional ocean model. As discussed in Section 3.2.1, this quasi-two-dimensional ocean is an extrapolation of the one-dimensional ocean, so it takes the one-dimensional ocean as a direct input. Of course, this extrapolation has process-specific inputs that are required and allows us to plot the ocean surface in three dimensions (x, y, height). The primary output of this second process block is a MATLAB matrix cube which contains the quasi-two-dimensional ocean heights for x, y, and t.

The third processing block, "Generate radar returns of the ocean surface" was originally developed to accept a one-dimensional ocean as input (a two-dimensional MATLAB matrix representing ocean heights for x and t). However, it had to be modified to accept a two-dimensional ocean. This modification is accomplished rather simply by modifying Equation (2 - 64) to incorporate cross-sea distance, y:

$$r_\tau[t_s] = \sqrt{(h - \boldsymbol{\eta}[x, t_s])^2 + x^2 + y^2} \qquad (3 - 33)$$

In addition, the one-dimensional slope scaling technique that was used required modification. Instead of using the slope of the ocean, we used the dot product of the normal unit vector and the radar line of sight unit vector at each position of the ocean. Figure 79 shows normal unit vectors overlaid on a sample ocean surface. Then, we took the product of the result of the dot product and the radar return. That is, for every point on the ocean surface, we performed the following:

$$V[\omega, t_s] = e^{-2jk \cdot R_s} \sum_{\tau=1}^{\mathrm{N_T}} \left(\vec{n}(x, y, t_s) \cdot \overrightarrow{u_{los}}(x, y, t_s)\right) e^{-2jkr_\tau[t_s]} \qquad (3 - 34)$$

Where $\vec{n}(x, y, t_s)$ is the normal unit vector and $\overrightarrow{u_{los}}(x, y, t_s)$ is the radar line of sight unit vector at coordinate position (x, y) and slow time t_s.

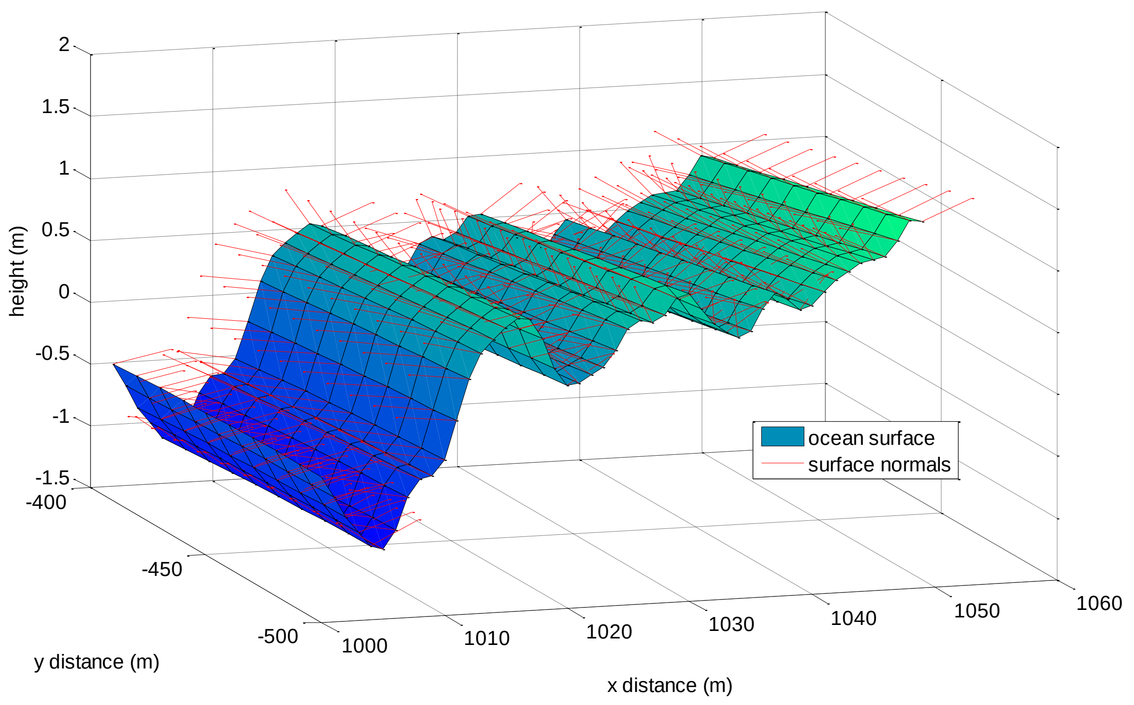


Figure 79. Surface normals drawn on a $50x100\ m$ grid of ocean surface. Parameters include: $wind\ speed = 13\ knots, x\ resolution = 1\ m, y\ resolution = 10\ m$, variance and smoothing toggled on, $x\ smoothing\ interval = 100\ m, y\ smoothing\ interval = 200\ m$

The last major addition to the architecture was to include boat wake. In our architecture, we generate boat wake after the targets are generated because the boat wake is dependent on the position and velocity of the targets. We consider boat wake as additional point targets. After that, the subsequent processing blocks do not need modification because they were already capable of multi-target range return generation. In fact, because there is a summation over targets when generating range returns, there is absolutely no difference in output size with or without toggled boat wake. Note that the boat itself is still modeled, as discussed in Section 2.2.3.

It is important to emphasize that the same calibration routine is used for the one-dimensional and two-dimensional ocean radar returns. Using the same calibration routine is valid because the computed radar scattering distribution is similarly shaped after dot product scaling. In addition, the calibration routine accepts radar returns as input, and the radar returns are the same size in the one- and two-dimensional ocean cases (adding targets in the cross-sea direction adds more targets, but radar returns are summed through targets).

3.3 Results and Discussion

In the background and methods of this chapter, we have discussed the development of a non-scanning, quasi-two-dimensional ocean scattering simulation. By leveraging the architecture of the non-scanning, one-dimensional ocean scattering simulation, we were successful at extending the one-dimensional model to a two-dimensional model. The parallelization and graphical user interface make this simulation practical and easy to use (discussed in Sections 5.3 and 6.2). Because the one-dimensional algorithms are used in part to generate the two-dimensional model, the verifications discussed in Section 2.3 also apply to the two-dimensional model. In this section, we focus on the verification and results of the modifications described in Section 3.2.3.

Similar to the verification of range and Doppler processing in the one-dimensional case, it was simple to verify such processing in the two-dimensional simulation. In particular, we were able to easily verify the modification described by Equation (3 - 34) and the geometry of the Kelvin wake by comparing the locations of target power in RTI plots and RDPs to expected range bins. We show an example of RTI plots and RDPs with boat and wake later in this section.

After we verified that the dot product of the ocean surface normal unit vectors and the radar line of sight unit vectors corresponded to the slope of the ocean, we verified that dot product scaling had beneficial effects on the ocean scattering distribution. Recall that computed ocean scattering distribution is mode shifted and dilated to fit the empirical radar scattering characteristics of the ocean. The operation of dot product scaling mode shifts and dilates the computed scattering distribution such that the computed distribution is a better fit to the empirical distribution (prior to log-normal calibration). A comparison between the un-scaled ocean scattering distribution and the dot product scaled distribution of the same quasi-two-dimensional ocean is shown in Figure 80. Notice that the un-scaled distribution spans approximately $35\,dB$ while the scaled distribution spans approximately $40\,dB$. Because the empirical distribution spans approximately $60\,dB$, the dot product scaling increased the width of the distribution toward the expected width. Also notice how the mode of the un-scaled distribution is approximately $10\,dB$ and the mode of the scaled distribution is approximately $-10\,dB$. The mode of the empirical distribution is approximately $-21\,dB$, so the dot product scale shifted the mode toward the expected mode. Finally, notice that the dot product scaled distribution is less skewed, and thus, more Gaussian in shape. Although we only show one comparison for dot product scaling, we found that it is effective for all wind speeds and both HH and VV polarization.

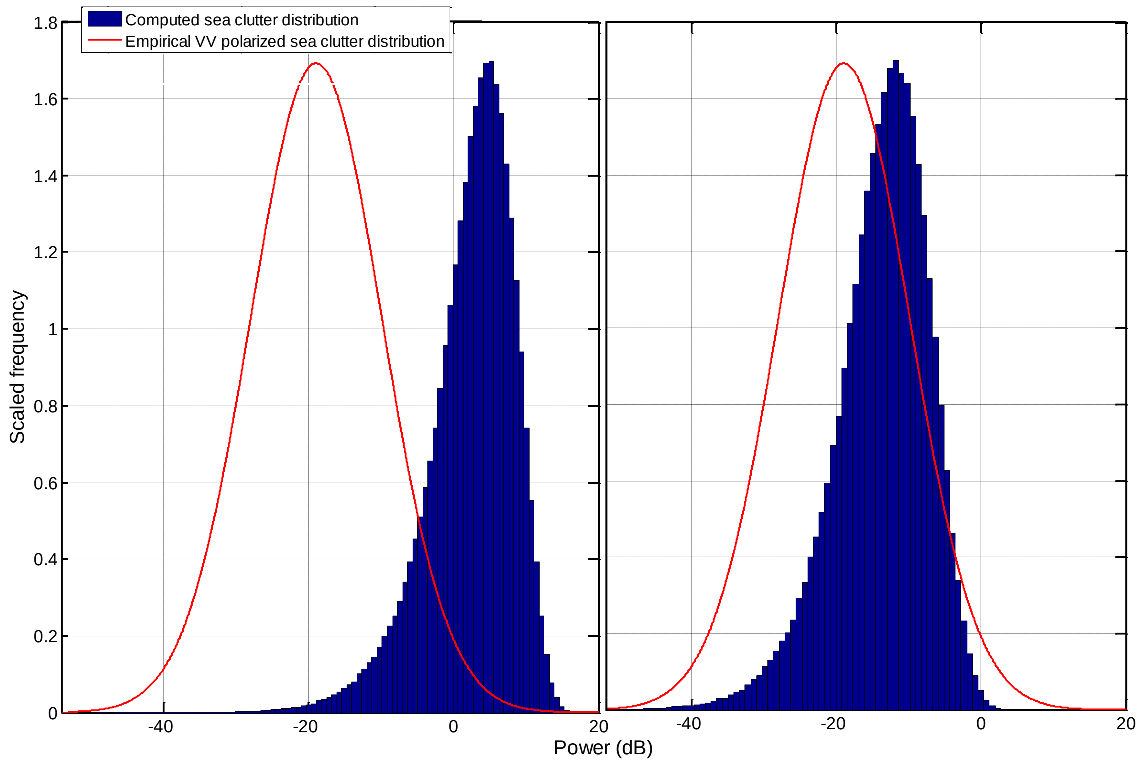


Figure 80. Calculated ocean scattering distributions for uniform quasi-two-dimensional ocean surface with no surface normal scaling (left) and with surface normal scaling (right). Parameters include: $f_0 = 10\ GHz$, $B = 100\ MHz, grazing\ angle = 5°, d_{alias-free} = 1000\ m,\ prf = 40\ Hz$, duration $= 5\ s$, wind speed $=$ $13\ knots,\ R_s = 1000\ m$, downsea propagation, VV polarization, cross sea range $= 200\ m$, cross sea resolution $= 10\ m$, Gaussian variance added to ocean, variance $= 1/3\ H_s$, cross sea smoothing intervals $=\ 5$, sea smoothing intervals $= 10$, time smoothing period $= 3\ s$, no windowing.

Figure 81 shows the RTI plots of the un-scaled and scaled distributions. Note how it is easier to distinguish the time smoothing property in the scaled case. We notice a significant difference when comparing RTI plots of the two dimensional surface with RTI plots of a one-dimensional surface. Specifically, it is more difficult to distinguish propagation, or group, velocity of the waves as a function of time. That is, there is no clear slope as was evident in one-dimensional RTI plots such as Figure 57. We believe that the primary reason for this difference is that the same range bin corresponds to many different waves, as depicted in Figure 83. In fact, we hypothesize that this effect also causes the nulls and high power regions in uniform quasi-two-dimensional oceans with low cross distance range, as highlighted in Figure 82 (note the relatively low cross range of $200\ m$). In this case, we think the nulls and high power regions occur where the same range bin represents exactly the wavelength of the dominant frequency component of the ocean at that range. The cross distance range governs the distance (toward or away from the radar) range of the range bins.

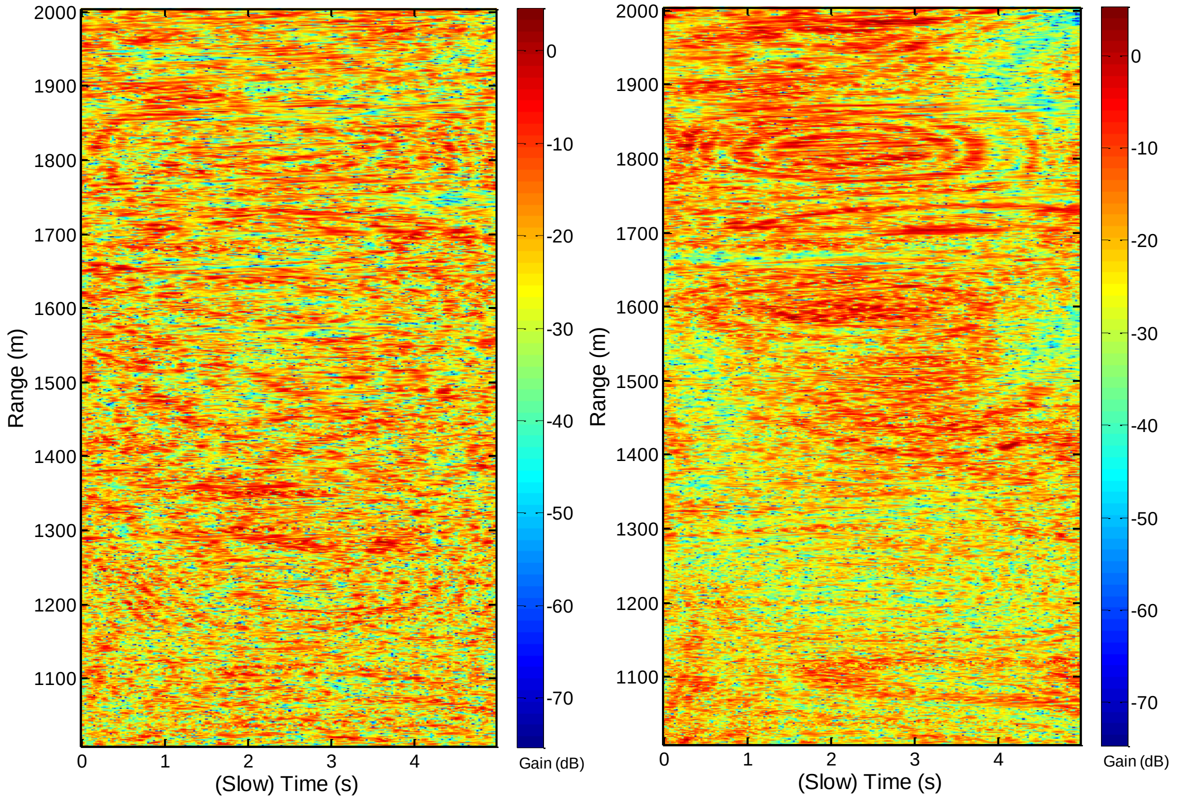


Figure 81. Comparison between RTI plot of a uniform quasi-two-dimensional ocean with no scaling (left) and dot product scaling (right). The same ocean and radar parameters as used in Figure 80 were used.

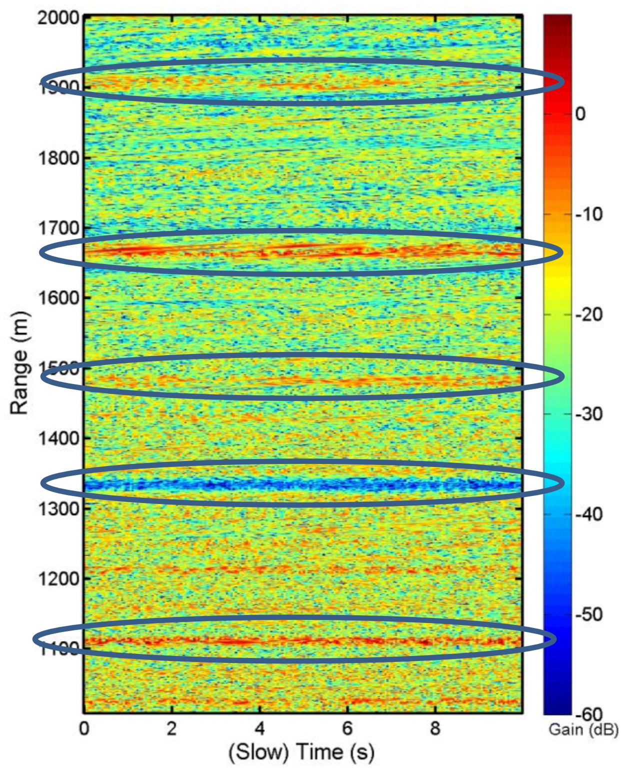


Figure 82. RTI of uniform quasi-two-dimensional ocean. Nulls and high power peaks are highlighted by the solid blue ellipses. Parameters include: : $f_0 = 10\ GHz,\ \ B = 100\ MHz, grazing\ angle = 5°, d_{alias-free} = 1000\ m, prf = 20\ Hz, T_{dwell} = 10\ s$, wind speed $= 13\ knots,\ \ R_s = 1000\ m$, dowqnsea propagation, VV polarization, cross sea range $= 200\ m$, cross sea resolution $= 10\ m$, no Gaussian variance or smoothing, no windowing.

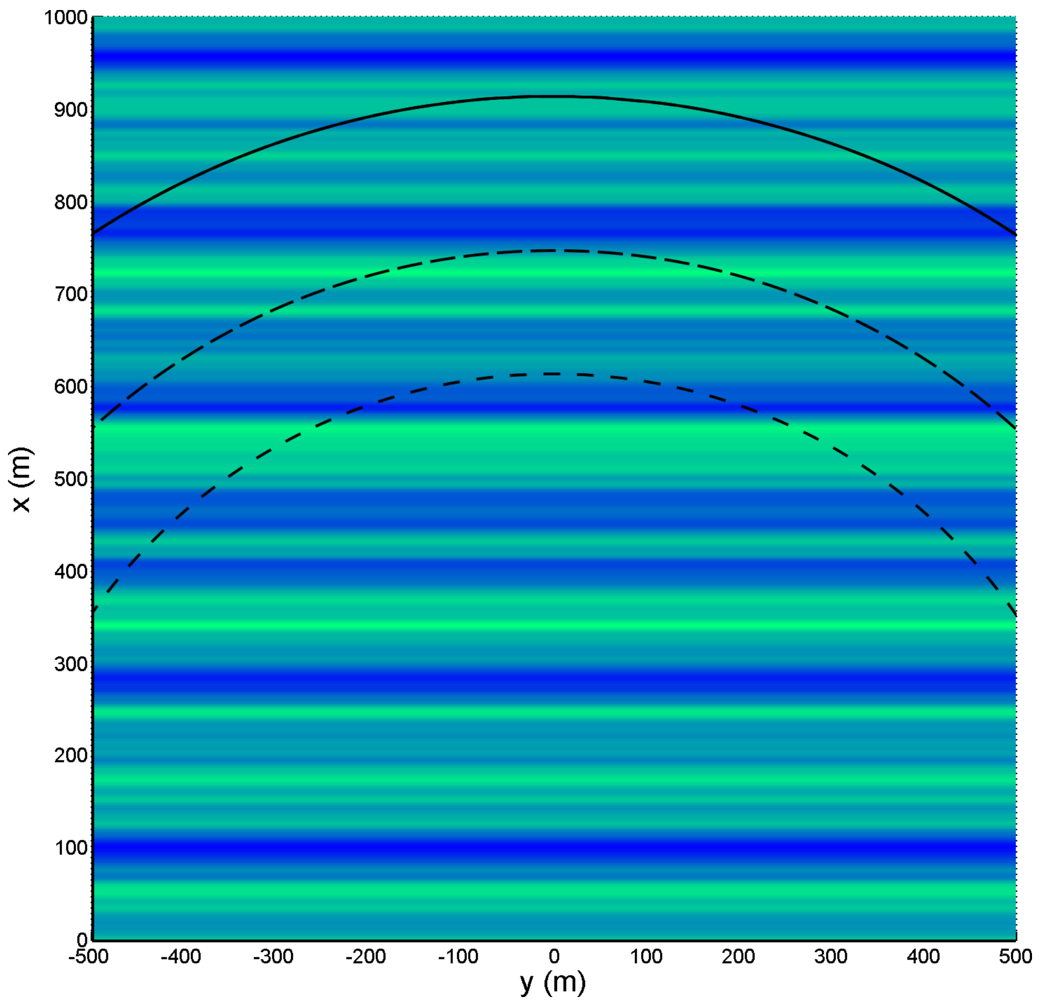


Figure 83. Overhead view of uniform quasi-two-dimensional ocean. Each point along each circular line would fall into the same range bin in an RTI and RDP. Here, the lines span many waves. At further distances (in x), the curvature would not be so great, and the locus of points which correspond to the same range bin may span exactly one wave and produce effects such as those in Figure 82.

Recall that three quasi-two-dimensional ocean models were developed: uniform, uniform with variance, and uniform with variance and smoothing. In the figures below, we show the RTI plot and RDP for each of these models. The same one-dimensional ocean surface was used as the base for each generation. A boat and its resulting wake were modeled on the surface of these ocean surfaces. The kinematics of the modeled boat are shown in Figure 84.

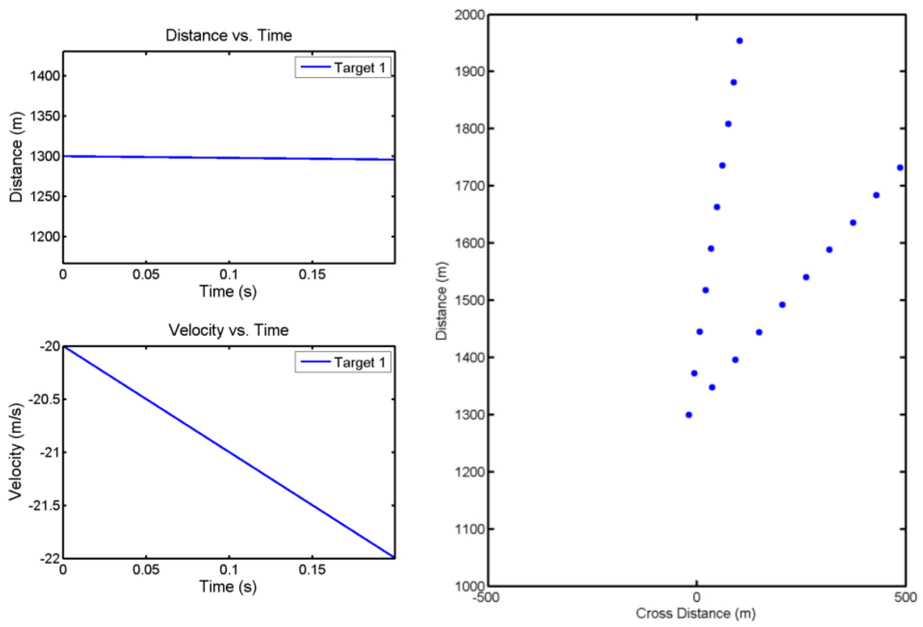


Figure 84. (Left) boat kinematics and (right) actual position of boat and its associated wake for the examples of Figure 85, Figure 86, and Figure 87. The boat started $1300\ m$ away from the radar with a velocity of $10\ m/s$ toward the radar at an angle of 30 degrees relative to the radar. The radar cross section of the boat was set to 10 times the mean radar cross section of the ocean. The radar cross section of the first cusp wave crests was set to $1/5$ the cross section of the boat.

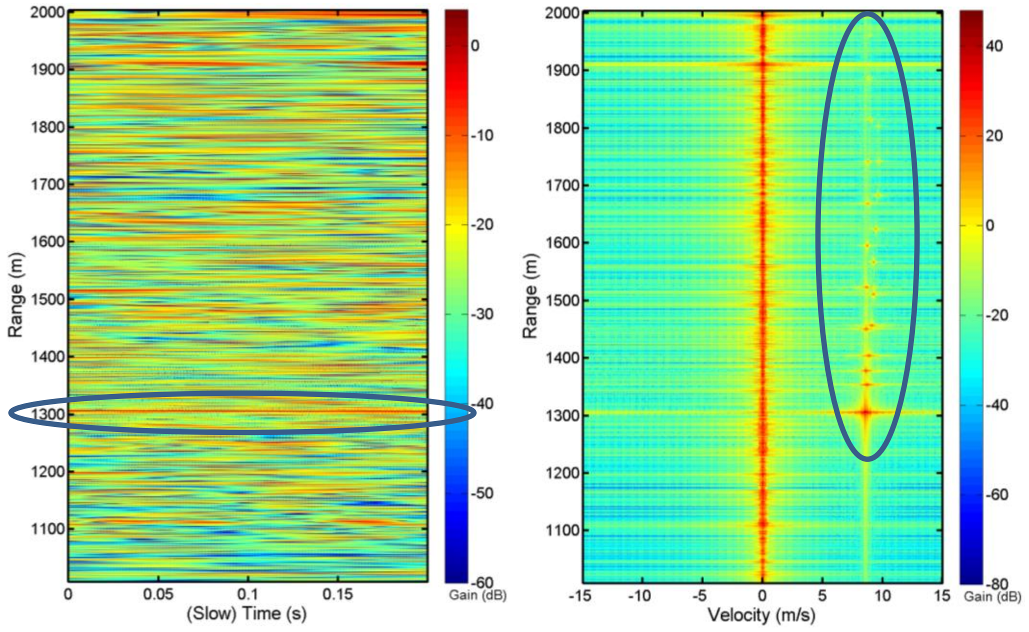


Figure 85. RTI (left) and RDP (right) of a uniform quasi-two-dimensional ocean with a target and wake. Solid blue circles highlight the boat and wake. Parameters include: $f_0 = 10\ GHz,\ \ B = 100\ MHz, grazing\ angle = 5°, d_{alias-free} = 1000\ m, v_{alias-free} = 15\ m/s, T_{dwell} = 0.2\ s$, wind speed $= 24\ knots,\ R_s = 1000\ m$, upsea propagation, VV polarization, cross sea range $= 1000\ m$, cross sea resolution $= 10\ m$, no Gaussian variance or smoothing, no windowing. The boat and wake of Figure 84 were used in these plots.

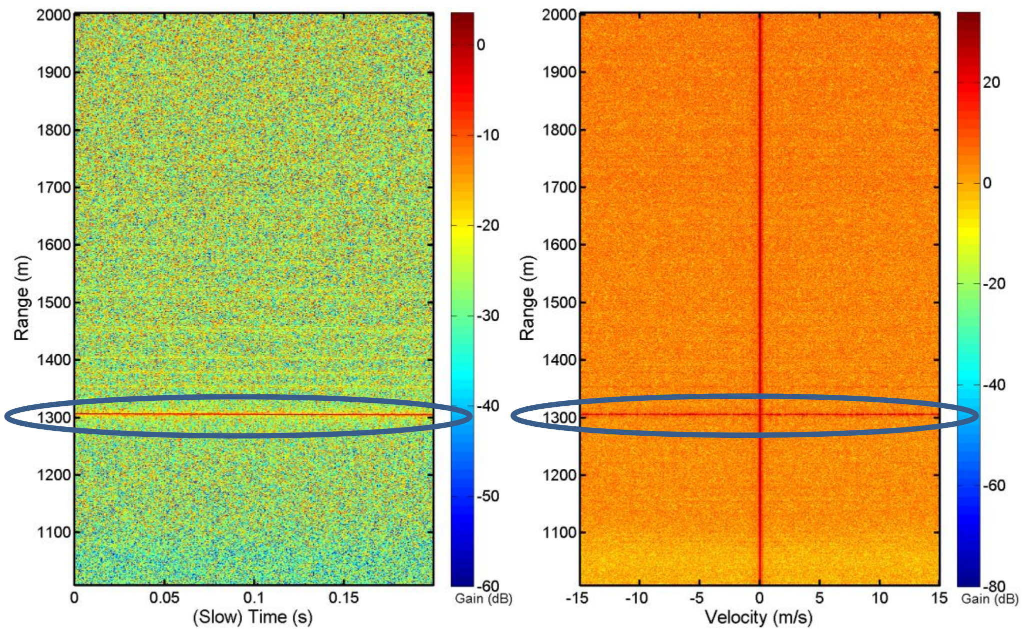


Figure 86. RTI (left) and RDP (right) of a uniform quasi-two-dimensional ocean with variance, target, and wake. Solid blue circles highlight the range of the bow of the boat. The same boat, uniform quasi-two-dimensional ocean, and radar parameters as were used in Figure 85 were used as a base model to generate these plots. The variance added to the ocean was: variance $= 1/3\ H_s$.

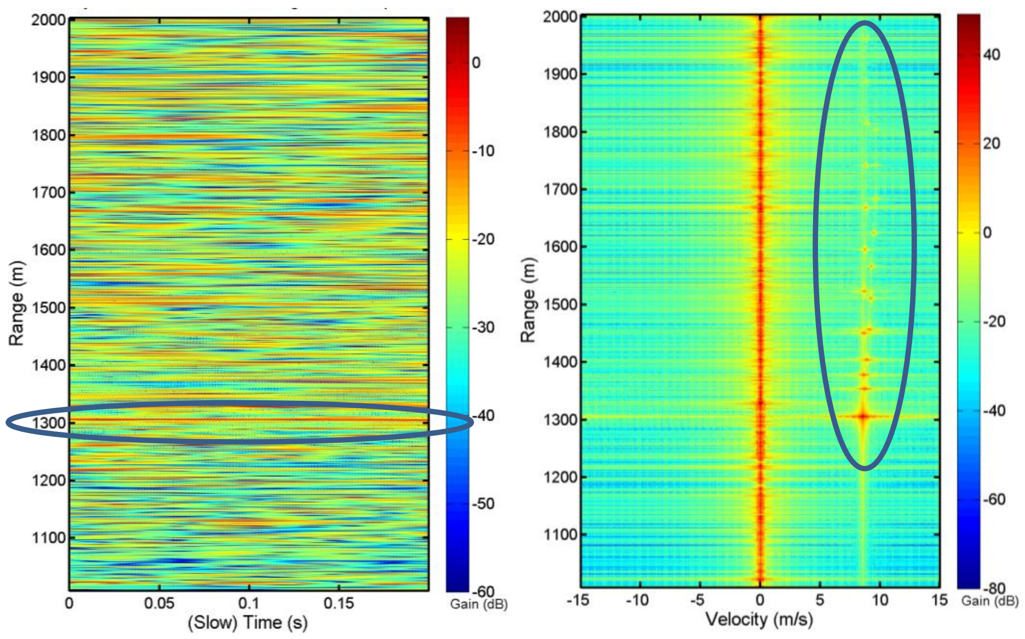


Figure 87. RTI (left) and RDP (right) of a uniform quasi-two-dimensional ocean with variance, smoothing, target, and wake. Solid blue circles highlight the boat and wake. The same boat, uniform quasi-two-dimensional ocean with variance, and radar parameters as were used in Figure 85 were used as a base model to generate these plots. The smoothing cross sea smoothing intervals $= 5$, sea smoothing intervals $= 10$, time smoothing period $= 3\ s$, no windowing.

We notice differences when comparing the RTI plots and RDPs of the three quasi-two-dimensional ocean surfaces to each other. Most notably, the RTI plot and RDP of the uniform quasi-two-dimensional ocean with variance looks drastically different than the other RTI plots and RDPs. Although the distribution of the RTI of all three of these examples have been matched to the empirical scattering distribution (including the case with variance), we believe the RTI and RDP of the ocean with variance is inaccurate. Unfortunately, there is little data to compare against in the public domain, so we cannot make this claim with evidence.

Because this two-dimensional ocean model is so connected to the one-dimensional model, it has the same limitations. Particularly, a Lagrangian mechanics model must be developed to generate more accurate RDPs. In the future, we also suggest the development of a two-dimensional ocean surface using the Philips model, which considers multi-directional propagation.

4 Phased Array Simulation

The non-scanning radar simulation previously discussed is a very capable model and aids in the understanding of maritime ocean scattering and allows for the development of small boat detection algorithms. However, phased array radar allows an extra degree of spatial freedom, and thus, enhances potential small boat detection performance. Therefore, it is important to develop a phased array simulation. In this section, we present background and methods on phased array antenna theory. As a future extension, it is possible to simulate ocean scatter using phased array radar as represented in Figure 88.

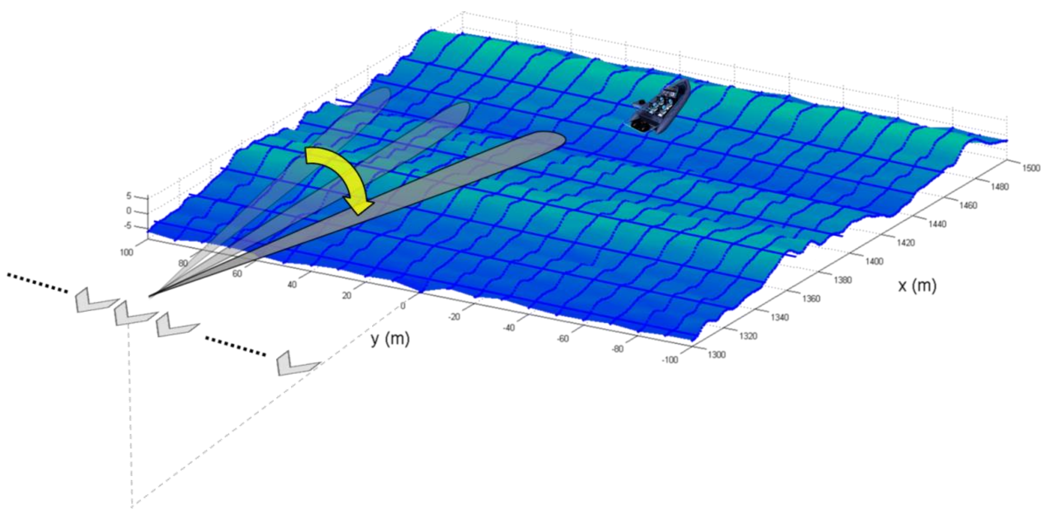


Figure 88. Visual representation of phased array radar, quasi-two-dimensional ocean, and target integration.

4.1 Background

Phased array radar consists of an array of antennas. This array can be one or two-dimensional, each dimension adding an extra degree of spatial freedom. In this discussion, we consider only the one-dimensional phased array that resolves in azimuth. Each of these antennas can be polarized HH or VV, and one polarization is used for the entire array. Phased array radars are very important in the maritime setting because the extra degree of freedom allows the determination of a target on a two-dimensional map. Because targets are always on or near the ocean surface, this two-dimensional map works well to identify the position of the target. Figure 89 shows an example of a phased array radar.

Figure 89. One-Dimensional phased array radar mounted on an airplane([Image of one-dimensional phased array], 2012).

Phased array radars work by varying the phase of the transmitted or received electromagnetic signals. These signals superimpose in free space according to N-slit diffraction. The overall effect of this superposition is an angled plane wave and a radiation pattern that is reinforced in a desired direction and suppressed in undesired directions. An example angled wave front is shown in Figure 90 and an example radiation pattern is shown in Figure 91. The angle of the resulting plane wave is called the scan angle, denoted θ. By changing the phase variation, the radiation pattern can be altered to scan at different angles. Because this change of phase variation can occur rapidly with electronic variable phase shifters, it is possible to electronically scan in angle.

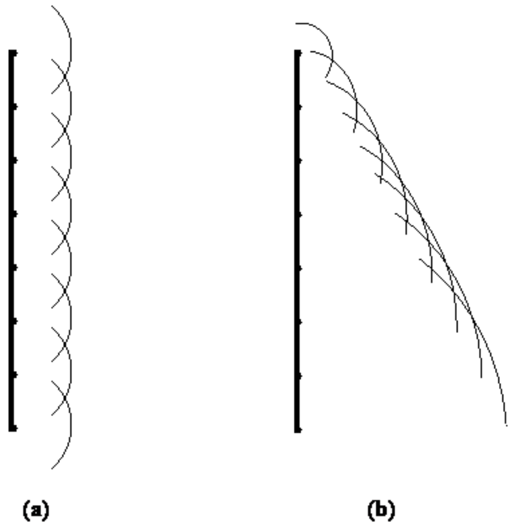


Figure 90. Phased array plane wave formation. (a) Waves emanating from discrete sources in a phased array interfere to create a plane wave that appears to come from a distant source behind the array (corresponds to a scan angle of 0°). (b) By emitting waves with different phase shifts, the array sources create a plane wave that propagates at an angle to the array surface (Wowk, 2007).

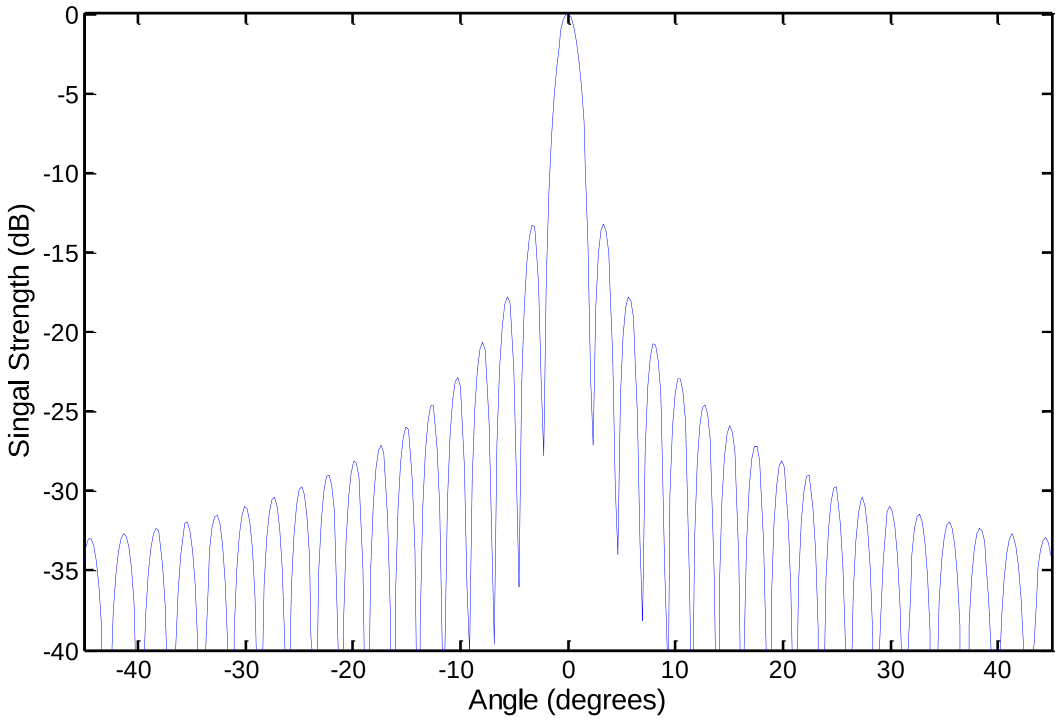


Figure 91. Example of a phased array antenna pattern at a scan angle of 0°.

The physical spacing and length of phased array radar determines its beam width, number of elements, and phase shift. Figure 92 shows the geometry that is described. Generally, the distance, d, between adjacent antennas is:

$$d = \frac{\lambda_0}{2} \tag{4 - 1}$$

where λ_0 is the center frequency wavelength. The beam width, denoted B_w, is the range of angles such that the amplitude of the antenna pattern is above $-3dB$. This beam width is approximated by:

$$B_w \approx \frac{\lambda_0}{D} \tag{4 - 2}$$

where D is the total array length. The beam width is particularly important because the beam width is the angle resolution. Using the element spacing described by Equation (4 - 1), it is possible to determine the number of elements, N_E, required for an array with a given array length.

$$N_E = \left\lceil \frac{D}{d} \right\rceil \tag{4 - 3}$$

Because it is impossible to have a fraction of an antenna, the ceiling function is taken. Using Equations (4 - 2) and (4 - 3), it is possible to determine the number of elements required given a desired beam width:

$$N_E = \left\lceil \frac{\lambda_0}{dB_w} \right\rceil \tag{4 - 4}$$

The phase shift between successive elements is (Monzingo & Miller, 1980, p. 50):

$$\Delta\phi = \frac{2\pi d}{\lambda}\sin(\theta_s[t_s]) = \frac{2\pi\lambda}{2\lambda}\sin(\theta_s[t_s]) = \pi\sin(\theta_s[t_s]) \tag{4 - 5}$$

where $\theta_s[t_s]$ denotes the scan angle of the phased array radar as a function of slow time (recall that phased array radars can electronically scan in time). Thus, the phase shift at element n is:

$$\phi_i = n \cdot \pi\sin(\theta_s[t_s]) \tag{4 - 6}$$

These parameters fully define the background necessary to simulate phased array radar.

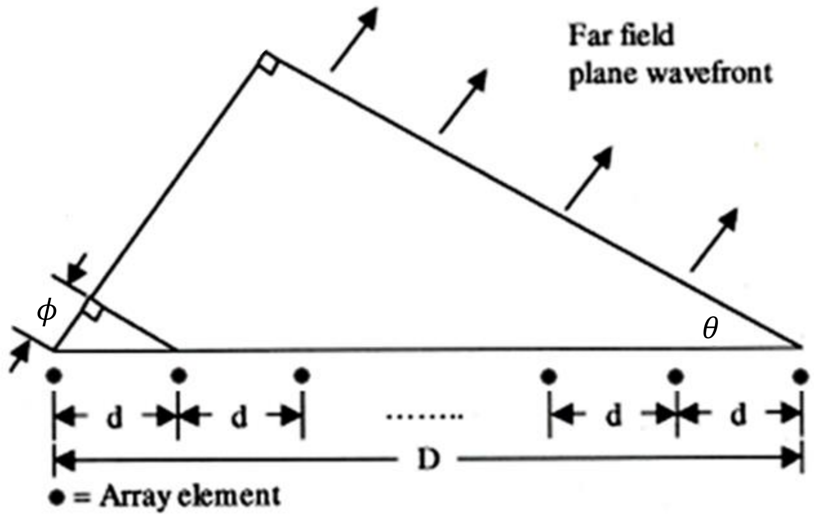


Figure 92. A phase-steered radar array (Toomay & Hannen, 2004, p. 31).

4.2 Methods

We now discuss the methods and simplifying assumptions used when implementing the phased array radar in MATLAB. Through the simplifying assumptions, we were able to realize 4000 times less computation. In this methods section, we first extend the radar equation for multiple antennas. After that, we add the effect of the phase shift on the radar return equation. Next, we investigate computational improvements to phased array processing. After radar returns have been generated, the same range, Doppler, and range-Doppler methods presented in Sections

2.2.1.2 to 2.2.1.4 apply to the phased array radar. Finally, through a discussion of phased array scanning, we extend the concept of the range time intensity plot to a range-angle plot.

In phased array radar, each antenna in the antenna array transmits simultaneously (and has a phase delay). An antenna of a phased array is called an element. After the transmitted signals scatter from targets and return to the phased array, each element receives each transmitted pulse. To help clarify the complexity of this process, consider the case where only one element is transmitting and the signal is received by each element (including the transmitting antenna), as shown in Figure 93. In the non-scanning radar case, there was only one antenna, and thus, one path for total travel distance. Now, there are multiple paths.

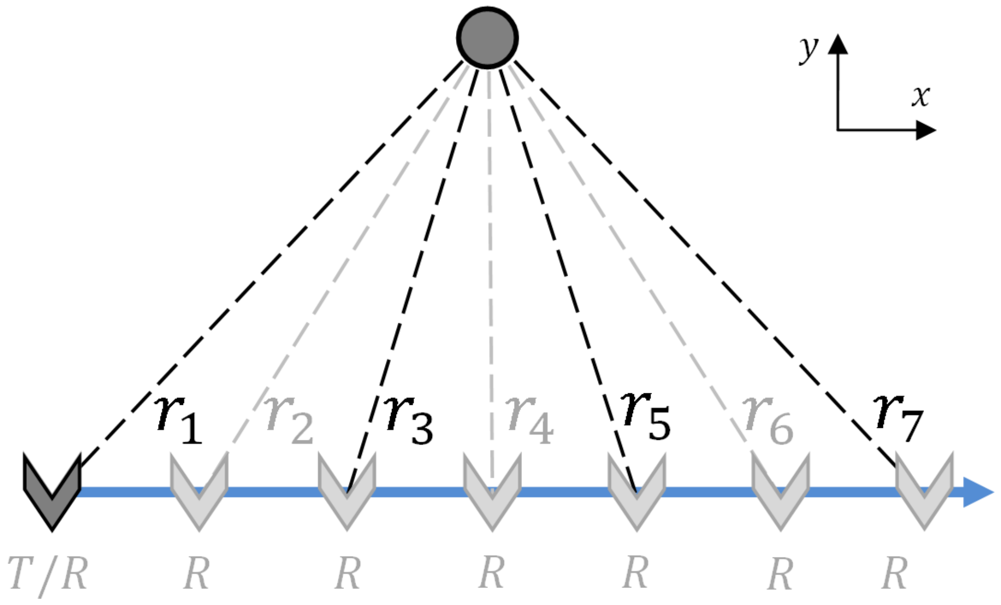


Figure 93. Antenna array with seven elements. The v shapes represent the elements, where the darker v shape at far left represents the transmitting element. Straight line paths are shown to the target, represented by the circle.

To use the radar return equation of the form of Equation (2 - 43), the total travel distance is required. In the non-scanning radar use of this equation, the total travel distance was given by Equation (2 - 4). Here, the total travel distance described by Equation (2 - 4) is only the case for the path from the transmitting element to the target and then back to the transmitting element. We define the one directional range from transmitting element ℓ to a stationary target as r_ℓ and the one directional range from the stationary target to receiving element n as r_n, as shown in Figure 93. Note that $r_\ell = r_n$ when $n = \ell$. The total travel distances for the paths from the first transmitting element are given by:

$$r_1 + r_n, \quad n = 1, 2, \dots, N_E \tag{4 - 7}$$

Equation (4 - 7) can be extended for synchronously transmitting elements:

$$r_\ell + r_n, \;\; n = 1, 2, \dots, N_E; \; \ell = 1, 2, \dots, N_E \tag{4 - 8}$$

The total number of paths is ${N_E}^2$. When the target is moving, we define $r_{\ell,\tau}[t_s]$ as the distance from the transmitting element to the moving target τ and $r_{n,\tau}[t_s]$ as the distance from τ to a receiving element in slow time. Furthermore, we express the height of the radar as h; absolute target distance as x_τ, absolute target cross-distance as y_τ , absolute element distance as 0, and absolute element cross-distance as y_E. We set the absolute position of the middle most element to the origin. Using these definitions and the Pythagorean Theorem, we further define $r_{\ell,\tau}(t_s)$:

$$r_{\ell,\tau}[t_s] = r_{n,\tau}[t_s] = \sqrt{h^2 + (y_E - y_\tau)^2 + {x_\tau}^2} \quad n = \ell = 1, 2, \dots, N_E \tag{4 - 9}$$

We can now update the radar scattering equation, Equation (2 - 43), to:

$$V[\omega, t_s] = W_\omega[\omega] W_{t_s}[t_s] e^{-2jk\cdot R_s} \sum_{\tau=1}^{\mathrm{N_T}} \sum_{\ell=1}^{N_E} \sum_{n=1}^{N_E} e^{-jk\left(r_{\ell,\tau}[t_s] + r_{n,\tau}[t_s]\right)} \tag{4 - 10}$$
$$n = 1, 2, \dots, N_E; \; \ell = 1, 2, \dots, N_E$$

The phase shift must also be implemented according to the path of the signal. Extending Equation (4 - 6), we found the total phase shift of a given path from transmitting element ℓ to receiving element n as:

$$(n + \ell)\, \pi \sin(\theta_s[t_s]), \quad n = 1, 2, \dots, N_E; \; \ell = 1, 2, \dots, N_E \tag{4 - 11}$$

We add this phase shift to the absolute phase term $-k\left(r_{\ell,\tau}[t_s] + r_{n,\tau}[t_s]\right)$ to update the radar scattering equation again:

$$V[\omega, t_s] = W_\omega[\omega] W_{t_s}[t_s] e^{-2jk\cdot R_s} \cdot$$
$$\sum_{\tau=1}^{\mathrm{N_T}} \sum_{\ell=1}^{N_E} \sum_{i=1}^{N_E} e^{-j\left(k\left(r_{\ell,\tau}[t_s] + r_{n,\tau}[t_s]\right) + (n+\ell)\,\pi \sin(\theta_s[t_s]),\right)}, \tag{4 - 12}$$
$$n = 1, 2, \dots, N_E; \; \ell = 1, 2, \dots, N_E$$

The peak of the antenna pattern allows distinguishability of targets in angle similar to how the peaks in range response allowed distinguishability in range. Similar to range and Doppler processing discussed in Sections 2.2.1.2 and 2.2.1.3, windowing enhances detection probabilities. Windowing in azimuth is possible by windowing over elements. The radar scattering equation for phased array radar is updated to:

$$V[\omega, t_s] = W_\omega[\omega] W_{t_s}[t_s] e^{-2jk\cdot R_s} \cdot \sum_{\tau=1}^{\mathrm{N_T}} \sum_{\ell=1}^{N_E} \sum_{n=1}^{N_E} W_{n,\ell}[n,\ell] e^{-j\left(k\left(r_{\ell,\tau}[t_s] + r_{n,\tau}[t_s]\right) + (n+\ell)\,\pi \sin(\theta_s[t_s])\right)}, \qquad (4 - 13)$$

$$n = 1, 2, \dots, N_E;\ \ell = 1, 2, \dots, N_E$$

Although Equation (4 - 13) has many components, it can be modeled relatively simply in MATLAB. Specifically, a five dimensional matrix could be generated with dimensions representing angular frequency (or, wavenumber), slow time, transmitting element index, receiving element index, and target index. After this matrix is generated according to the radar return equation, a sum could be applied across the last three dimensions to return a function of angular frequency, slow time, and mode, as desired.

Equation (4 - 13) was verified by finding the antenna pattern produced by the equation and comparing it to phased array antenna patterns found in literature. The antenna pattern can be found in simulation by fixing scan angle and generating the radar returns of a target traveling a circular arc centered on the radar. The signal strength as a function of the angle of the target to the radar is the antenna pattern in the plane of the target. The normalized two-way (transmit and receive) antenna pattern for phased array radar using a scan angle of 20° is shown below. Note that the first sidelobe has a height that is -26.4 dB down from the main lobe.

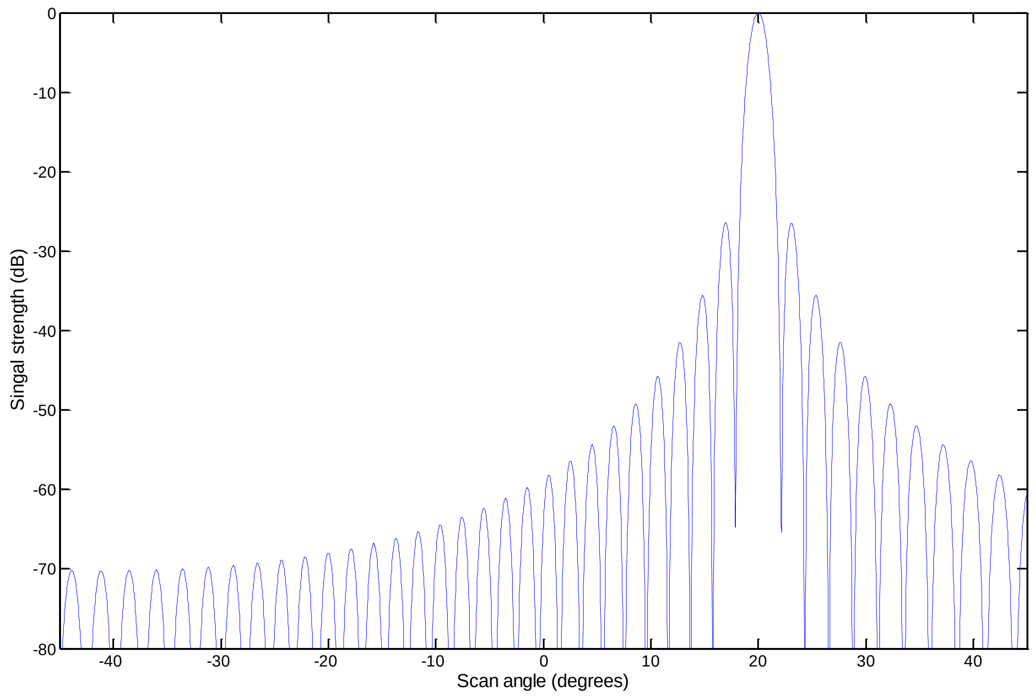


Figure 94. Two-way antenna pattern for a phased array using a scan angle of $20°$. Other parameters include: $f_0 = 10\ Ghz, B = 100\ MHz, BW = 2°$.

While Equation (4 - 13) is easily computed using MATLAB, it is not computationally efficient due to the sheer number of computations required. In fact, using big O notation, we realize that this function is $O\left(N_\omega \cdot N_{t_s} \cdot {N_E}^2 \cdot \mathrm{N_T}\right)$ given that N_ω is the number of angular frequencies and N_{t_s} is the number of slow times. The number of computations necessary could be reduced after noting the symmetry inherent in the element to element travel paths and after using the distributive property. That is, Equation (4 - 13) is equivalent to:

$$V[\omega, t_s] = W_\omega[\omega] W_{t_s}[t_s] e^{-2jk\cdot R_s} \cdot \sum_{\tau=1}^{\mathrm{N_T}} \left(\sum_{n=1}^{N_E} W_n[n] e^{-j\left(kr_{n,\tau}[t_s] + n\,\pi \sin(\theta_s[t_s])\right)} \right)^2, \qquad n = 1, 2, \dots, N_E \tag{4 - 14}$$

Now, the number of computations necessary is $O\left(N_\omega \cdot N_{t_s} \cdot N_E \cdot \mathrm{N_T}\right)$. Thus, N_E times less computations are necessary. For reference, when the desired angle resolution (beam width of the main beam) is approximately $2°$, 57 elements are required. In this case, using Equation (4 - 14) instead of Equation (4 - 13) yields on the order of 57 times less computation; thus theoretically, 57 times less computation duration. In reality, the MATLAB multi-dimensional vectorization tools have a small amount of built in parallelization, so the computational time decrease is less (in this case, on the order of 20 times less). With smaller desired angle resolution, proportionally more elements are required, and more time is saved.

Further improvements are possible after rearranging Equation (4 - 14) to:

$$V[\omega, t_s] = W_\omega[\omega] W_{t_s}[t_s] e^{-2jk\cdot R_s} \cdot \left(\sum_{\tau=1}^{\mathrm{N_T}} e^{-2jkr_{0,\tau}[t_s]}\right)\left(\sum_{n=1}^{N_E} W_n[n] e^{-j(k\mathscr{r}_n[t_s] + n\,\pi \sin(\theta_s[t_s]))}\right)^2 \qquad (4 - 15)$$

$$n = 1, 2, \dots, N_E$$

This rearrangement is derived in Appendix A.3. Please refer to the appendix for the definition of $\mathscr{r}_n[t_s]$. The reorganization reduces the computation to $O\big(N_\omega \cdot N_{t_s} \cdot N_{\mathrm{T}} + N_\omega \cdot N_E \cdot N_{t_s}\big)$. Thus, Equation (4 - 15) takes $\frac{N_\omega \cdot N_E \cdot N_{\mathrm{T}}}{N_{\mathrm{T}} + N_E}$ times less computation than Equation (4 - 14). We also note that we recognize the rightmost term of Equation (4 - 15) as the definition of the antenna pattern.

We make one final computational improvement by noting that the antenna pattern does not appreciably change with k for narrowband signals. For example, Figure 95 shows two overlaid antenna patterns at the extents of a $100\ MHz$ bandwidth signal with a center frequency of $10\ GHz$. Clearly, the difference between the two antenna patterns is small. Thus, we can approximate k by k_0, the center wavenumber:

$$V[\omega, t_s] = W_\omega[\omega] W_{t_s}[t_s] e^{-2jk\cdot R_s} \cdot \left(\sum_{\tau=1}^{\mathrm{N_T}} e^{-2jkr_{0,\tau}[t_s]}\right)\left(\sum_{n=1}^{N_E} W_n[n] e^{-j(k_0\mathscr{r}_n[t_s] + n\,\pi \sin(\theta_s[t_s]))}\right)^2 \qquad (4 - 16)$$

$$n = 1, 2, \dots, N_E$$

This function is $O\big(N_\omega \cdot N_{t_s} \cdot N_{\mathrm{T}} + N_E \cdot N_{t_s}\big)$, which requires approximately ${N_E}^2$ less computations than the original phased array radar return equation, Equation (4 - 13) (assuming $N_\omega \cdot N_{\mathrm{T}} \gg N_E$). For a moderate beam width, the use of this function corresponds to approximately 4000 times less computation than the original phased array radar return equation.

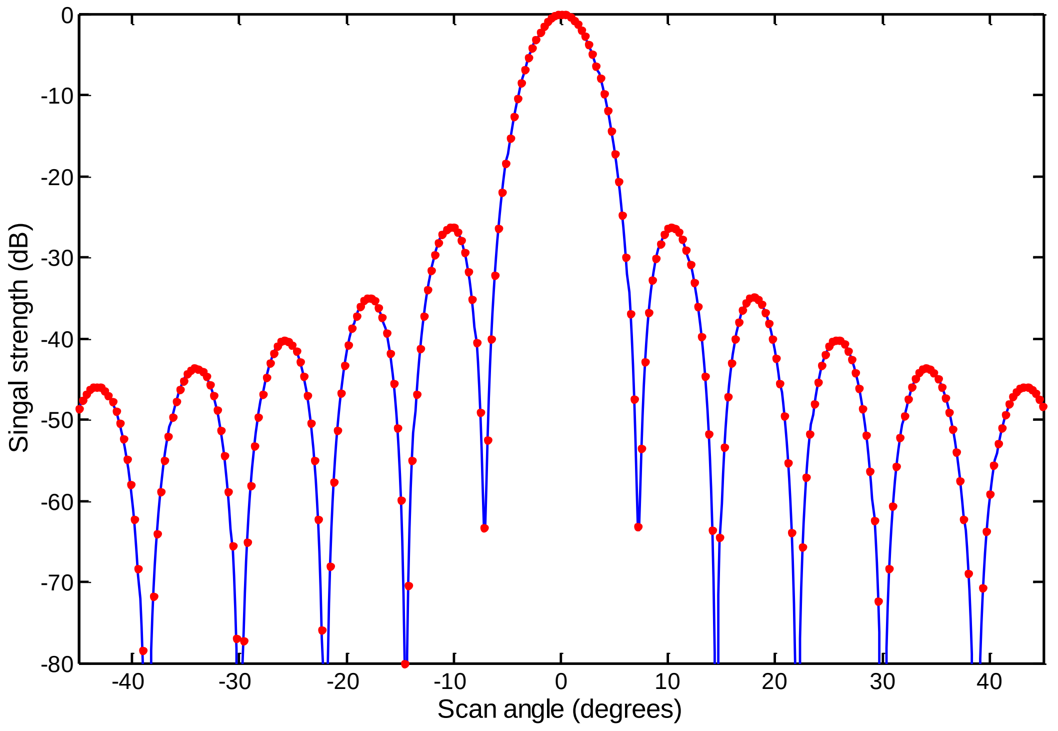


Figure 95. Two antenna patterns with different center frequencies. The solid blue line represents the antenna pattern for a center frequency of 10 GHz and the red markers represent the antenna pattern for a center frequency of 10.1GHz. The beam width was 7 degrees in both cases. Clearly, there is not a significant difference between the two antenna patterns.

After the radar returns are generated, the same range-Doppler processing techniques that were presented in Section 2.2.1.4 can be applied to generate RTIs and RDPs. Because the scan angle of the radar is a function of slow time, the slow time axis of the RTI can be transformed to scan angle. The result is referred to as a Range Angle Plot (RAP), and is very useful because it allows target distinguishability in both range and angle. With these two parameters, it is possible to locate a target in two-dimensional space.

4.3 Results and Discussion

Similar to the discussion of RTI and RDP in Section 2.2.1.4, the RAP and RDP of the phased array simulation can be broken into dwell times and a time series can be created. The time series movies of the RAP and RDP for this example can be found in the accompanying attachment to this Major Qualifying Project report. Example time slices for the RAP and RDP are shown in Figure 96 and Figure 97, respectively.

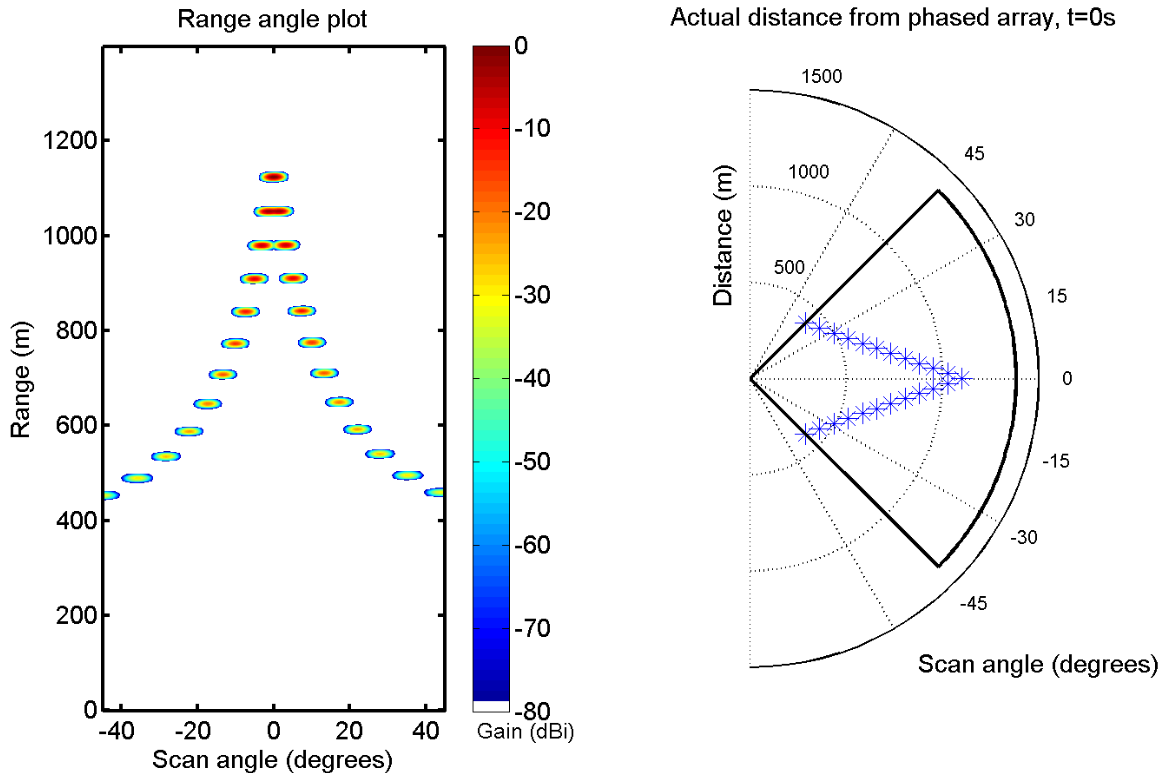


Figure 96. Example of phased array range angle plot. The parameters used to generate this plot were: $f_0 = 10\ GHz, B = 50\ MHz, r_{alias-free} = 1400, v_{alias-free} = 20\frac{m}{s}, B_W = 2°, h = 200m.$ Target parameters include: $d_0 = 1100\ m, V_{boat} = 10\ m/s$, traveling away from radar at 0 degree angle, Kaiser windowing.

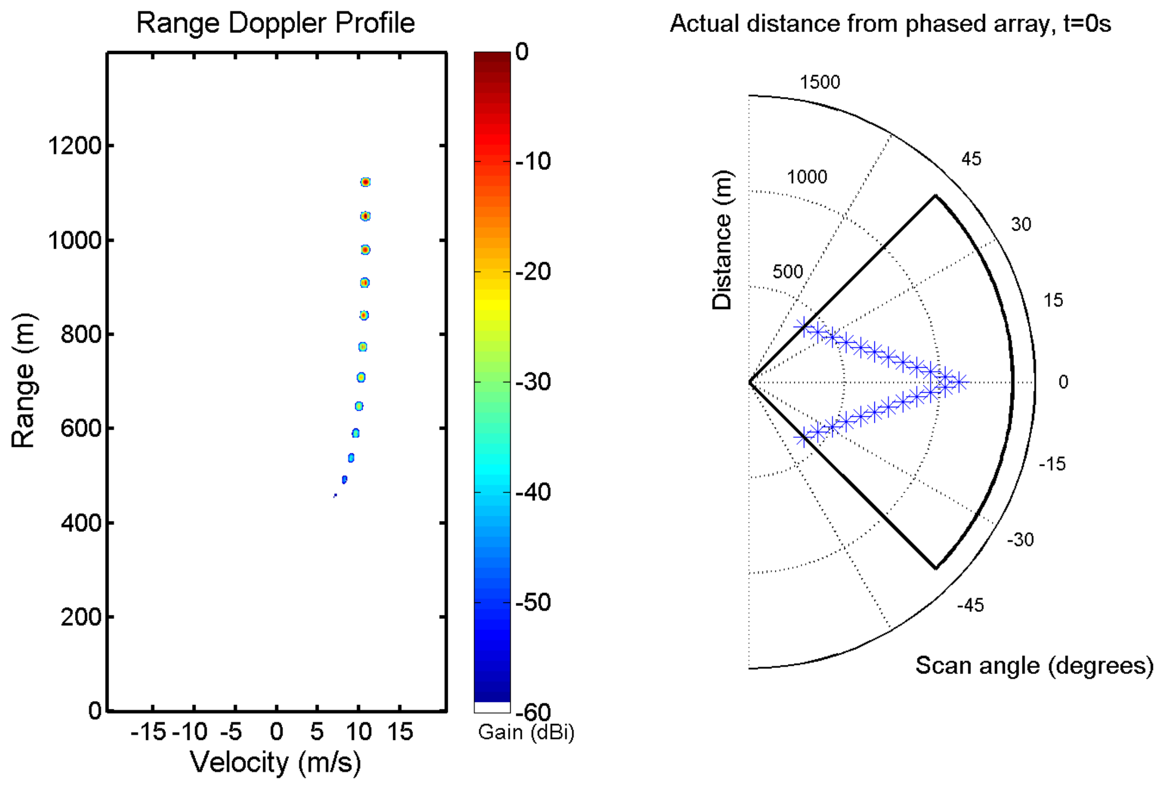


Figure 97. Example of phased array range Doppler profile. The parameters of Figure 96 were used.

We were very successful at developing a computationally efficient phased array radar simulation capable of producing the radar returns of boats and their wakes (ocean clutter was not modeled in this simulation). The computation speed can be further improved via parallelization as discussed in Section 5.2.2. Such computational improvements are critical to enable realistic computation of a phased array radar interrogating a quasi-two-dimensional ocean model. With this phased array simulation, the quasi-two-dimensional ocean model, and the architecture framework discussed in Section 3.2.3, we recommend the development of a phased array quasi-two-dimensional ocean scattering simulation as a future extension to this project.

5 Parallelization

Processing time is a significant consideration when modeling maritime radar due to the large volumes of data. An ocean can be modeled as a set of points sampled from a simulated ocean. These oceans can be tens of kilometers on each axis and sampled at sub-meter intervals. Furthermore, an ocean may be simulated for large extents of time with high time resolution. This huge amount of data and computation are compounded more by the radar simulation. First, radar pulses are sent up to thousands of times per second for our application. Next, each of the thousands of reflected pulses must be sampled thousands of times. Phased arrays further exacerbate the problem as there are many antennas each performing these operations. The angle and range resolution required by the maritime processing environment require tens to hundreds of antennae and even more samples per pulse. This large number of samples equates to trillions of calculations of processing for one second of simulated radar time.

5.1 Background

These enormous calculations highlight the need to optimize computation efficiency. Efficiency can be enhanced through both more efficient or simplified algorithms and through the use of parallel processing. More efficient and simplified algorithms for our application are discussed in Section 4.2. Parallelization is generally broken into two categories, data parallelism and task parallelism. Task parallelism focuses on the parallel distribution of tasks across multiple processors whereas data parallelism distributes data across processors. For example, running a job that requires hundreds of different operations distributed over many processors to be executed on one set of data would be task parallelism while a large set of data distributed over many processors with one set of operations would be data parallelism.

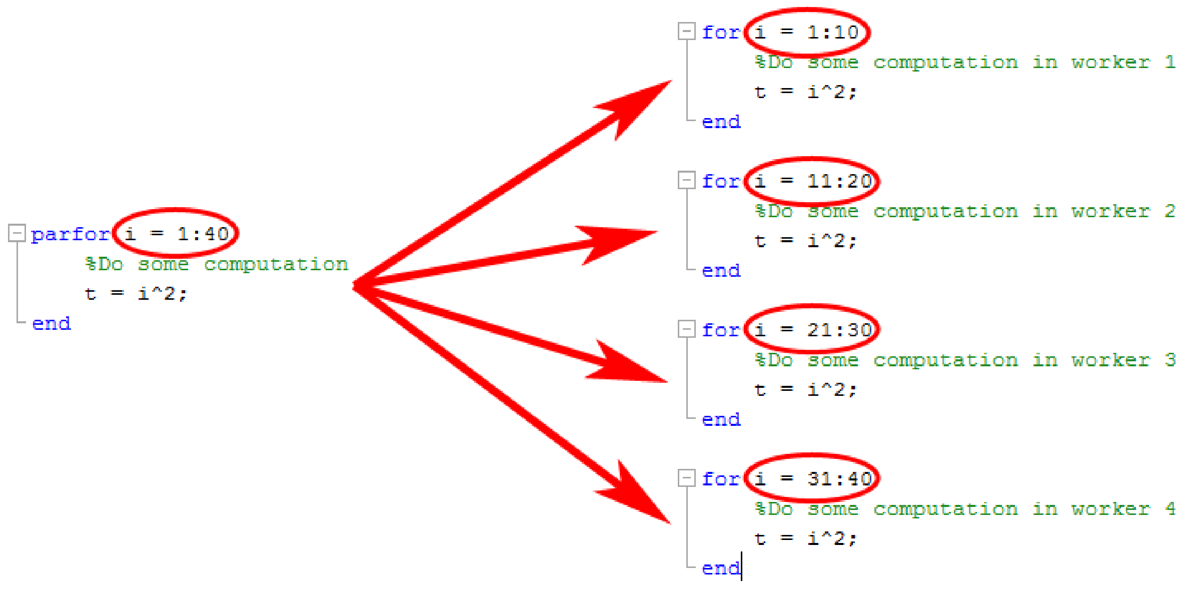
In the past, most parallel processing was performed on large expensive computers or a computer cluster, many of which leverage multiple processors. In recent years, multiple core processors have become prevalent even for personal workstations. MIT Lincoln Laboratory has both of these technologies as well as two MATLAB libraries for parallelization. The first library is pMatlab for use on a compute cluster called LLGrid or on a workstation. The laboratory also has a license for the MATLAB Parallel Computing Toolbox. By leveraging the ability to parallelize processing and computation among many different processors or cores, our model realized significant speedups in overall computation times. Furthermore, parallelization considerably aids in the usability of the software and bolsters this model for use in research and further development.

One of the major limitations of parallel processing lies in the serial portion of the code. Amdahl's law (Amdahl, 1967, pp. 483-485) states that the theoretical maximum overall speedup is:

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

where N is the number of processors/cores and P is the portion of code that can be made parallel. In practice, however, speedups close to those predicted by Amdahl's law are hard to realize due to additional setup required in the program to support parallelism and bus limitations.

The MATLAB parallel computing toolbox allows for easy parallelization using parallel "for" loops (`parfor`) and the `spdm` command. Both commands rely on "pools" of "workers." A worker can be either a node on a computing cluster or a thread which is scheduled to run on a logical processor within a multicore computer. A pool constitutes a collection of these workers and may consist of either local workers or nodes on a computer cluster. The `parfor` command operates by splitting a "for" loop by iteration and allocating portions of the total iterations to individual workers. A loose visual interpretation of a `parfor` is show in Figure 98. The `spmd` command is an acronym for "single program multiple data". As suggested by the acronym, each worker executes the same program on different data. Most programs leverage the command by distributing pieces of a large set of data to each worker. This process is illustrated in Figure 99. In the figure inside of the node boxes, the variable distributionScheme (a parameter passed to `zeros` function) refers to how the data are distributed to the workers. Data may be distributed in a block, cyclic, or block cyclic scheme as shown in Figure 100.



```
parfor i = 1:40
    %Do some computation
    t = i^2;
end
```

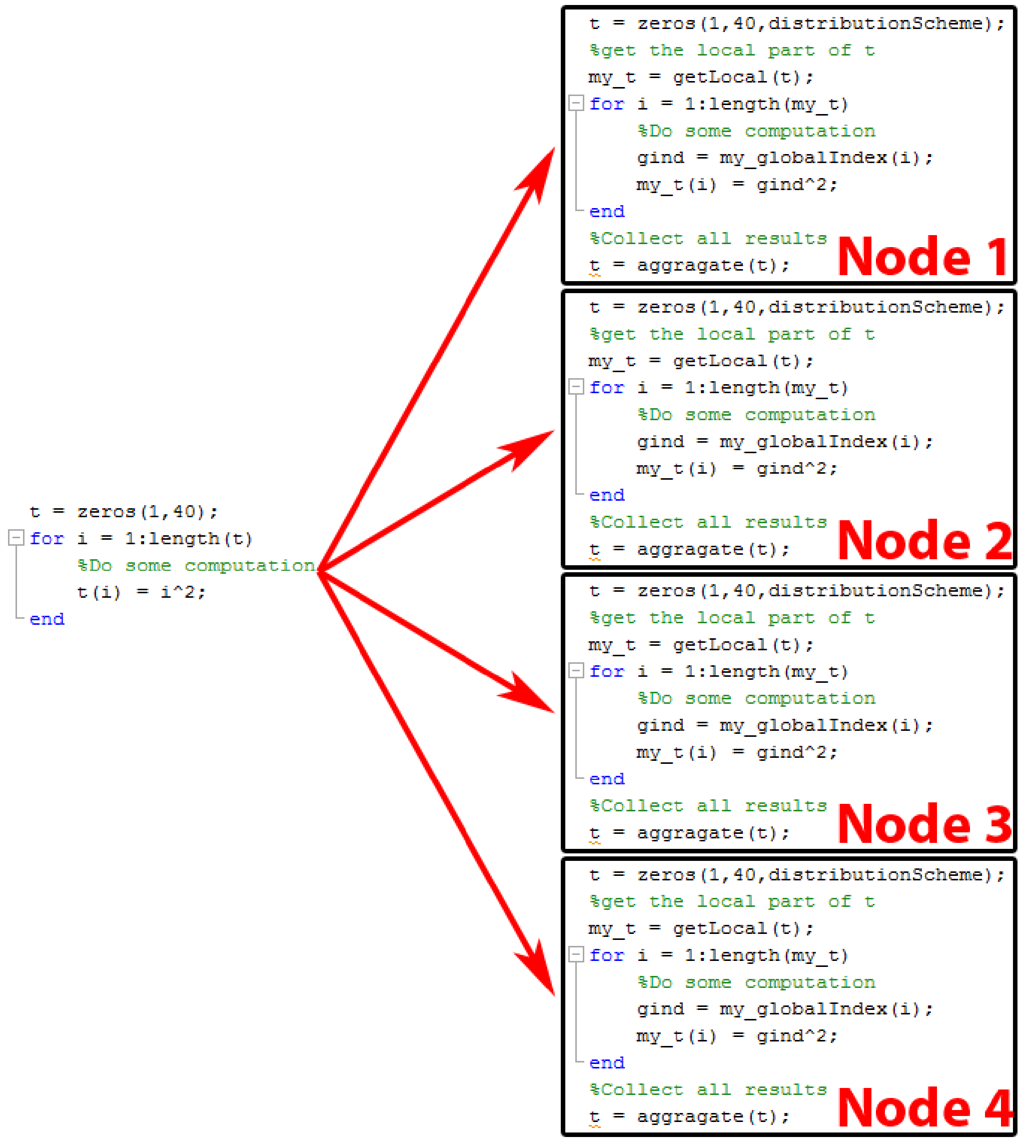
```
for i = 1:10
    %Do some computation in worker 1
    t = i^2;
end
```

```
for i = 11:20
    %Do some computation in worker 2
    t = i^2;
end
```

```
for i = 21:30
    %Do some computation in worker 3
    t = i^2;
end
```

```
for i = 31:40
    %Do some computation in worker 4
    t = i^2;
end
```

Figure 98. MATLAB command `parfor` distributing “for” loops to multiple workers.



```
t = zeros(1,40);
for i = 1:length(t)
    %Do some computation
    t(i) = i^2;
end
```

```
t = zeros(1,40,distributionScheme);
%get the local part of t
my_t = getLocal(t);
for i = 1:length(my_t)
    %Do some computation
    gind = my_globalIndex(i);
    my_t(i) = gind^2;
end
%Collect all results
t = aggragate(t);
```

Figure 99. Method used by MATLAB command `spmd` and pMatlab for distributing data to multiple workers or nodes. Each box represents a different processor on a workstation or compute node on a cluster. Note that the code is the same for each node but the data is different.

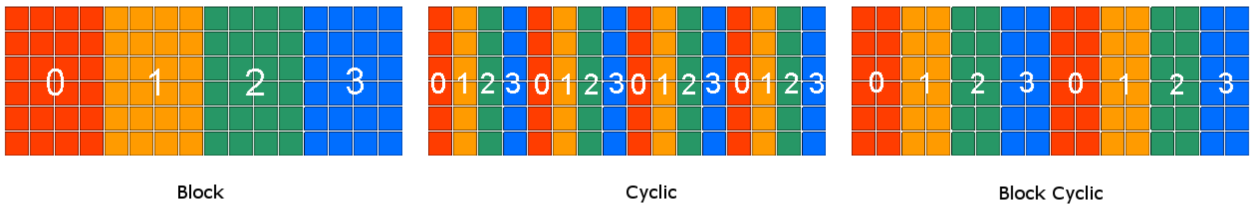


Figure 100. Matrix data distribution schemes.

In effect, the `parfor` operation is a simplified `spmd` command in that there is one "program" within the "for" loop which executes on different data. When using `parfor`, the data are sent as needed to the workers rather than all at once as in the `spmd` command, which can lead to higher communication overhead compared to the `spmd` command. These capabilities can be extended through the use of the MATLAB Distributed Computing Server which distributes the parallel code and data to machines in a cluster similar to the LLGrid.

Recall that our other parallel computing option is the LLGrid. The LLGrid was constructed in 2003 and is composed of 750 Dell PowerEdge 2850s. Each node contains two Intel Xeon 3.2 GHz single-core processors. We were allocated 64 of these processors for the duration of the project. There are two methods of interfacing with the LLGrid. In the first method, the user provides a function which is called on each node of the cluster. This method requires that all data which are to be passed out of the function are written to disk within the function, slowing subsequent computation. Additionally, parallelization of small regions within a set of computations is difficult because a function must enclose the computations. The second method has an environment setup script which evaluates an entire script in parallel with distributed data structures. This method is identical to the theory behind the `spmd` command in the Parallel Computing Toolbox; again, Figure 99 illustrates this concept. Using distributed data structures, it is possible to perform many computations on a set of data before finally reconstructing the distributed data into a data set with the entire results of the computation. Given the nature of our project, the second method is optimal due to the ease at which data can be distributed for computation. The largest considerations when using the parallel MATLAB solutions of the MIT Lincoln Laboratory is the setup time and communication overhead. Our tests found a significant amount of setup overhead when starting a parallel job. Additionally, small jobs are significantly impacted by communication between processing nodes.

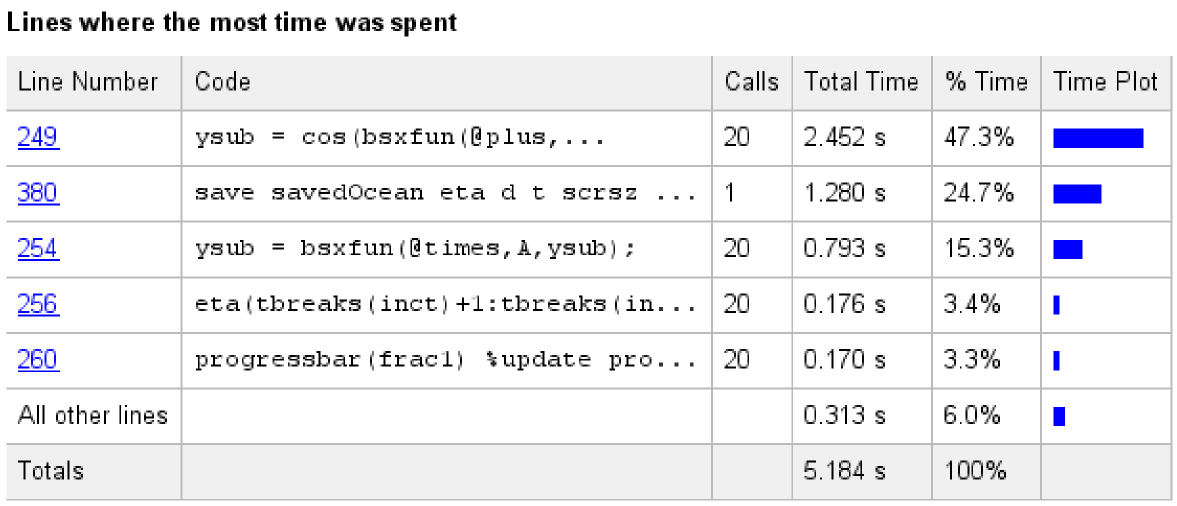
5.2 Methods

Performance enhancements required a thorough understanding of the code, its structure, and the underlying theory which the code expresses. The team was able to implement a parallel version which leveraged pMATLAB and the LLGrid. The first step to implementation was to understand where parallelization was needed through profiling.

5.2.1 Profiling

Often the first step in improving the performance of a program is to “profile” the program. Simple statistical profiling can be performed by having the code under test interrupted at regular intervals and sampled for their place in execution. From the samples, “hot spots” within the code can be identified by virtue of a larger number of samples occurring at a specific point in the code. MATLAB provides the `profile` function to perform real time profiling on MATLAB functions or scripts. The MATLAB profiler is somewhat more complex than a simple statistical profiler but the same general theory holds. Analysis and enhancement were performed in the logical order which the different sub-modules of the simulation are run according to Figure 78. Using the ocean model, we will discuss the process of profiling and, in turn, parallelization. These processes hold for the other sub-modules of the simulation.

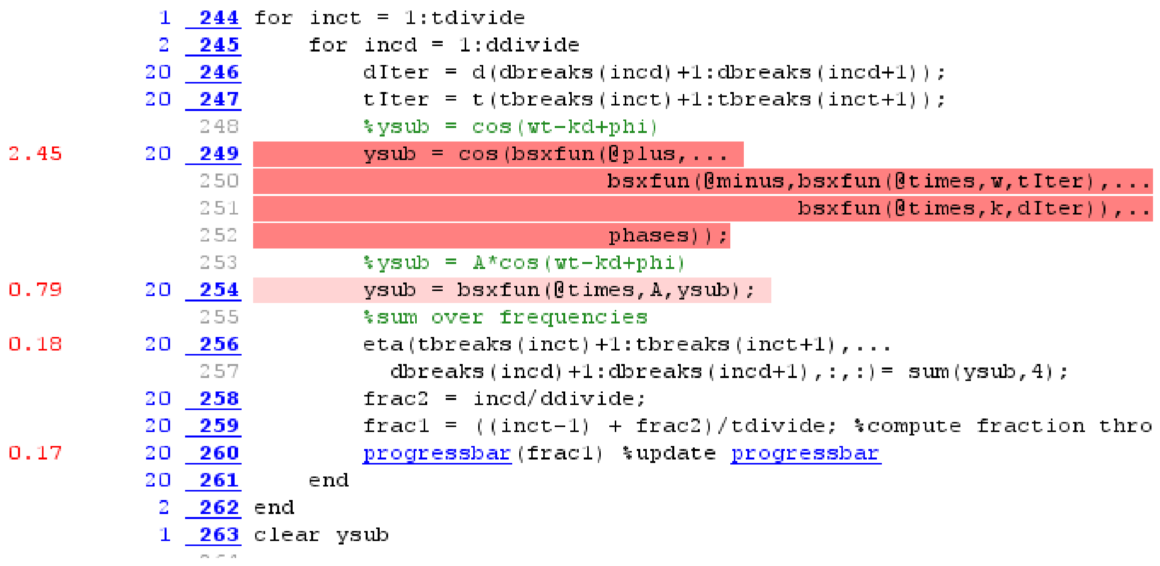
After profiling the ocean model, it was apparent that the calculation and summation of the different wave components consumed the majority of time. The MATLAB profiler displays this information in two different ways: 1) by tabulating the results sorted by most time spent on an individual line (Figure 101), and 2) by adding profiling statistics inline with the code (Figure 102). The tabulated version is useful to determine the computationally intensive sections of code. These sections may then be further analyzed for restructuring/optimizing in the inline statistics version. The first column of numbers in Figure 102 is the amount of time spent on that line. The next column of numbers is the number of times the line is called. It is apparent that, for this ocean, the nested loops resulted in twenty iterations of the code. Also notice how a red hue is added to lines which require significant time where the shading darkness increases for lines with more time.



Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
249	`ysub = cos(bsxfun(@plus,...`	20	2.452 s	47.3%	▬▬▬▬
380	`save savedOcean eta d t scrsz ...`	1	1.280 s	24.7%	▬▬
254	`ysub = bsxfun(@times,A,ysub);`	20	0.793 s	15.3%	▬
256	`eta(tbreaks(inct)+1:tbreaks(in...`	20	0.176 s	3.4%	▮
260	`progressbar(frac1) %update pro...`	20	0.170 s	3.3%	▮
All other lines			0.313 s	6.0%	▮
Totals			5.184 s	100%	

Figure 101. Listing of lines where the most time was spent.



```
          1  for inct = 1:tdivide
          2      for incd = 1:ddivide
         20          dIter = d(dbreaks(incd)+1:dbreaks(incd+1));
         20          tIter = t(tbreaks(inct)+1:tbreaks(inct+1));
                     %ysub = cos(wt-kd+phi)
2.45     20          ysub = cos(bsxfun(@plus,...
                                 bsxfun(@minus,bsxfun(@times,w,tIter),...
                                        bsxfun(@times,k,dIter)),..
                                 phases));
                     %ysub = A*cos(wt-kd+phi)
0.79     20          ysub = bsxfun(@times,A,ysub);
                     %sum over frequencies
0.18     20          eta(tbreaks(inct)+1:tbreaks(inct+1),...
                       dbreaks(incd)+1:dbreaks(incd+1),:,:)= sum(ysub,4);
         20          frac2 = incd/ddivide;
         20          frac1 = ((inct-1) + frac2)/tdivide; %compute fraction thro
0.17     20          progressbar(frac1) %update progressbar
         20      end
          2  end
          1  clear ysub
```

Figure 102. Code with inline profiling statistics.

From the summary of lines consuming the most time, we observe that most (greater than 69 percent) of the time is spent in the loops summing the cosine components of the waves together. Analysis of the code within the loops reveals that there exists no inter-loop dependencies i.e. any one loop does not depend on the results of a prior loop. This lack of interdependency suggests the workload is embarrassingly parallel (Harwood, 2003).

5.2.2 Parallelization

An embarrassingly parallel problem is defined as a computational problem which is easily subdivided into parallel tasks. Radar processing is considered an embarrassingly parallel problem as it is easily decomposed into smaller parallel subtasks. The simulation of an ocean is also an embarrassingly parallel problem as the ocean can be broken down into spatial regions and/or temporal regions which can be computed independently. Embarrassingly parallel tasks gain substantial speedup easily through parallelization. An additional benefit of sub-dividing the calculations is the reduction of system memory needed.

For example, consider a one-dimensional ocean that is 10 km long. This ocean is sampled at one meter intervals and every 2,000th of a second for 10 seconds. The resulting matrix to hold the ocean would be size 20,000 by 10,000 with each element consuming 8 bytes. The resulting matrix would be approximately 1,526 Megabytes. This matrix consumes less than the amount of random access memory (RAM) standard with computer systems sold today (2012); however, during intermediate processing steps, an additional dimension is added to this matrix. Our model regularly has up to 126 component frequencies which compose this dimension. In this case, the computation requires a matrix that is 20,000 by 10,000 by 126 resulting in the intermediate matrix approximately 188 Gigabytes in size which is significantly larger than the available RAM in typical computers today. This large matrix highlights the need for subdividing a job into smaller, more manageable sizes. The simplest way of subdividing the job is to iterate over one of the dimensions of the matrix. It is ideal if the dimension chosen does not result in inter-iteration dependencies. In the case of the ocean model, all of the component frequencies are summed as part of each iteration; therefore, the dimensions containing time or distance are better choices to iterate over than component frequencies.

A more complex, yet faster option is to distribute the data to multiple physical computers. Both pMatlab and the Parallel Computing Toolbox support distributed matrices. Figure 103 shows an example of how the matrices are distributed in pMatlab and a very similar scheme is used in the Parallel Computing Toolbox. On the other hand, parallel processing using the `parfor` command does not allow the programmer to determine the distribution of data to each of the workers. Within the `parfor` command, new data are supplied to a worker every time it completes an iteration of the “for” loop.

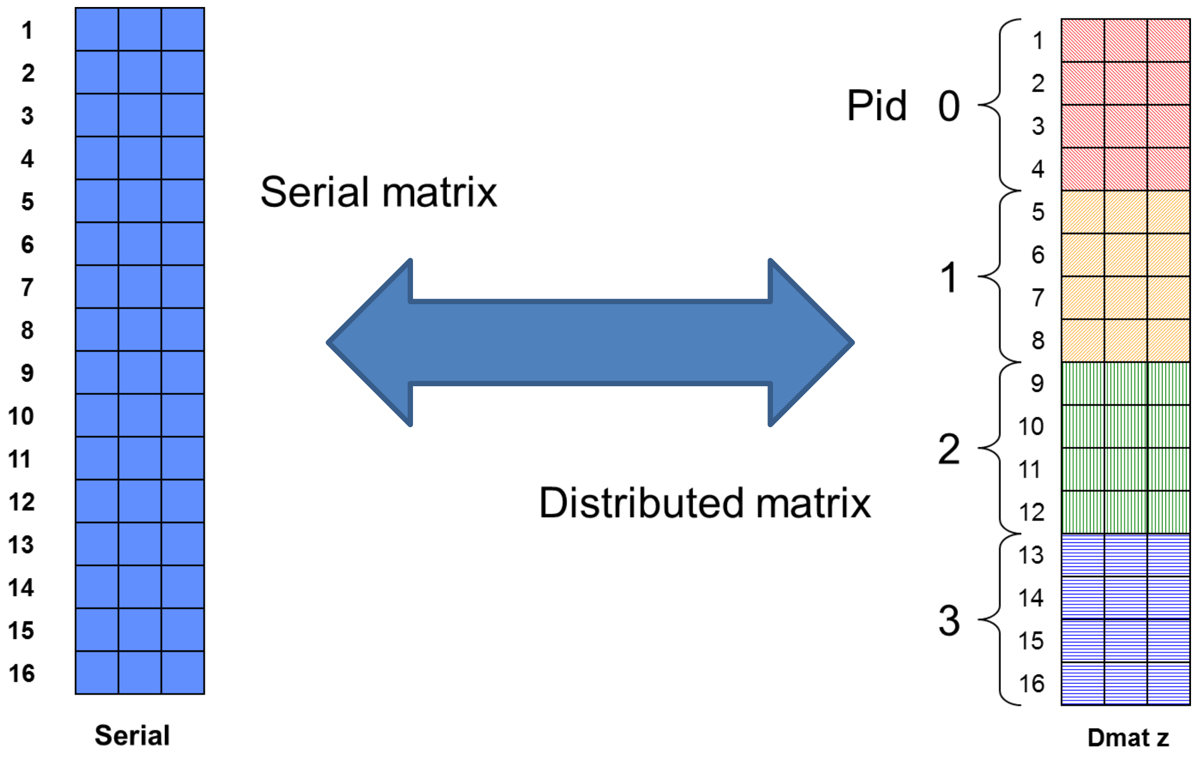


Figure 103. Serial and distributed matrices. The matrix on the left is a serial matrix whereas the matrix on the right is a distributed pMATLAB matrix (Byun, 2012).

All of our models are easily parallelized by subdividing over either distance, time, or both. The choice of this subdivision is performed dynamically by evaluation of the resulting sub-matrix sizes. It is ideal to operate on as large of a matrix as possible within the memory limits of the system to reduce communication overhead, and thus, realize speedups. Figure 104 illustrates the memory usage over time for the largest ocean our test computer could generate. Region 1 is the orthogonal replication of the ocean. Region 2 is the calculation of the dot product of the surface normal unit vector and the radar line of sight unit vector. Region 3 is the calculation of the radar returns described in Equation (3 - 34) over a portion of the total simulated time and a single wave number. Region 4 is the repetition of the aforementioned calculation for each wave number. Regions 2, 3, and 4 repeat over each portion of the total simulated time. Once the computations have been performed, the data are aggregated, as shown in Figure 105.

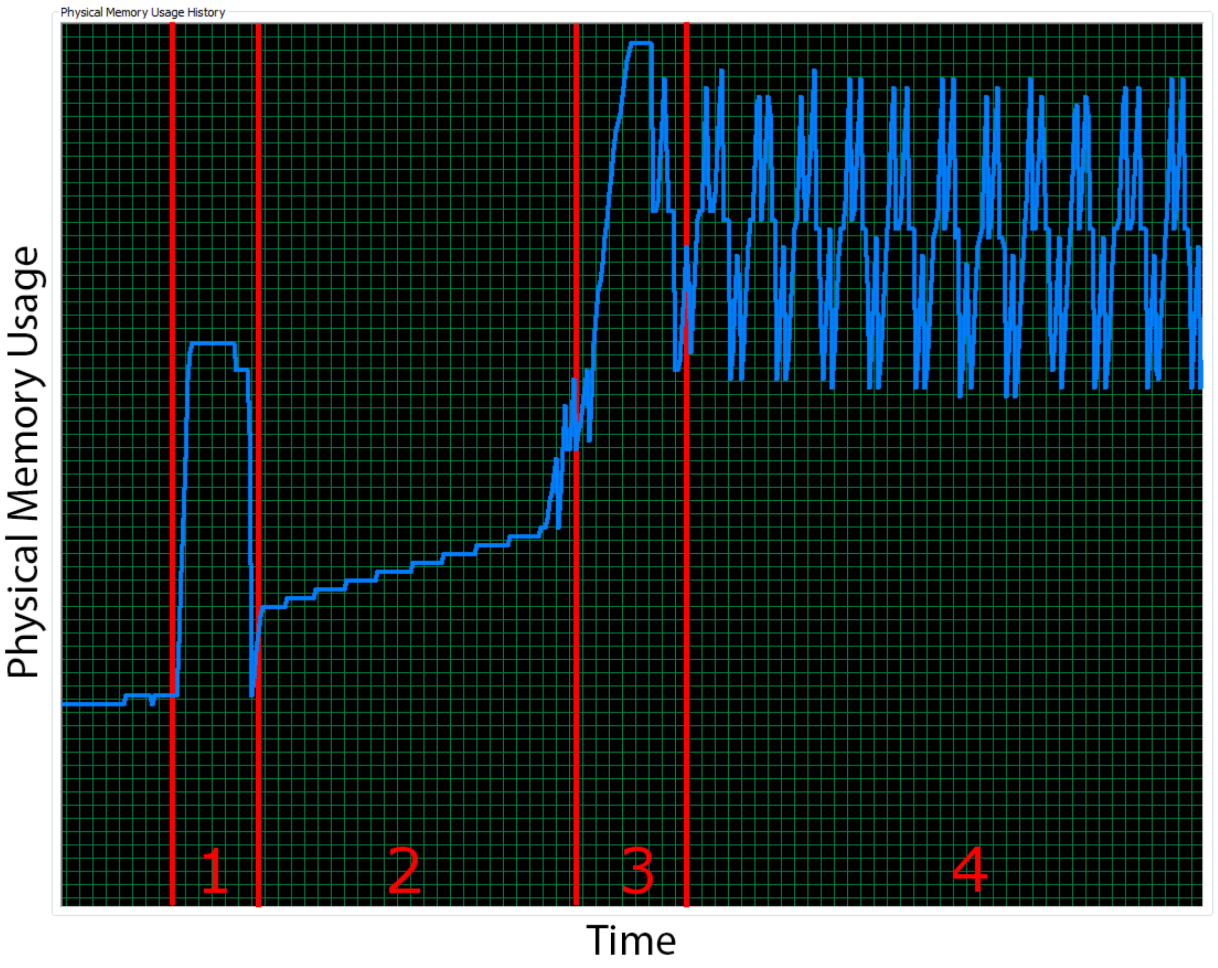


Figure 104. Memory usage over time for a two dimensional ocean. Parameters include: $f_0 = 10\ GHz,\ \ B = 100\ MHz, grazing\ angle = 5°, Ocean\ Length = 1000\ m,\ prf = 2000\ Hz$, simulation time = 3.85 s, wind speed $= 24\ knots,\ R_s = 1000\ m$, upsea propagation, VV polarization, cross sea range $= 1000\ m$, cross sea resolution $= 10\ m$, No Gaussian variance or smoothing, no windowing.

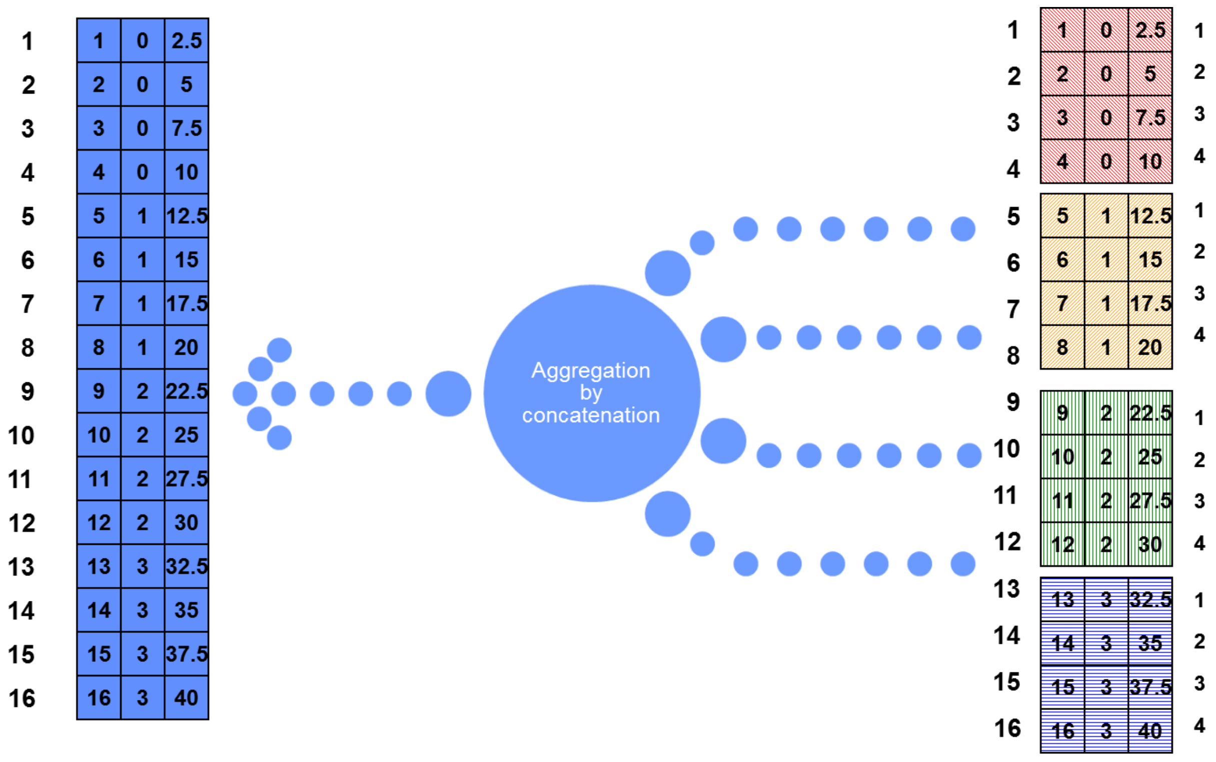


Figure 105. Aggregation of a distributed matrix. pMatlab and codistributed Parallel Computing Toolbox matrices are recombined at the end of processing using a method similar to that depicted above.

We investigated two primary methods of parallelization using pMatlab: 1) launching one function multiple times where each launch of the function is treated as a separate job with one worker and 2) one script may be launched and treated as one job with many workers. We found our application to be better suited to be executed using the latter method with one job and many workers. Each sub-job is a separate instance of the script running, whereas each node executes the entire script. Only when the data are distributed does pMatlab see significant returns in terms of reduction of compute time and memory usage.

5.3 Results and Discussion

Overall, we were successful at implementing a parallel solution in pMatlab. This parallelism allows our simulation to be easier to use by the Lincoln staff. In our investigation, we found the Parallel Computing Toolbox to be lacking for our application. We first discuss the shortcomings of the Parallel Computing Toolbox and then we discuss our findings with pMatlab.

We explored three different methods of parallelization: pMATLAB, Parallel Computing Toolbox `parfor` statements, and Parallel Computing Toolbox `spmd` statements. The MATLAB

Parallel Computing Toolbox provides the natively interpreted `parfor` and `spmd` commands for parallel computing. Native interpretation is advantageous because the code runs without substantial modification and is easy to implement; thus, it is a significant architectural boon in terms of maintainability. On the other hand, pMatlab parallel implementation requires significant modification. The modification necessary is shown in Appendix D, which consists of example code for the three parallelization techniques as well as a single threaded version for reference. Despite the drawback of modification, our tests found pMatlab to be a superior method because we were unable to realize major increases in performance with the MATLAB Parallel Computing Toolbox as observed in the data from Table 4.

Our analysis of the Parallel Computing Toolbox revealed the primary cause of low performance gains with this toolbox. Through profiling, we determined the way in which the toolbox distributes and aggregates data to be incompatible with our data structure. Communication between threads required such a significant amount of time that even with eight parallel threads, the non-parallel code still preformed the simulation with little performance decrease. A further factor in understanding the results in Table 4 is for each additional worker, memory usage increases linearly. This in turn mandates significantly more RAM be installed in computers simulating large oceans. In addition, we found that the Mathworks documentation and analysis tools were severely lacking for the Parallel Computing Toolbox. An example of this deficiency was the lack of documentation related to communication overhead. Inversely, the onsite help available with pMatlab proved to be invaluable in the development and execution of our model because the LLGrid staff were able to work through our issues and propose solutions quickly.

Table 4. Parallel Computing Toolbox Speedups. Computers used are those denoted in Table 6. The code snippets in Appendix D were used in this test.

	Single Threaded	**`parfor`**	**`spmd`**
Computer 1	72.02 (s)	70.54 (s)	92.46 (s)
	Speedup	1.02x	0.77x
Computer 2	21.16 (s)	12.89 (s)	18.57 (s)
	Speedup	1.64x	1.14x

Parallelization of the loops within the simulation allowed us to realize speedups of up to 60 times with 64 processors, a result close to the theoretical limit. These gains were realized through the use of pMatlab on the LLGrid. Figure 106 shows the dependence of processing time of the ocean generation loop found in Appendix D as a function of the number of LLGrid processors available for 4, 8, 16, 32, and 64 processors. We could not measure processing time for fewer processors due to memory constraints. Figure 107 shows a plot of speedup vs. number of processors for the data shown in Figure 106 with linear extrapolation of the speedup for the one and two processor cases. Both figures were generated using the parameters specified in Table 5. The speedup and estimated time were based on the projection of time to generate an ocean on one processor.

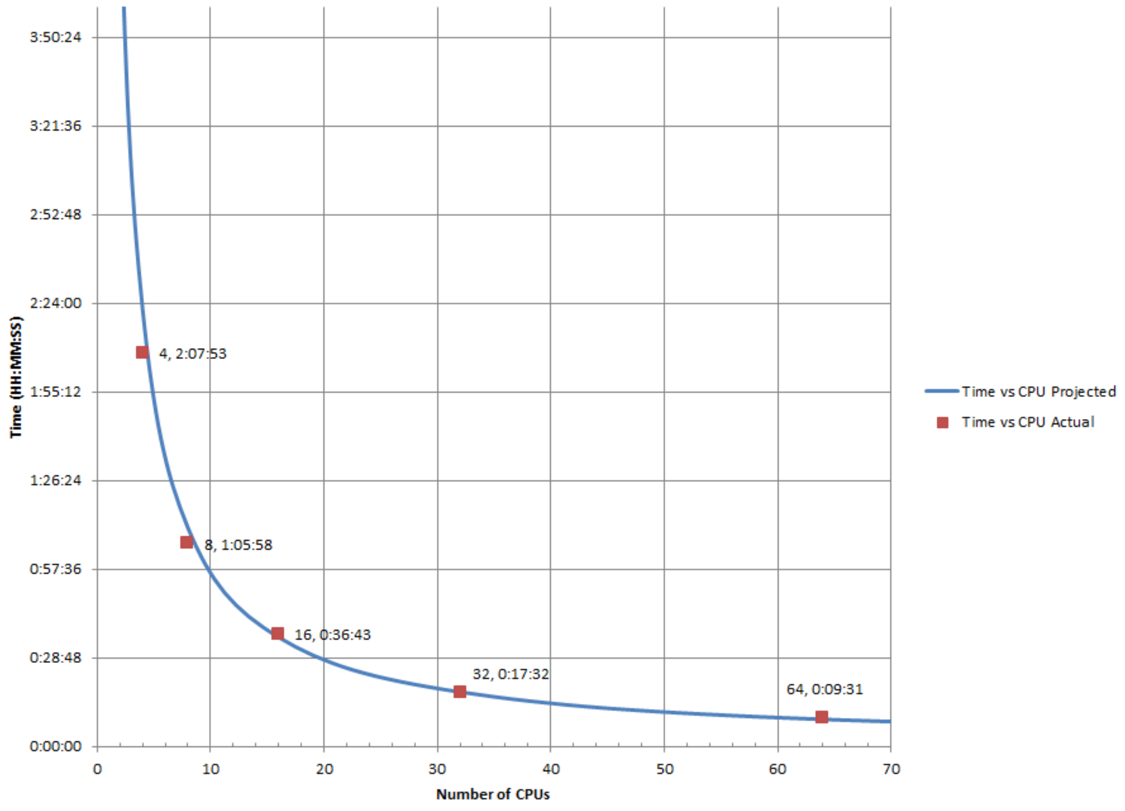


Figure 106. Processing time vs. Number of CPUs.

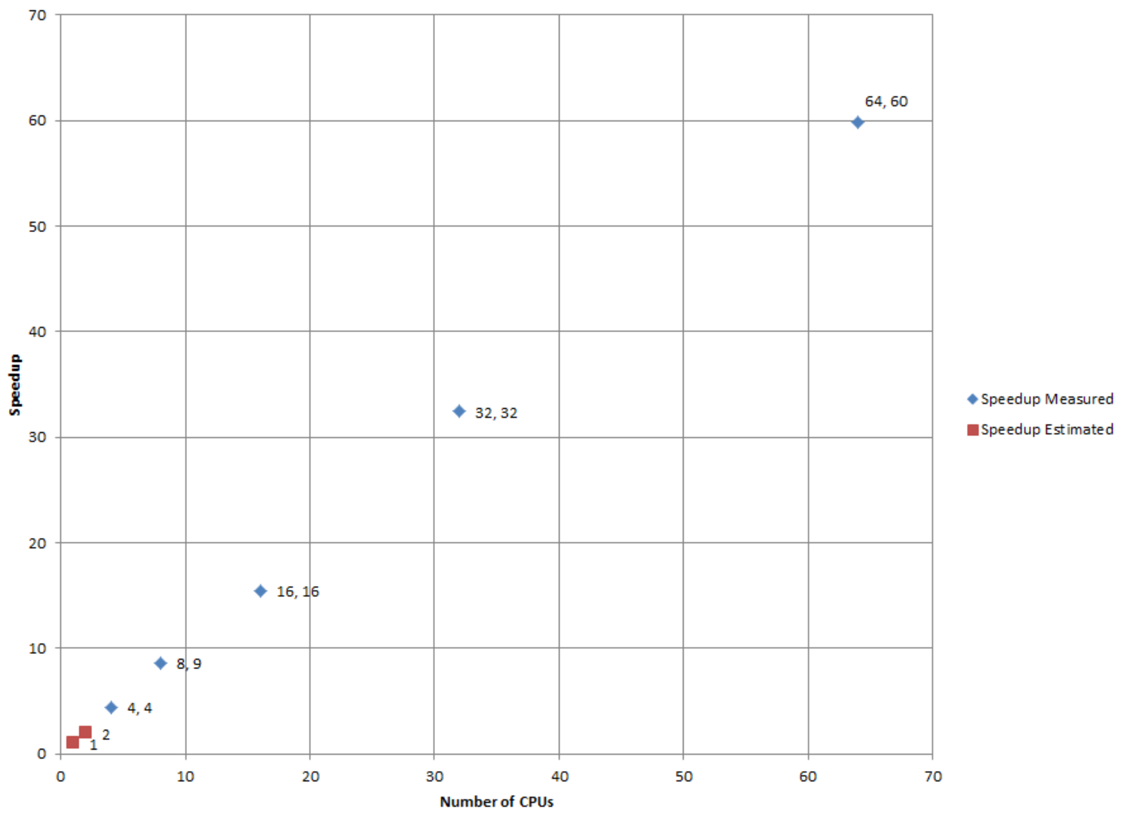


Figure 107. Speedup vs. Number of CPUs of the generation of ocean heights. The speedup was not measured for less than four processors due to memory constraints. Values for one or two processors are estimated. The number pairs near the data points are of the form (number of CPUs, speedup).

Table 5. Parameters used in testing the LLGrid speedup

Parameter	Value
Ocean propagation direction	Upsea
Radar Offset	1000 m
Ocean Length	10000 m
Spatial Ocean Sampling	1 m
Windspeeds	[13 18.5 24] knots
Center Requency	10e3 MHz
Bandwidth	100 MHz
PRF	2000 Hz
Grazing Angle	5°
Simulation Time	10 (s)
Compute Method	pMATLAB
Use LLGrid	Yes
Number of CPUs	Varies (4-64)

Comparing the LLGrid with 64 available nodes to the LLGrid with only one available node resulted in impressive speedups; however, it is important to compare the LLGrid to the workstations available at MIT Lincoln Laboratory. The sample workstations in Table 6 and the parameters in Table 5 were used to benchmark the speedup provided by the LLGrid as compared to the workstations. The parameters were the same except Compute Method on the workstations (set to Normal) and the Number of CPUs on the LLGrid (set to 64). The results are shown in Table 7.

Table 6. Test Computer Configurations

Computer 1	
Computer Model Number	Dell Precision WorkStation 490
Operating System	Microsoft Windows 7 Enterprise SP1
Processor	Intel Xeon 5160 @ 3.00 GHz
Logical Processors	2
Memory	8192MB
MATLAB Version	R2012a (7.14.0.739) win64
Computer 2	
System Model	Dell Precision WorkStation T5500
Operating System	Windows
Processor	2x Intel Xeon X5677 @3.47GHz
Logical Processors	8
Memory	12288 MB
MATLAB Version	R2012a (7.14.0.739) win64
Computer 3	
System Model	Dell Precision WorkStation T5400
Operating System	CentOS release 5.8 (Final)
Kernel Version	2.6.18-308.13.1.e15
Processor	2x Intel Xeon X5460 @3.16GHz
Logical Processors	8
Memory	8982 MB
MATLAB Version	R2012a (7.14.0.739) glnxa64

Table 7. Compute time for different computer configurations

	Time (s)	**Speedup**
Computer 1	6039.31	1
Computer 2	2066.605	2.92
Computer 3	1683.058	3.59
LLGrid (64 workers)	772.753	7.82

Computer 1 was the slowest of our test systems and was used as a speedup reference. Table 7 shows that the LLGrid was 7.82 times faster than Computer 1. A speedup of 7.82 is moderate; however, this speedup is for the entire simulation where not only the ocean is generated, but the radar returns are generated and calibrated. Part of the entire simulation is executed on the computer submitting the job which may reduce the overall speedup. Due to part of the task running on the host computer, speedup may be limited by the serial portions masking the speedup provided by the grid. If, in Figure 108, task A is the serial portion to be completed on the desktop while task B is the portion executed on the LLGrid it is apparent why the speedup appears to be lacking. Performance gains should be more apparent with larger models. Additionally, as more logical processors are available on a workstation, the gains provided by pMatlab on the LLGrid diminish rapidly. These diminishing gains were unexpected.

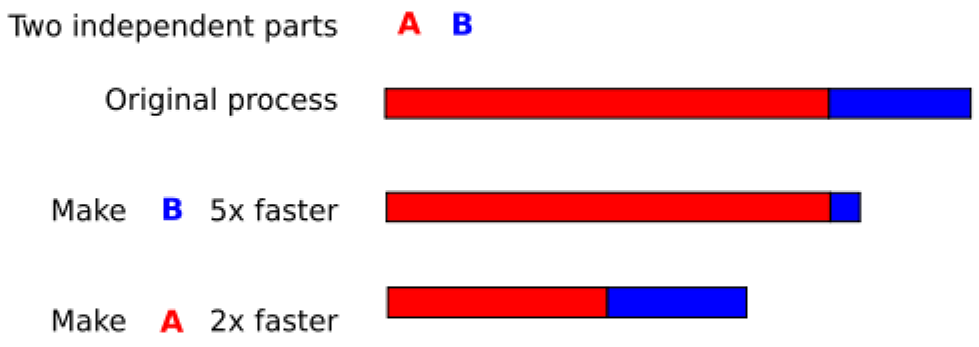


Figure 108. Speedup gained by changing different parts of an algorithm. (Wikipedia, 2012)

We determined three explanations for these diminishing gains. First, it is important to consider that unless MATLAB is launched with the parameter `-singleCompThread`, some of the MATLAB functions support parallelization natively (Mathworks). Our simulation makes extensive use of the function `bsxfun` which is one of the functions which MATLAB natively parallelizes. This native parallelization allows MATLAB to realize significant performance increases without any forethought on the designer's part. Second, the quasi-serial MATLAB solution does not suffer from the same communication overhead as the LLGrid. Specifically, the LLGrid must write all variables to a network disk in between steps, while the serial solution maintains the variables in the RAM. Third, the processing nodes of the LLGrid are approaching nine years of age. By virtue of Moore's law, a modern processor is approximately twenty-three times faster than an LLGrid node. From these results, we conclude that it is best to use the LLGrid when executing the simulation from older computers or on large data sets. However, new high performance workstations with at least eight logical processors may outperform the LLGrid. If LLGrid used modern processors and an updated network infrastructure, then speedups predicted by our model could be realized. In fact, Lincoln Laboratory has plans to upgrade the LLGrid within the next two years.

Future extensions and improvements include evaluation of Mathworks and third party parallelization toolboxes for the addition of general-purpose computing on graphics processing units (GPGPU) and conversion of MATLAB code to MEX code written in C or C++. In fact, during research on multithreaded MATLAB functions, we found many of the functions we wished to preform, i.e. multi-dimensional variable expansion and ocean generation, already have commercial or documented solutions for GPGPU implementations. These technologies leverage the stream processing provided by a graphics processing unit which can provide up to 3,072 (Nvidia) processing nodes per GPU.

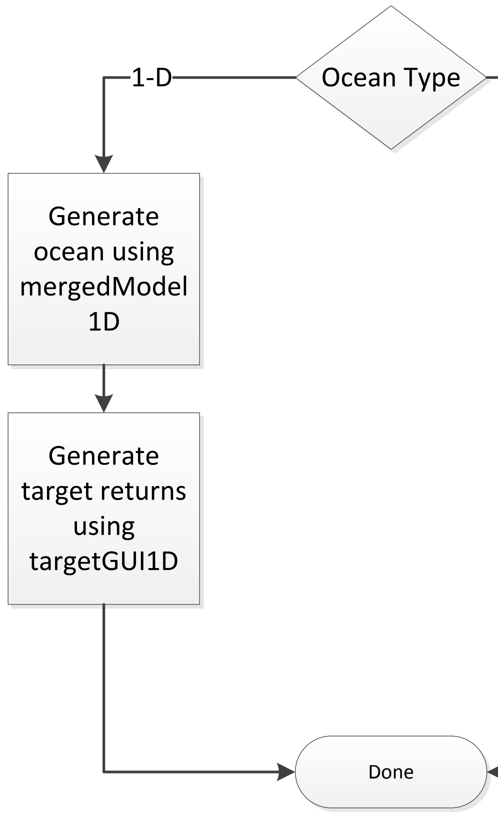
6 Graphical User Interface

6.1 Background

A graphical user interface (GUI) exists primarily to enhance the usability of software. MATLAB has extensive support for quickly producing usable GUIs in the form of the GUIDE, a GUI development environment. A useful function that can be performed by GUIs is input validation. Input validation is the process of checking the range of values input to a program for determining whether supplied values will crash the program or to check if the supplied values fall within the range valid for a particular problem.

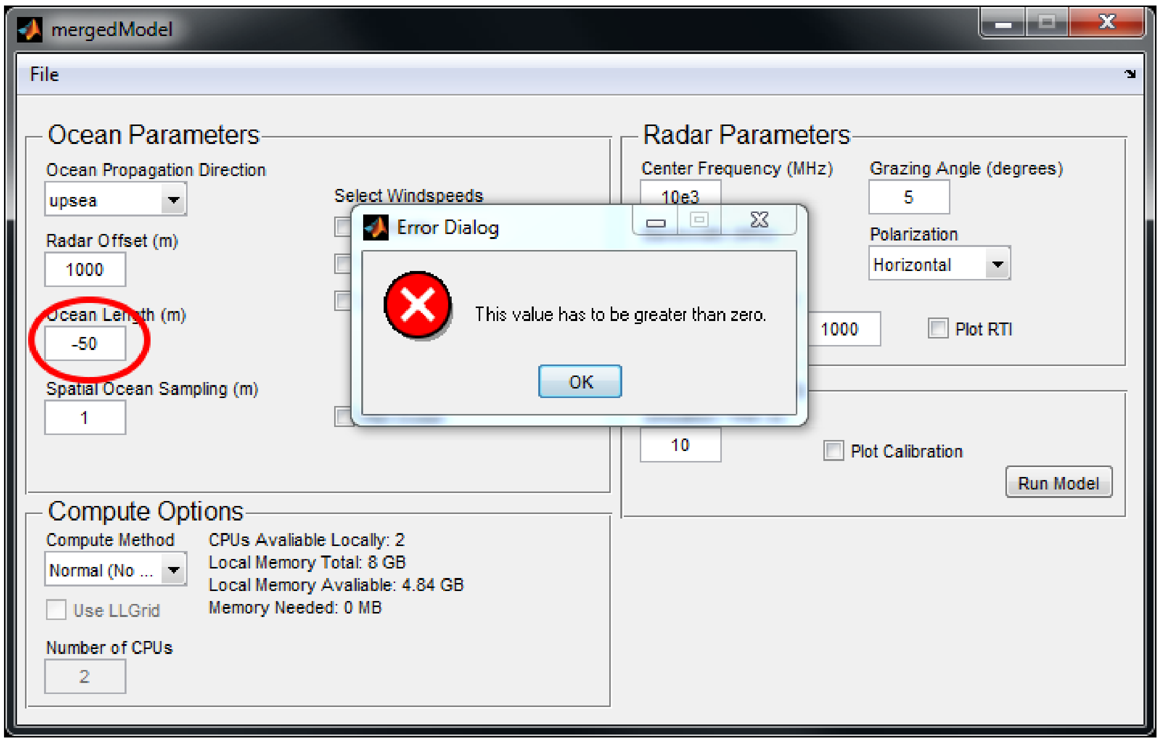
6.2 Methods and Results

The graphical user interface was designed using GUIDE to allow for easy use of the simulations. Four GUIs were developed: `mergedModel1D`, `targetGUI1D`, `mergedModel2D`, and `targetGUI2D`. The intended use of these interfaces is outlined in Figure 109. The GUIs disallow input values outside of a correct range i.e. a negative ocean length as shown in Figure 110. Additionally, the GUIs selectively enable and disable items depending on whether the items are valid for the given situation, as shown in Figure 111. The GUIs also have the ability to determine the amount of memory necessary to generate the given models and compare the needed memory to the available memory. Comparing the needed memory and available memory is then used to warn the user when the model would fail due to memory limitations. Memory limit warnings work for both the locally submitted jobs as well as the LLGrid jobs. Table 8 outlines the input bounds and notes for the input fields in the `mergedModel1D` while Figure 110 and Figure 111 depict the GUI.



Ocean Type
1-D
2-D
Generate ocean using mergedModel 1D
Generate ocean using mergedModel 2D
Generate target returns using targetGUI1D
Generate target returns using targetGUI2D
Done

Figure 109. GUI use flow diagram.



Ocean Parameters

Ocean Propagation Direction: upsea

Radar Offset (m): 1000

Ocean Length (m): -50

Spatial Ocean Sampling (m): 1

Select Windspeeds

This value has to be greater than zero.

Radar Parameters

Center Frequency (MHz): 10e3

Grazing Angle (degrees): 5

Polarization: Horizontal

1000

Plot RTI

10

Plot Calibration

Compute Options

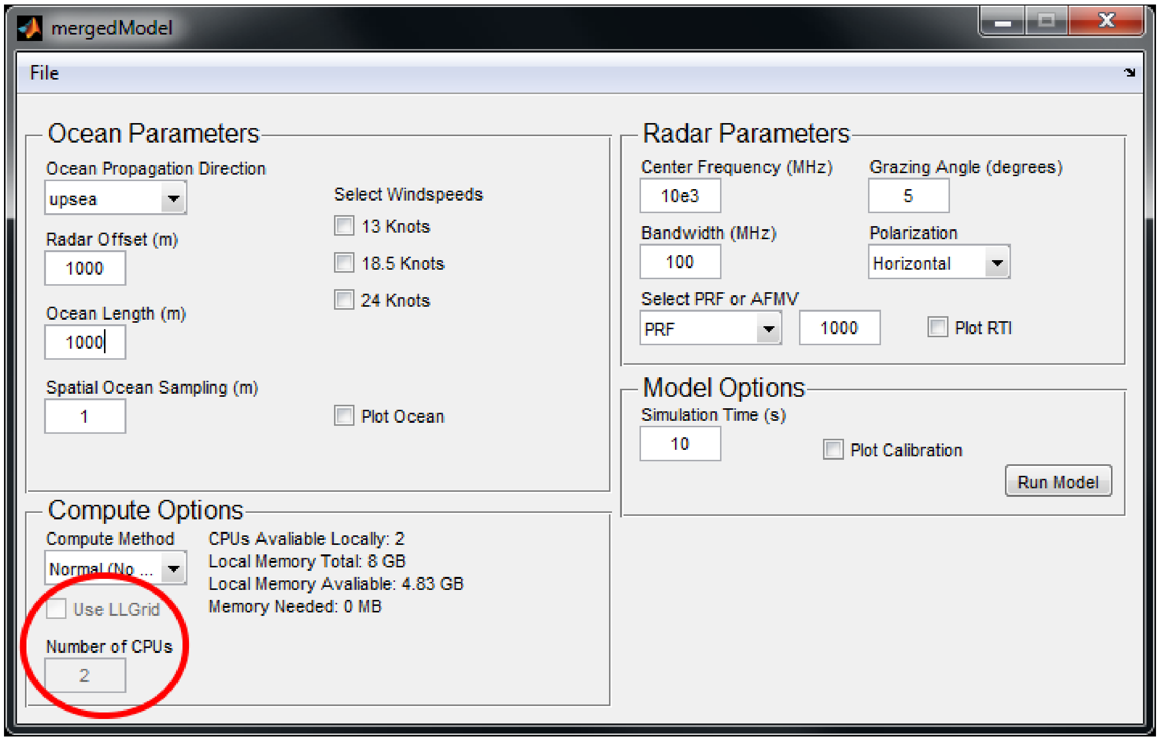
Compute Method: Normal (No ...

Use LLGrid

Number of CPUs: 2

CPUs Avaliable Locally: 2
Local Memory Total: 8 GB
Local Memory Avaliable: 4.84 GB
Memory Needed: 0 MB

Figure 110. Error resulting from out of bounds input.


mergedModel

Ocean Parameters

Ocean Propagation Direction: upsea

Radar Offset (m): 1000

Ocean Length (m): 1000

Spatial Ocean Sampling (m): 1

Select Windspeeds
- [ ] 13 Knots
- [ ] 18.5 Knots
- [ ] 24 Knots

- [ ] Plot Ocean

Radar Parameters

Center Frequency (MHz): 10e3

Grazing Angle (degrees): 5

Bandwidth (MHz): 100

Polarization: Horizontal

Select PRF or AFMV: PRF 1000

- [ ] Plot RTI

Model Options

Simulation Time (s): 10

- [ ] Plot Calibration

Compute Options

Compute Method: Normal (No ...

- [ ] Use LLGrid

Number of CPUs: 2

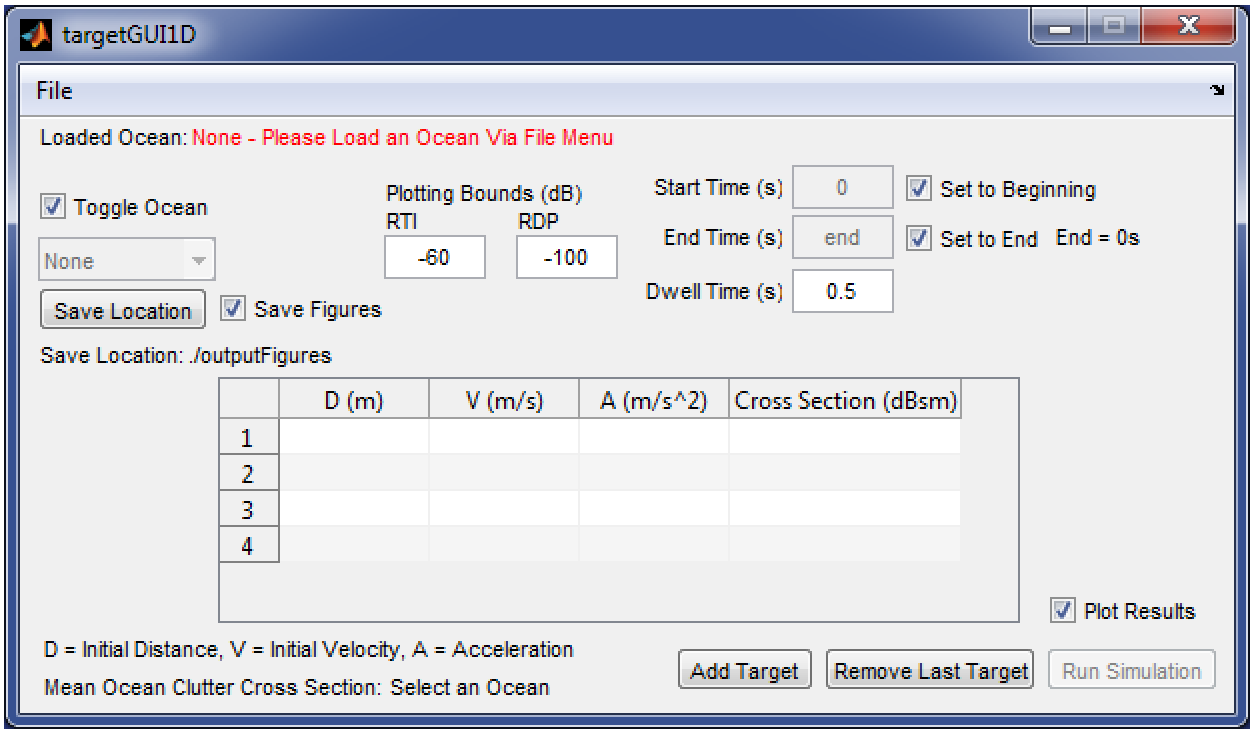
CPUs Avaliable Locally: 2
Local Memory Total: 8 GB
Local Memory Avaliable: 4.83 GB
Memory Needed: 0 MB

Figure 111. Selectively disabled input fields. Here, the “Use LLGrid” checkbox and “Number of CPUs” field are disabled because the “Compute Method” is not set to “pMatlab.” This interface is used to generate 1-D oceans and calibrated range time intensity data.

Table 8. Available `mergedModel` GUI input variable bounds and notes.

Property	**Bounds**	**Notes**
Radar Offset	$0 < value$	
Ocean Length	$0 < value$	
Spatial Ocean Sampling	$0 < value < Ocean\ Length$	
Center Frequency	$0 < value$	
Bandwidth	$0 < value$	
Grazing Angle	$0 < value < 45$	
PRF/AFMR	$0 < value$	
Simulation Time	$0 < value$	
Number of CPUs	$0 < value$	If LLGrid is not selected the upper bound is the number of CPUs Available Locally. Requires pMATLAB to be selected under Compute Method to be enabled.
Use LLGrid	N/A	Requires pMATLAB to be selected under Compute Method in order to be enabled.
Plot Ocean	N/A	Requires Normal to be selected under Compute Method in order to be enabled.
Plot RTI	N/A	Requires Normal to be selected under Compute Method in order to be enabled.

Once an ocean or ocean set has been created, the user may add point targets with velocity and acceleration via the `targetGUI1D`. The `targetGUI1D` is shown in Figure 112. To use the GUI, users first select the ocean generated by `mergedModel1D` and select `Load Saved Ocean` under the `File` menu. Additionally, the user may select to save and load targets from the `File` menu as illustrated in Figure 113. Once an ocean has been loaded, the oceans available are automatically populated into the ocean select menu presented in Figure 114. Table 9 lists the input bounds and notes for the `targetGUI1D`.


targetGUI1D

File

Loaded Ocean: None - Please Load an Ocean Via File Menu

☑ Toggle Ocean

None

Save Location ☑ Save Figures

Save Location: ./outputFigures

Plotting Bounds (dB)

RTI	RDP
-60	-100

Start Time (s)	0	☑ Set to Beginning	
End Time (s)	end	☑ Set to End	End = 0s
Dwell Time (s)	0.5		

	D (m)	V (m/s)	A (m/s^2)	Cross Section (dBsm)
1				
2				
3				
4				

☑ Plot Results

D = Initial Distance, V = Initial Velocity, A = Acceleration

Mean Ocean Clutter Cross Section: Select an Ocean

Add Target | Remove Last Target | Run Simulation

Figure 112. Depiction of `targetGUI1D`. This interface consumes a previously generated 1-D ocean and returns range time intensity and range Doppler profile data and plots. Targets which move along the ocean may be added.

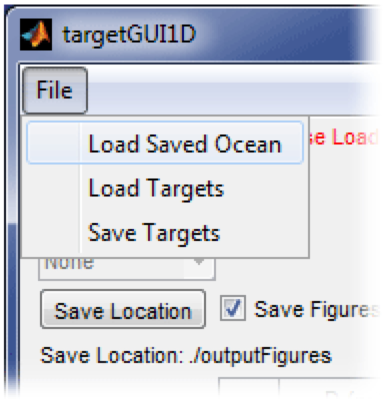


Figure 113. Options under the `File` menu for the `targetGUIs`

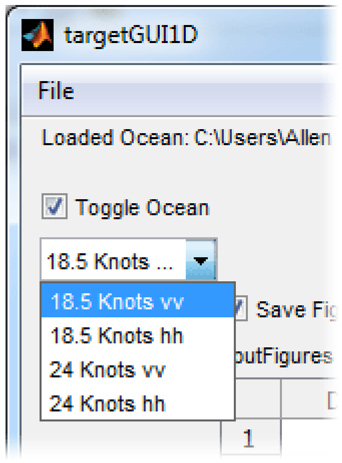
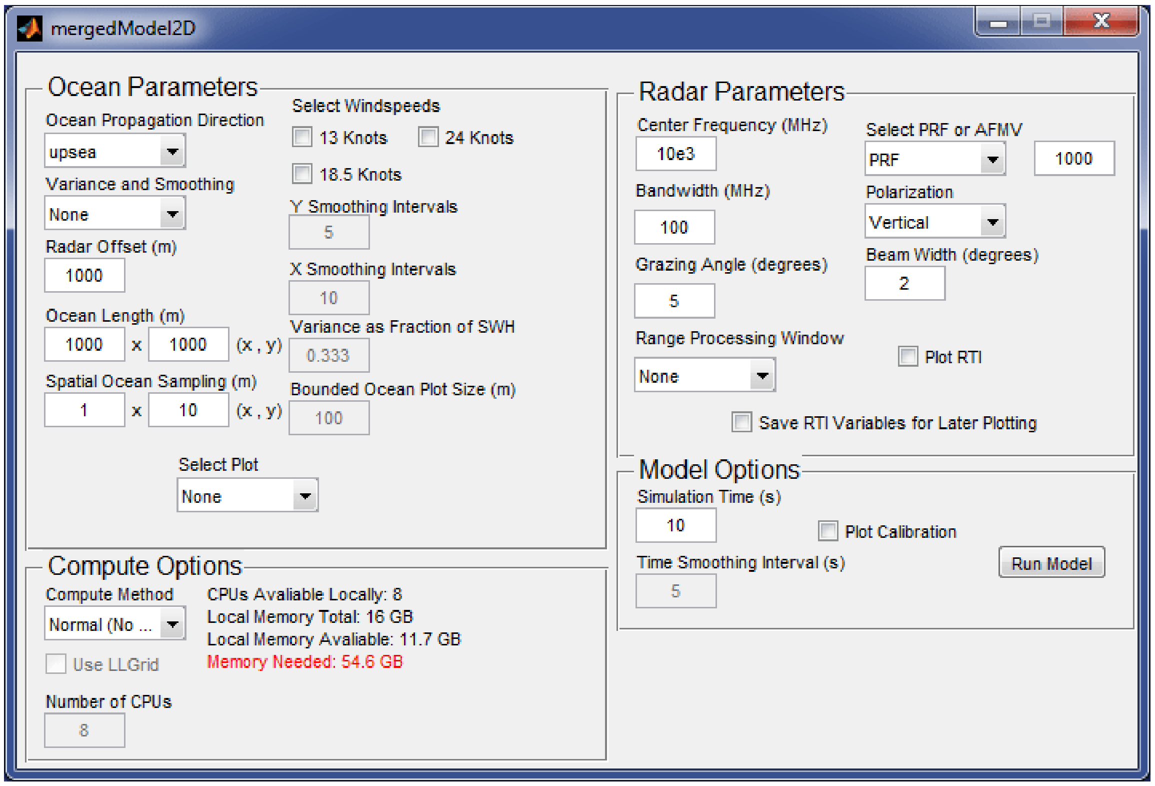


Figure 114. Auto-populated ocean select menu.

Table 9. Available `targetGUI1D` input variable bounds and notes.

Property	Bounds	Notes
Plotting Bounds RTI and RDP	$0 < value$	Requires Plot Results to be selected.
Start Time	$0 \leq value < End\ Time$	
End Time	$Start\ Time < value < Max\ Time$	
Dwell Time	$0 < value \leq End\ Time$	
Bandwidth	$0 < value$	

The `mergedModel2D` and the `targetGUI2D` GUI were simply extensions of their one-dimensional counterparts. The `mergedModel2D` is shown in Figure 115 and `targetGUI2D` is displayed in Figure 116. Table 10 outlines the bounds and notes for the fields found in `mergedModel2D`.


mergedModel2D

Ocean Parameters

Ocean Propagation Direction: upsea

Variance and Smoothing: None

Radar Offset (m): 1000

Ocean Length (m): 1000 x 1000 (x , y)

Spatial Ocean Sampling (m): 1 x 10 (x , y)

Select Windspeeds: ☐ 13 Knots ☐ 24 Knots ☐ 18.5 Knots

Y Smoothing Intervals: 5

X Smoothing Intervals: 10

Variance as Fraction of SWH: 0.333

Bounded Ocean Plot Size (m): 100

Select Plot: None

Compute Options

Compute Method: Normal (No ...

☐ Use LLGrid

Number of CPUs: 8

CPUs Avaliable Locally: 8
Local Memory Total: 16 GB
Local Memory Avaliable: 11.7 GB
Memory Needed: 54.6 GB

Radar Parameters

Center Frequency (MHz): 10e3

Bandwidth (MHz): 100

Grazing Angle (degrees): 5

Range Processing Window: None

Select PRF or AFMV: PRF 1000

Polarization: Vertical

Beam Width (degrees): 2

☐ Plot RTI

☐ Save RTI Variables for Later Plotting

Model Options

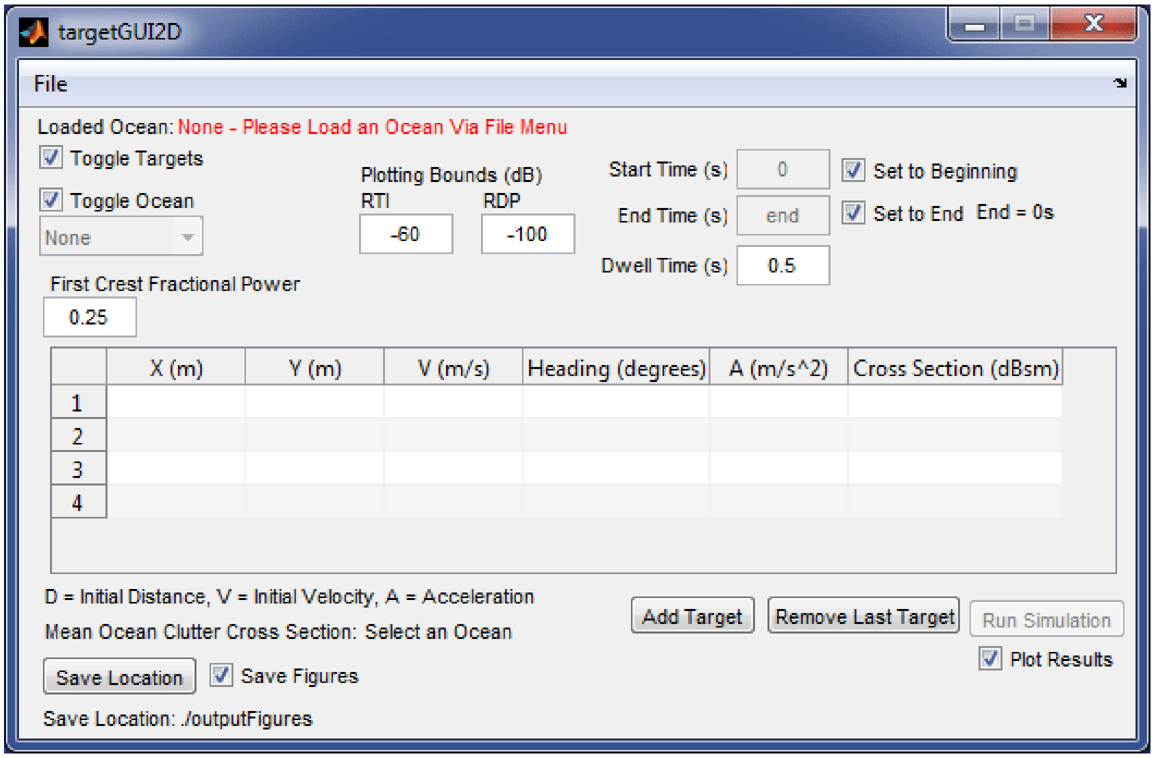
Simulation Time (s): 10

Time Smoothing Interval (s): 5

☐ Plot Calibration

Run Model

Figure 115. The `mergedModel2D` user interface. This interface is used to generate 2-D oceans and calibrated range time intensity data.


targetGUI2D

File

Loaded Ocean: None - Please Load an Ocean Via File Menu

☑ Toggle Targets

☑ Toggle Ocean

None

First Crest Fractional Power: 0.25

Plotting Bounds (dB): RTI -60, RDP -100

Start Time (s): 0 ☑ Set to Beginning

End Time (s): end ☑ Set to End End = 0s

Dwell Time (s): 0.5

	X (m)	Y (m)	V (m/s)	Heading (degrees)	A (m/s^2)	Cross Section (dBsm)
1						
2						
3						
4						

D = Initial Distance, V = Initial Velocity, A = Acceleration

Mean Ocean Clutter Cross Section: Select an Ocean

Add Target | Remove Last Target | Run Simulation

☑ Plot Results

Save Location ☑ Save Figures

Save Location: ./outputFigures

Figure 116. The `targetGUI2D` user interface. This interface consumes a previously generated 2-D ocean and returns range time intensity and range Doppler profile data and plots. Targets which move around the ocean may be added as well as boat wake.

Table 10. Input variable bounds and notes for `mergedModel2D` in addition to those found in Table 8.

Property	**Bounds**	**Notes**
Ocean Length Y	$0 < value$	Requires Plot Results to be selected.
Spatial Ocean Sampling Y	$0 \leq value < Ocean\ Length\ Y$	
Select Plot	N/A	This menu is active when the *Compute Method* is set to "Normal".
Y Smoothing Intervals	$0 < value$	This field is active when "Variance with Smoothing" is selected in the *Variance and Smoothing* menu.
X Smoothing Intervals	$0 < value$	This field is active when "Variance with Smoothing" is selected in the *Variance and Smoothing* menu.
Variance as Fraction of SWH	$0 < value$	This field is active when "Variance" or "Variance with Smoothing" is selected in the *Variance and Smoothing* menu.
Bounded Ocean Plot Size	$0 < value < smallest\ dimension\ of\ Ocean\ Length$	This field is active when the *Select Plot* menu is set to "Bounded Ocean"
Time Smoothing Interval	$0 < value$	This field is active when "Variance with Smoothing" is selected in the *Variance and Smoothing* menu.

These graphical user interfaces provide the user with an easy to use interface. They provide warnings when improper variables are input and they alert the user to proper ranges for the values.

7 Conclusions and Future Extensions

The goal of this project was to develop a simulation capable of generating the radar response of small boats and realistic ocean clutter. The objective of this project was to enable the development and test of detection algorithms for small boats on the ocean surface. After better detection algorithms are established, the threat to U.S. naval assets from small boats and the amount of drug-running, smuggling, and piracy can be reduced. In order to accomplish our goal and objective, we produced each of our deliverables:

- Non-scanning one-dimensional ocean radar scattering simulation: A non-scanning chirp radar simulation capable of range and Doppler processing was integrated with a one-dimensional ocean model defined by sum of sinusoids defined in amplitude and frequency by the Pierson-Moskowitz power spectral density. The ocean height was scaled according to the Beaufort scale and the radar returns of the ocean surface were calibrated to publicly available data.
- Non-scanning quasi-two-dimensional ocean radar scattering simulation: The Non-scanning one-dimensional ocean radar scattering simulation was extended to produce a more capable model. Specifically, a quasi-two-dimensional ocean model was developed via replications of the one-dimensional ocean. The two-dimensional ocean has optional Gaussian variance and smoothing. Targets on this ocean support cross-sea movement and optional wake.
- Phased array radar scattering simulation: A computationally efficient phased array radar simulation was developed and integrated to include targets with wake. The phased array enables range, velocity, and azimuth processing.
- Graphical user interface: An easy to use graphical interface was developed for the models. The interfaces provide easy access to all of the input parameters and performs input validation.
- Parallelization: pMATLAB was leveraged for parallel computation using the LLGrid. Within the LLGrid, speedups of up to 60 times were observed. Moderate overall performance gains were recorded, but are limited to the hardware of the LLGrid.

Our product meets the customer requirements of MIT Lincoln Laboratory Group 105 to be robust, easy to use, accurate, and contain strong documentation.

While developing our product, areas of improvement were recognized. One simple change that could be made to our model that would increase accuracy is to use more calibration data. In the project, we used the small ocean scattering data set compiled by Long. We also recognized that the calibration algorithm discussed in Section 2.2.3 could be improved to only require one mode shift. Also, the effects of polarization on radar returns could be modeled rather than directly calibrating to known data (Section 2.3). In addition, we recognized several major future extensions that would help to answer questions that arose during this work. The staff at MIT Lincoln Laboratory has also expressed interest in these extensions. We have organized these extensions by importance:

1. Non-coherent integration: A common first step toward boat detection is non-coherent integration, a process which sums the RTI or RDP over dwell times. Because the power of the radar return from the boat is more constant in time than the power of the ocean clutter, the summation over dwell times substantially reduces the clutter from the ocean.

2. Lagrangian mechanics and material derivative model: Our ocean model is a surface model such that each modeled point target is spatially stationary in x and y and varies in height. Thus, the Doppler returns inaccurately measure the velocity of height change. A more accurate model would consider a particle which can vary its x, y, and height values. This more accurate model may is possible using the material derivative.

3. Phased array radar quasi-two-dimensional ocean scattering simulation: The phased array model of Chapter 4 and the quasi-two-dimensional ocean model of Section 3.2.1 can be integrated to model the ocean scattering that a phased array radar might see.

4. Fluctuating target cross section (Swerling model): In our model, boats are modeled as point targets with a constant cross section. A more accurate model may include boats with cross sections that vary in time.

5. Two-dimensional phased array radar quasi-two-dimensional ocean scattering simulation: Our one-dimensional phased array radar simulation (Chapter 4) can be extended to a two-dimensional phased array radar simulation that is capable of resolving targets in range, velocity, azimuth, and elevation. This two-dimensional simulation can be integrated with the quasi-two-dimensional ocean to obtain the radar returns of the ocean surface.

6. Two-dimensional phased array radar cross sea ocean scattering simulation: The two-dimensional phased array simulation of (5) can be integrated with a more complex two-dimensional ocean model, such as the Philips model, which more accurately represents the propagation of the ocean.

7. GPGPU parallel implementation: Parallel processing on graphical processing units might realize further parallel speedups.

References

[Image of one-dimensional phased array]. (2012). Retrieved August 22, 2012, from http://sitelife.aviationweek.com/ver1.0/Content/images/store/13/7/7d634054-f899-41a1-b7ca-552c8df19915.Full.jpg

[Image of semi-submersible vessel]. (2012). Retrieved August 22, 2012, from http://covertshores.blogspot.com

[Image of USS Cole]. (2012). Retrieved August 22, 2012, from http://en.wikipedia.org/wiki/file:INTEL-COGNITIVE-Cole.jpg

The Beaufort Wind Scale. (2007, 03 01). Retrieved 07 25, 2012, from NOAA/National Weather Service Hydrometeorological Prediction Center: http://www.hpc.ncep.noaa.gov/html/beaufort.shtml

Wikipedia. (2012).

Amdahl, G. (1967). Validity of the single-processor approach to achieving large scale computing capabilities. *AFIPS Conference Proceedings* (pp. 483-485). Atlantic City, N.J.: AFIPS Press.

Byun, C. (2012). MIT Lincoln Laboratory pMATLAB Tutorial.

Currie, I. (1974). *Fundamental Mechanics of Fluids.* New York: McGraw-Hill.

Ewell, Tuley, & Horne. (1984). Temportal and Spatial Behavior of High Resolution Sea Clutter. *IEEE National Radar Conference*, 100-104.

Griffiths, D. (1999). *Introduction to Electrodynamics* (3rd ed.). Upper Saddle River: Prentice-Hall.

Harwood, A. (2003, 10 22). *Embarrassingly parallel*. Retrieved 10 6, 2012, from Parallel algorithms: http://ww2.cs.mu.oz.au/498/notes/node40.html

Hennings, I., Romeiser, R., Alpers, W., & Viola, A. (1999). Radar imaging of Kelvin arms of ship wakes. *International Journal of Remote Sensing, 20*(13), 2519-2543.

Jackson, C. R., & John, A. R. (2004). *Synthetic Apeture Radar Marine User's Manual.* Washington: U.S. Department of Commerce.

Kanevsky, M. B. (2009). *Radar Imaging of the Ocean Waves.* Oxford: Elsevier.

Kinsman, B. (2002). *Wind Waves: Their Generation and Propagation on the Ocean Surface.* Englewood Cliffs, NJ: Dover Publications.

Lathi, B. (2005). *Linear Systems and Signals* (2nd ed.). New York: Oxford University Press.

Levanon, N. (1988). *Radar Principles.* New York: John Wiley & Sons.

Long, M. (2001). *Radar Reflectivity of Land and Sea.* Boston: Artech House.

Mathworks. (n.d.). *Which MATLAB functions benefit from multithreaded computation?* Retrieved September 21, 2012, from Mathworks: http://www.mathworks.com/support/solutions/en/data/1-4PG4AN/?solution=1-4PG4AN

Mitchell, J. L. (2005). *Real-Time Synthesis and Rendering of Ocean Water.* AMD.

Monzingo, R., & Miller, T. (1980). *Introduction to Adaptive Arrays.* New York: John Wiley & Sons.

"Particles and Waves". (n.d.). Retrieved July 20, 2012, from Goshen College: http://www.goshen.edu/physix/204/gco/2slit.php

National Oceanic and Atmospheric Administration. (2011, 11 28). *How are significant wave height, dominant period, average period, and wave steepness calculated?* Retrieved 08 15, 2012, from National Data Buoy Center: http://www.ndbc.noaa.gov/wavecalc.shtml

Nvidia. (n.d.). *Tesla GPU Computing Solutions for Servers*. Retrieved September 21, 2012, from Tesla Solutions for Servers: http://www.nvidia.com/object/tesla-servers.html

Papoulis, A. (2002). *Probability, Random Variables, and Stochastic Processes.* New York City: McGraw-Hill.

Rihaczek, A. (1969). *Principles of High Resolution Radar.* New York: McGraw-Hill.

Skolnik. (2008). *Radar Handbook* (3rd ed.). New York: McGraw-Hill.

Skolnik, M. (2001). *Introduction to Radar Systems* (3rd ed.). New York: McGraw-Hill.

Toomay, J., & Hannen, P. (2004). *Radar Principles for the Non-specialist* (3rd ed.). Raleigh, NC: SciTech Publishing.

Tussendorf, J. (2001). *Simulating Ocean Water.* University of California San Diego.

Whitham, G. B. (1999). *Linear and Nonlinear Waves.* New York: Wiley.

Wolff, C. (2012). *Radar Principles*. Retrieved 8 30, 2012, from Radar Tutorial: http://www.radartutorial.eu/

Wowk, B. (2007). *Phased Array Optics*. Retrieved 9 1, 2012, from http://www.phased-array.com/1996-Book-Chapter.html

Appendix

Appendix A - Derivations

Appendix A.1 - Equation (2 - 25)

Let $C(\omega)$ be the Fourier domain representation of a chirp signal:

$$C(\omega) = u(\omega - \omega_1) - u(\omega - \omega_2)$$

where ω is angular frequency and u is the Heaviside step function.
Using the inverse Fourier transform,

$$c(t) = \mathcal{F}^{-1}[C(\omega)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} C(\omega)e^{j\omega t}dw$$

$$c(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}[u(\omega-\omega_1) - u(\omega-\omega_2)]\, e^{j\omega t}d\omega = \frac{1}{2\pi}\int_{\omega_1}^{\omega_2} e^{j\omega t}d\omega$$

$$c(t) = \frac{1}{2\pi}\left[\frac{e^{jwt}}{jt}\right]_{w=w_1}^{w_2} = \frac{1}{2\pi jt}\left(e^{jw_2t} - e^{jw_1t}\right)$$

We now use a change of variables:

$$\omega_1 = w_0 - \pi B \quad \leftrightarrow \quad f_1 = f_0 - \frac{B}{2}$$

$$\omega_2 = w_0 + \pi B \quad \leftrightarrow \quad f_2 = f_0 + \frac{B}{2}$$

Note that this form is directly related to the frequency representation as shown.

$$c(t) = \frac{1}{2\pi jt}\left(e^{j(\omega_o+\pi B)t} - e^{j(\omega_o-\pi B)t}\right) = \frac{e^{j\omega_0 t}}{2\pi jt}\left(e^{j(\pi B)t} - e^{j(-\pi B)t}\right)$$

Recall Euler's formula:

$$\sin(z) = \frac{e^{jz} - e^{-jz}}{2j}$$

We can rearrange our work above to:

$$c(t) = \frac{e^{j\omega_0 t}}{\pi t}\sin(\pi Bt) = \boxed{B \cdot e^{j\omega_0 t}\,\mathrm{sinc}(Bt)}$$

Appendix A.2 - Equation (2 - 39)

Let $V(t_s)$ be the slow time Doppler radar return, then

$$V(\omega_d) = \mathcal{F}[V(t_s)] = \int_{t_1}^{t_2} e^{-2jk_0vt_s}e^{-j\omega_d t}dt_s = e^{-jt_s(2k_0v+\omega_d)}dt_s$$

where ω_d is the Doppler angular frequency. Let $x = 2k_0v + \omega_d$

$$V(w) = \int_{t_1}^{t_2} e^{-jxt_s}dt_s = \left[\frac{e^{-jxt_s}}{-jx}\right]_{t_s=t_1}^{t_2} = \frac{1}{-jx}\left(e^{-jxt_2} - e^{-jxt1}\right)$$

We now use a charge of variables:

$$t_1 = t_0 - \frac{T_p}{2}$$

$$t_2 = t_0 + \frac{T_p}{2}$$

where t_0 is the center time and T_p is the pulse duration $\left(\frac{1}{prf}\right)$.

$$V(w) = \frac{1}{-jx}\left(e^{-jx\left(t_0+\frac{T_p}{2}\right)} - e^{-jx\left(t_0-\frac{T_p}{2}\right)}1\right) = \frac{e^{-jxt_0}}{-jx}\left(e^{-jx\left(\frac{T_p}{2}\right)} - e^{jx\left(\frac{T_p}{2}\right)}\right)$$

Using Euler's formula,

$$V(w) = 2e^{-jxt_0}\sin\left(\frac{xT_p}{2}\right) = \pi T e^{-jxt_0} sinc\left(\frac{xT_p}{2\pi}\right)$$

$$= \boxed{\pi T_p e^{-j(2k_0 v+\omega_d)t_0} sinc\left(\frac{(2k_0 v + \omega_d)T_p}{2\pi}\right)}$$

Appendix A.3 - Equation (4 - 15)

We start with Equation (4 - 14):

$$V[\omega, t_s] = W_\omega[\omega] W_{t_s}[t_s] e^{-2jk\cdot startR} \cdot$$

$$\sum_{\tau=1}^{N_T}\left(\sum_{n=1}^{N_E} W_m[n] e^{-j\left(kr_{n,\tau}[t_s]+n\,\pi\sin(\theta_s[t_s])\right)}\right)^2,$$

$$n = 1, 2, \dots, N_E$$

Here, we only show manipulations of the middle term: $W_\omega[\omega] W_{t_s}[t_s] e^{-2jk\cdot startR}$ and n are not modified:

We redefine $r_{n,\tau}[t_s] = \begin{cases} r_{0,\tau}[t_s] + r_{n,o}[t_s], & N_E\ odd \\ r_{0,\tau}[t_s] + r_{n,e}[t_s], & N_E\ even \end{cases}$

where

$$r_{n,o}[t_s] = \left(n - \left\lceil\frac{N_E}{2}\right\rceil\right) d\sin(\theta_s[t_s]),\text{ and}$$

$$r_{n,e}[t_s] = \left(n - \frac{N_E+1}{2}\right) d\sin[\theta_s]$$

and generically, $r_{n,\tau}[t_s] = r_{0,\tau}[t_s] + r_n[t_s]$

Where $r_{0,\tau}[t_s]$ is the distance to a target from the center of the phased array:

$$r_{0,\tau}[t_s] = \sqrt{h^2 + {y_\tau}^2 + {x_\tau}^2}$$

$$\sum_{\tau=1}^{\mathrm{N_T}}\left(\sum_{n=1}^{N_E} W_n[n]e^{-j\left(kr_{n,\tau}[t_s]+n\,\pi\,\sin(\theta_s[t_s])\right)}\right)^2$$

Substituting $r_{n,\tau}[t_s] = r_{0,\tau}[t_s] + \mathscr{r}_n[t_s]$:

$$\sum_{\tau=1}^{\mathrm{N_T}}\left(\sum_{n=1}^{N_E} W_n[n]e^{-j\left(k\left(r_{0,\tau}[t_s]+\mathscr{r}_n[t_s]\right)+n\,\pi\,\sin(\theta_s[t_s])\right)}\right)^2$$

Separating exponentials:

$$\sum_{\tau=1}^{\mathrm{N_T}}\left(\sum_{n=1}^{N_E} W_n[n]e^{-j(k\mathscr{r}_n[t_s]+n\,\pi\,\sin(\theta_s[t_s]))}e^{-jkr_{0,\tau}[t_s]}\right)^2$$

$e^{-jkr_{0,\tau}[t_s]}$ is not a function of n, so it can come out of the sum:

$$\sum_{\tau=1}^{\mathrm{N_T}}\left(e^{-jkr_{0,\tau}[t_s]}\sum_{n=1}^{N_E} W_n[n]e^{-j(k\mathscr{r}_n[t_s]+n\,\pi\,\sin(\theta_s[t_s]))}\right)^2$$

Distribute the square:

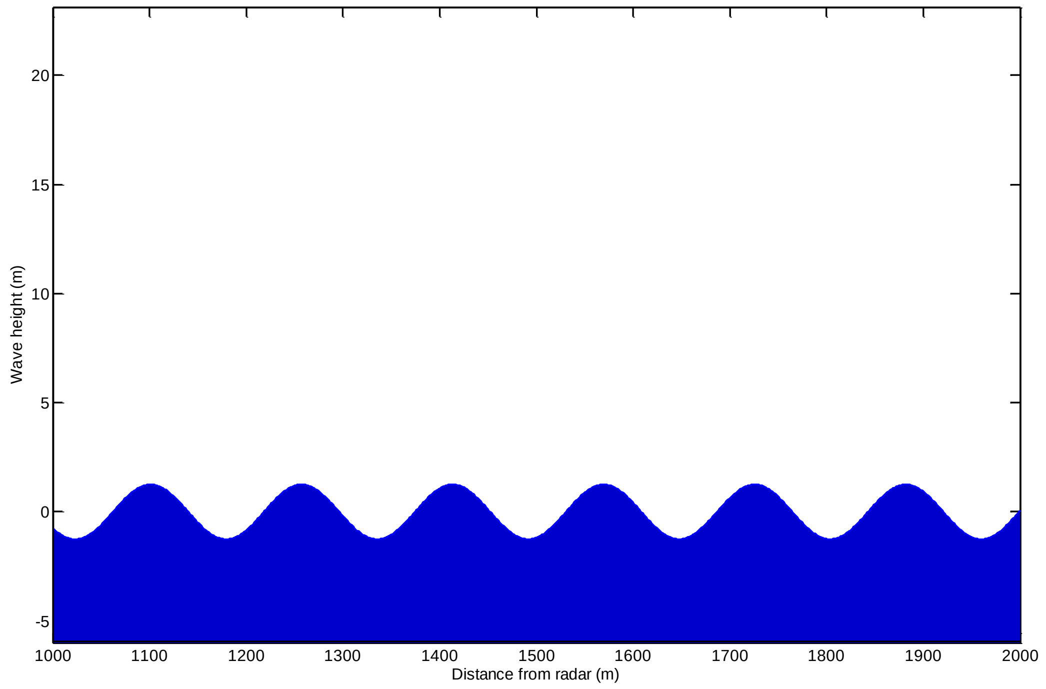
$$\sum_{\tau=1}^{\mathrm{N_T}}e^{-2jkr_{0,\tau}[t_s]}\left(\sum_{n=1}^{N_E} W_n[n]e^{-j(k\mathscr{r}_n[t_s]+n\,\pi\,\sin(\theta_s[t_s]))}\right)^2$$

The right summation is not a function of τ:

$$\left(\sum_{\tau=1}^{\mathrm{N_T}}e^{-2jkr_{0,\tau}[t_s]}\right)\left(\sum_{n=1}^{N_E} W_n[n]e^{-j(k\mathscr{r}_n[t_s]+n\,\pi\,\sin(\theta_s[t_s]))}\right)^2$$

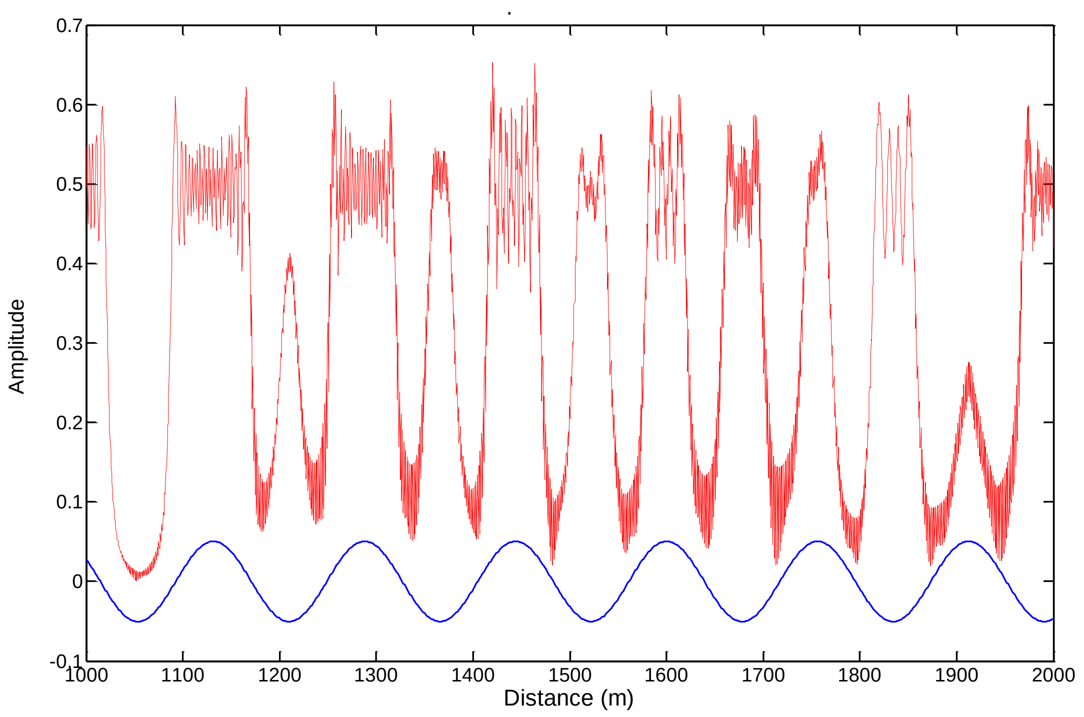
Appendix B - Slope Scaling Comparison

In this appendix, we show the difference between slope scaled RTI and un-scaled RTI. We used a single frequency ocean, shown below, for simplicity. The wind speed of this generated ocean was 24 knots, $f_0 = 10\ GHz, B = 100\ MHz, d_{alias-free} = 1000,$ grazing angle $= 5°$.

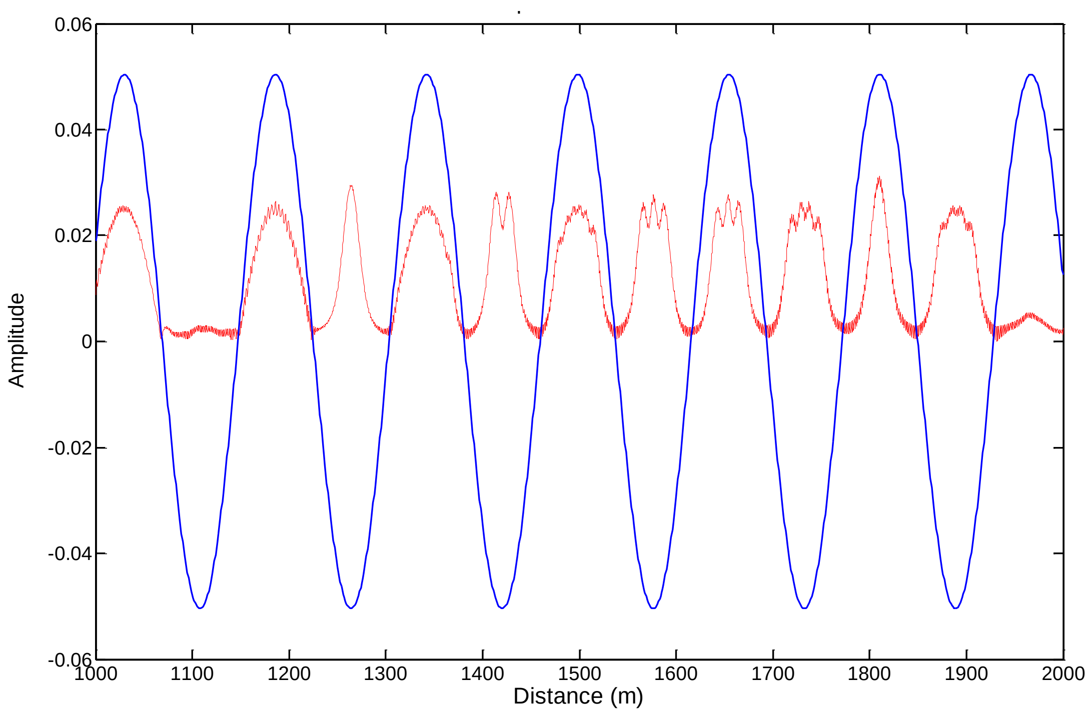


In the following two plots, the red line represents the power of the RTI in Watts vs. distance and the blue line represents the slope of the ocean vs. distance. Both of these plots represent the radar return strength and slope at $t = 0$. The un-slope-scaled plot is shown first, and the scaled plot follows:

Un-slope-scaled:



Slope-scaled:



Appendix C - Find-Closest Algorithm

The find-closest algorithm is used for ocean calibration routines. Specifically, it is used to find the index corresponding to the closest match of $aValue$ in $aVector$. The inputs are:

- $aVector$ - A vector of positive, real numbers to search through,
- $aValue$ - A value to search closest to.

Using these inputs, we use the following process:

1) Subtract $aValue$ from every element of $aVector$.
    a. Now, the closest element in $aVector$ to $aValue$ is the closest element to 0.
2) Take the absolute value of 1).
    a. Now, the closest element in $aVector$ to $aValue$ is the lowest element.
3) Take the minimum of 2).
4) Find the index in 2) that is equal to 3). If a tie occurs, the first index is selected (for our application, this is preferred).

This algorithm can be iterated to accommodate a vector of $aValue$s to search for within $aVector$. Furthermore, the algorithm can be vectorized using singleton expansion, permutations, and a modulo operation.

Appendix D - Example Parallelization of Ocean Generation

Appendix D.1 - Single Threaded Version

```
%pre-allocate memory
eta = zeros([length(t) length(d) length(oceans)]);
for incd = 1:length(d)
    %distance index. In this example we iterate over distance
    d_ = startD+DSample*(incd-1);
    %eta = A*cos(wt-kd+phi)
    ysub = bsxfun(@times,k,d_);
    ysub2 = bsxfun(@times,w,t);
    ysub = bsxfun(@minus,ysub2,ysub);
    ysub = bsxfun(@plus,ysub,phases);
    ysub = cos(ysub);
    ysub = bsxfun(@times,A,ysub);
    %sum over frequencies
    eta(:,incd,:)= sum(ysub,4);
end
```

Appendix D.2 - SPMD Version

```
%make a distribution scheme to distribute eta over the distance ‘dimension’
codist = codistributor('1d',2);
%tell MATLAB this code should be performed in parallel
spmd
    etad = zeros([length(t) length(d) length(oceans)],codist);
    for incd = drange(1:length(d))
        %distance index. In this example we iterate over distance
        d_ = startD+DSample*(incd-1);
        %eta = A*cos(wt-kd+phi)
        ysub = bsxfun(@times,k,d_);
        ysub2 = bsxfun(@times,w,t);
        ysub = bsxfun(@minus,ysub2,ysub);
        ysub = bsxfun(@plus,ysub,phases);
        ysub = cos(ysub);
        ysub = bsxfun(@times,A,ysub);
        %sum over frequencies
        etad(:,incd,:)= sum(ysub,4);
    end
end
%gather the resulting data from each of the workers
eta = gather(etad);
```

Appendix D.3 - Parfor Version

```
parfor incd = 1:length(d)
    %distance index. In this example we iterate over distance
    d_ = startD+DSample*(incd-1);
    %ysub = A*cos(wt-kd+phi)
    ysub = bsxfun(@times,k,d_);
    ysub2 = bsxfun(@times,w,t);
    ysub = bsxfun(@minus,ysub2,ysub);
    ysub = bsxfun(@plus,ysub,phases);
    ysub = cos(ysub);
    ysub = bsxfun(@times,A,ysub);
    %sum over frequencies
    eta(:,incd,:)= sum(ysub,4);
end
```

Appendix D.4 - pMatlab Version

```
%pre-allocate eta
eta = zeros(length(t), length(d), length(oceans),map3d);
%get the indicies of the portion of the matrix which this node will
%operate over
my_ind_global = global_ind(eta, 2);
%get the local piece of eta to operate on
my_eta = local(eta);
for incd = 1:length(my_ind_global)
    %get the global index to index d with
    gind = my_ind_global(incd);
    %ysub = A*cos(wt-kd+phi)
    ysub = bsxfun(@times,k,d(gind));
    ysub2 = bsxfun(@times,w,t);
    ysub = bsxfun(@minus,ysub2,ysub);
    ysub = bsxfun(@plus,ysub,phases);
    ysub = cos(ysub);
    ysub = bsxfun(@times,A,ysub);
    %sum over frequencies
    my_eta(:,incd,:)= sum(ysub,4);
end
%put the locally computed data back into eta
eta = put_local(eta, my_eta);
%aggregate all of the data into one matrix on the host process
eta_final = oagg(eta);
```